AN EFFICIENT RANKING AND CLASSIFICATION METHOD FOR LINEAR FUNCTIONS, KERNEL FUNCTIONS, DECISION TREES, AND ENSEMBLE METHODS

A Dissertation
Submitted to
the Temple University Graduate Board

In Partial Fulfillment
of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY

---

by
Jesse Glass
May 2020

Examining committee members:

Prof. Zoran Obradovic, Dissertation Advisory Chair, Department of Computer and Information Sciences
Prof. Slobodan Vucetic, Department of Computer and Information Sciences
Prof. Kai Zhang, Department of Computer and Information Sciences
Prof. Edoardo Airoldi, Department of Fox School of Business

ABSTRACT

Structural algorithms incorporate the interdependence of outputs into the prediction, the loss, or both. Frank-Wolfe optimizations of pairwise losses and Gaussian conditional random fields for multivariate output regression are two such structural algorithms. Pairwise losses are standard 0-1 classification surrogate losses applied to pairs of features and outputs, resulting in improved ranking performance (area under the ROC curve, average precision, and F-1 score) at the cost of increased learning complexity. In this dissertation, it is proven that the balanced loss 0-1 SVM and the pairwise SVM have the same dual loss and the pairwise dual coefficient domain is a subdomain of the balanced loss 0-1 SVM with bias dual coefficient domain. This provides a theoretical advancement in the understanding of pairwise loss, which we exploit for the development of a novel ranking algorithm that is fast and memory efficient method with state the art ranking metric performance across eight benchmark data sets. Various practical advancements are also made in multivariate output regression. The learning time for Gaussian conditional random fields is greatly reduced and the parameter domain is expanded to enable repulsion between outputs. Last, a novel multivariate regression is presented that keeps the desirable elements of GCRF and infuses them into a local regression model that improves mean squared error and reduces learning complexity.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Structural algorithms can incorporate relational information between outputs into the prediction, but that is only capable of pulling together or pushing apart predictions. Furthermore, it requires storing additional information for the prediction which can be memory intensive. This strategy is used in graph neural nets, Gaussian conditional random fields, and network lasso. In this dissertation, singular value decomposition is incorporated into Gaussian conditional random fields in order to improve learning time and enable outputs to be repelled away from one another. Memory and speed gains are expounded for the special case where the interdependence of outputs is defined as Kronecker product of two network structures. So as not to be limited to implementation improvements, a novel multivariate output regression optimization that takes the convex combination of graph structures and results in a method that produces lower mean squared error and is less computationally expensive than GCRF is presented.

Alternatively or in addition to incorporating structure into the predictive function, the loss function can also be used to model the interdependence of predictions. When using loss interdependence in the loss function it is possible to incorporate output features into the structure (which will be hidden in testing) and directly optimize multivariate loss functions, and can be even be used in progressive ways – such as reducing racial bias.

Bipartite ranking extends classification problems in order to minimize multivariate losses. The domain of bipartite ranking has a growing body of social applications including student retention and reducing convict recidivism. Appropriately ranking the risk that a person will drop out or commit a crime is more specific than a 0-1 prediction and it better enables appropriate allocation of resources to reduce dropouts and crime.

The Frank-Wolfe algorithm can optimize any loss so long as the parameter domain is convex. Structural support vector machines are a variant of the Frank-Wolfe algorithm limited to the unit sphere loss, but can integrate many types of non-linear constraints. This dissertation thoroughly investigates using Wolfe dual coordinate descent for binary and multiclass SVM losses. Existing literature focuses on GPU optimization of SVM by alterating the SMO algorithm, in contrast, this work demonstrates the GPU gains for Wolfe dual coordinate descent which is a more natural fit.

The Frank-Wolfe algorithm can efficiently optimize various pairwise surrogate losses. Optimizing pairwise loss is equivalent to maximizing AUC. Maximizing AUC improves other metrics such as average precision, discounted cumulative gain, the number of positives before the first negative, and F-1 score. Although the Frank-Wolfe algorithm can optimize those metrics directly, the generalization error tends to be higher when algorithms directly optimize them – optimizing AUC produces better testing results on the wide variety of metrics. Furthermore, the equivalence between AUC and pairwise loss enables AUC optimization to learn faster than alternate ranking metrics.

The subdifferential of the pairwise hinge loss can be applied as sample weights to univariate hinge, logistic, or exponential losses and remain a upper bound of 1 - AUC. The hinge and logistic serve as a trust region approximate algorithms for Frank-Wolfe iterations optimizing pairwise hinge. They have equal gradients and Hessians and they remain within the convex dual parameter domain. The trust algorithm improves the AUC and average precision of RBF kernel predictions and greatly reduces convergence times compared to structural SVM. The linear functions are improved by using the logistic regression as the

trust function. The logit function can be incorporated to turn decision tree probabilities into decision function scores which can then be integrated into the proposed novel variant Frank-Wolfe algorithm – this produces the best AUC, average precision, F-1 without as large an increase in run time as working with pairwise features.

Symbol	Domain	Definition
N	$\mathbb{N}$	number of examples in the data
F	$\mathbb{N}$	number of features in the data
K	$\mathbb{N}$	number of classes in the data
Q	$\mathbb{N}$	number of queries in the data
n^+	$\mathbb{N}$	number of positive labeled examples (binary labels)
n^-	$\mathbb{N}$	number of negative labeled examples (binary labels)
i	$\mathbb{N}$	indexes used for positive labels
j	$\mathbb{N}$	indexes used for negative labels
k	$\mathbb{N}$	indexes used for any class labels
x	$\mathbb{R}^F$	feature vector for an example
X	$\mathbb{R}^{N \times F}$	feature matrix for a set of observations
$\mathcal{X}$		feature space $x \in \mathcal{X}$
w	$\mathbb{R}^F$	linear function
b	$\mathbb{R}$	bias
y	$\mathbb{R}$	decision function score $\langle w, x \rangle + b$
t	$\{ 0, 1 \}$	output label (changes given the application)
$\mathcal{Y}$	output space	$y \in \mathcal{Y}$
f	$f : \mathcal{X} \to \mathcal{Y}$	prediction function
$\mathcal{L}$		Loss
ξ	$\xi : ty \to \mathbb{R}$	margin loss
K	$\mathbb{R}^{N \times N}$	kernel matrix
ϕ	$K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle$	kernel function
$\mathcal{R}$	[0,1]	risk
$\mathbb{P}$	[0,1]	probability
α	$\mathbb{R}^{\lvert\mathcal{Y}\rvert}$	Wolfe dual coefficients
s	$\{0,1\}^{\lvert\mathcal{Y}\rvert}$ $s.t.$ $\sum s = 1$	Frank-Wolfe constraint
λ	$\mathbb{R}^N$	Lagrangian univariate dual coefficients
ω	$\mathbb{R}^N$	sample weights
ρ	$\mathbb{R}^{n^+ \times n^-}$	Lagrangian pairwise dual coefficients
$\mathcal{S}$	$\mathbb{R}^{\lvert\mathcal{Y}\rvert \times N}$	matrix of subdifferentials
Δ	$\mathbb{R}^{\lvert\mathcal{Y}\rvert}$	structural margin vector

Table 1.1: Notation Summary Reference Table

CHAPTER 2

CLASSIFICATION PRELIMINARIES

This chapter reviews the three most common classification losses and the dual form of their L2 regularization. These losses are the primitive units that are used in more complex losses that span the variety of applications detailed in the following section. The foundational optimization algorithms for these losses and their more advanced extensions are also presented. In addition, the algorithms used for hyperparameter optimization are reviewed. The three most common 0-1 classification losses, represented by ξ, are the hinge (2.1), the logistic (2.2), and the exponential (2.3). These losses are tight upper bounds of the 0-1 loss. Each of these losses is defined over each example independently, indexed by i.

$$\xi_k = max\{0, 1 - t_k y_k\} \tag{2.1}$$

$$\xi_k = log(1 + exp\{-t_k y_k\}) \tag{2.2}$$

$$\xi_k = exp\{-t_k y_k\} \tag{2.3}$$

These 0-1 losses are known to to converge to the Bayesian optimal solution (Bartlett et al. (2006)). The L2 normalized primal loss, $\mathcal{L}_p$, for all of the above losses can be represented by equation 2.4 over N examples with hyperparameter C.

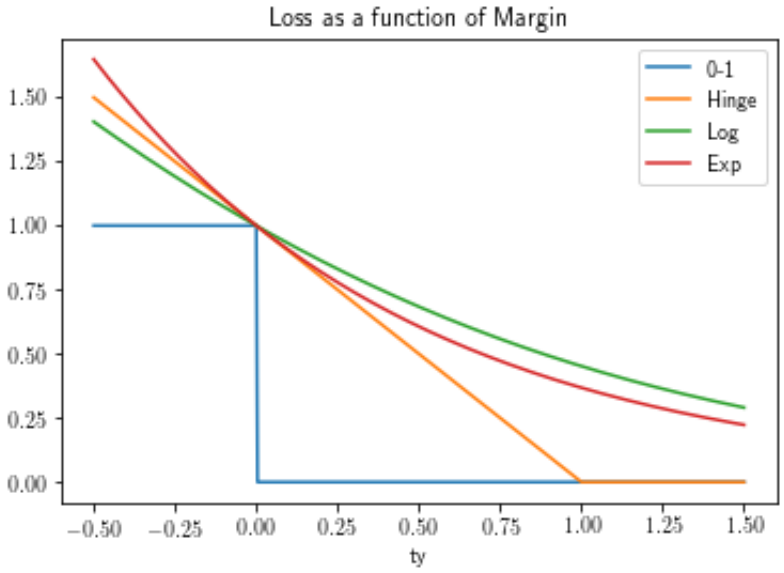

FIGURE 2.1: The various convex upper bounds on 0-1 loss as a function of margin, ty.

$$\mathcal{L}_p = \frac{1}{2}||w||^2 + \frac{C}{N}\sum_i^N \xi_i \tag{2.4}$$

All of the above losses have a Lagrangian dual which utilize the Karush-Kuhn-Tucker generalization of Lagrangian constraints. All the dual losses can also be defined as a Wolfe dual, which has a vastly larger dimension and greater flexibility.

2.1 Problem Types

Binary classification serves as a foundation for bipartite ranking, list ranking, collaborative filtering, multiclass classification, multilabel classification, and ordinal classification. Those extensions to classification are further united by the fact that optimizing pairwise loss has a positive impact on their respective target metrics. Let $\mathcal{X}$ be a feature space and let $\mathcal{X} \subset \mathcal{R}^m$ be an instance of the feature space $x \in \mathcal{X}$. The output changes per task, however, let t represent the labeled output and let $\mathcal{Y}$ be the output space of the predictive model, and let y be an instance of the output space $y \in \mathcal{Y}$. The predictive function f, maps the input space to the output space $f : \mathcal{X} \to \mathcal{Y}$.

Binary Classification

In binary classification, an observed binary label $t \in \{-1, 1\}$. Two univariate metrics commonly used for classification evaluation are 0-1 accuracy and balanced accuracy (p-q). The three losses discussed above are referred to as marginal losses because their error is evaluated with respect to and can be evaluated in terms of ty, where $y \in \mathbb{R}$. The risk, $\mathcal{R}$, for the 0-1 and balanced classification loss are defined below. Let p and q denote class probabilities, $p = \mathbb{P}(t = 1)$ and $q = \mathbb{P}(t = -1)$.

$$\mathcal{R}_{0-1} = \mathbb{P}(ty \leqslant 0) \tag{2.5}$$

$$\mathcal{R}_{p-q} = q \cdot \mathbb{P}(y \leqslant 0 | t = 1) + p \cdot \mathbb{P}(y \geqslant 0 | t = -1) \tag{2.6}$$

Bipartite Ranking

The bipartite ranking setting can be defined in many ways. For the pairwise modeling the feature space is defined by $\mathcal{X} \times \mathcal{X}$ with output prediction space $\mathcal{Y} \times \mathcal{Y}$. This can be further abstracted to the Wolfe dual domain of feature input $\mathcal{X}^N$ and prediction output $\mathcal{Y}^N$. Although the Wolfe dual spaces are required for non-decomposible losses (average

precision, discounted cumulative gain, and F-1 score), this work focuses on AUC which can simply use the definitions from binary classification in order to formulate the AUC risk.

$$\mathcal{R}_{AUC} = \mathbb{P}(y_i \leqslant y_j | t_i = 1, t_j = -1) \tag{2.7}$$

This work focuses on AUC because the algorithms designed specifically for the other rank metrics perform slightly better on training but not consistently better in testing, and they tend to be much slower at learning.

$$AUC = \mathbb{P}(y_i > y_j) \tag{2.8}$$

List Ranking

List ranking is also sometimes known as k-partite ranking. In this setting, the label is any ordinal list set such as $t \in \{0, 1, 2\}$. The output remains a single real valued score. However, the risk is then defined over all pairs of labels.

$$\mathcal{R}_{List} = \mathbb{P}(y_i \leqslant y_j | t_i > t_j) \tag{2.9}$$

The list ranking risk generalizes the AUC risk.

Collaborative Filtering

This problem type is found in the information retrieval domain. Given a set of queries, $\mathcal{Q}$ each query q, $q \in \mathcal{Q}$, is a separate list to be optimized. The loss is defined by pairwise comparisons within a query. The total loss is the average of loss per query. The AUC risk can be defined in the collaborative filtering setting by equation 2.10.

$$\mathcal{R}_{CF} = \mathbb{P}(y_i \leqslant y_j | t_i > t_j \ \& \ i, j \in q) \tag{2.10}$$

Multiclass Classification

This setting can either use one functions per class or one function that returns a vector of scores. This work focuses on the latter. There are K classes and $\mathbf{y} \in \mathbb{R}^K$. $f : x \rightarrow \mathbf{y}$. Let $t \in \{1, ..., K\}$, then the multiclass risk is defined below.

$$\mathcal{R}_{multiclass} = \mathbb{P}(argmax_k(\mathbf{y}) \neq t) \tag{2.11}$$

In multiclass problems, the labels per example need to be ranked appropriately in order for the top ranked class to be the true class.

Other Problem Types

Other extensions of multiclass classification such as taxonomy classification, multilabel classification, and ordinal classification all also benefit from improved ranking. Taxonomy classification are similar to the multiclass with larger penalties for misclassifications which occur with fewer shared ancestors. Multilabel classification is a bipartite ranking problem per example and ordinal classification is a list ranking problem per example.

2.2 Support Vector Machines

The optimization strategy employed by SVMs is a helpful tool for understanding the relationship between regularized convex losses and their constrained dual losses. This understanding enables us to expand upon the capability and efficient of SVMs and other convex losses. It also serves as the foundation for the proposed ranking algorithm. The primal loss (equation 2.4) can be written alternately as the below constrained optimization problem (equation 2.12).

$$\begin{gathered} \mathcal{L}_p = \frac{1}{2}||w||^2 + \frac{C}{N}\sum_k^N \xi_k \\ s.t.\ t_k y_k \geqslant 1 - \xi_k \quad \& \quad \xi_k \geqslant 0\ \forall k \end{gathered} \tag{2.12}$$

When the optimization problem is written in this form, Lagrangian constrained optimization with Karush-Kuhn-Tucker conditions can be applied in order to derive a Lagrangian primal-dual loss with N constraints (equation 2.13).

$$\begin{gathered} \mathcal{L}_{pd} = \frac{1}{2}||w||^2 + \frac{C}{N}\sum_i^N \xi_i + \sum_i^N \lambda_i(1 - t_i y_i - \xi_i) + \frac{C}{N}\sum_i^N \lambda_i' \xi_i \\ s.t. \quad \lambda_i, \lambda_i' \geqslant 0\ \forall i \end{gathered} \tag{2.13}$$

Then take the first order derivatives with respect to the primal loss variables (w, ξ, b) in the constrained optimization (equation 2.13). The optimal dual parameters will occur where the first derivative of the primal loss variables equal zero and the dual coefficients maximize the remaining loss. The dual coefficient maximization is what yields the dual loss and kernels representations 2.14.

$$\frac{\partial \mathcal{L}_{pd}}{\partial w} = w - \sum_i^N \lambda_i t_i x_i = 0 \Rightarrow w = \sum_k^N \lambda_k t_k x_k \tag{2.14}$$

The dual representation yields a prediction function.

$$y_l = \langle w, x_l \rangle = \sum_k^N \lambda_k t_k x_k^T x_l = \sum_k^N \lambda_k t_k K_{kl} \tag{2.15}$$

The linear kernel $K_{ij} = x_i^T x_j$ can be replaced with kernel functions ϕ so that $K_{ij} = \phi(x_i)^T \phi(x_j)$. Instead of calculating ϕ, the transformed matrix can be directly calculated.

$$y_k = \langle w, \phi(x_k) \rangle \tag{2.16}$$

This enables infinite dimensional features transformations such as radial basis functions and also makes scenarios where the number of features exceeds the number of examples more efficient to optimize.

$$y = Xw = K(\lambda \odot t) \tag{2.17}$$

The remaining constraints are the linear constraints (2.18) and the box constraints (2.19).

$$\frac{\partial \mathcal{L}_{pd}}{\partial b} = \sum_i^N \lambda_i t_i = 0 \tag{2.18}$$

$$\frac{\partial \mathcal{L}_{pd}}{\partial \xi_i} = -\lambda_i + \frac{C}{N}\lambda_i' = 0 \Rightarrow \frac{C}{N}\lambda_i' = \lambda_i \Rightarrow \lambda_i \in [0, C\frac{1}{N}] \tag{2.19}$$

The dual representation can then be used to substitute out w.

$$\mathcal{L}_{pd} = \frac{1}{2}\sum\sum \lambda_i \lambda_j t_i t_j x_i x_j + \frac{C}{N}\sum_i^N \xi_i + \sum_i^N \lambda_i (1 - t_i \sum_j (\lambda_j t_j x_j) x_i) - \frac{C}{N}\sum_i^N \lambda_i' \xi_i \tag{2.20}$$

$$\begin{aligned} \mathcal{L'}_d &= -\frac{1}{2}\sum\sum \lambda_i \lambda_j t_i t_j x_i x_j + \sum_i^N \lambda_i \\ s.t.\ &\sum_k^N \lambda_k t_k = 0 \quad \& \quad \lambda_k \in [0, \frac{C}{N}]\ \forall k \end{aligned} \tag{2.21}$$

Typically the negative of the maximization function is taken in order to max the problem a minimization problem and be a loss function.

$$Q = K \odot tt^T$$

$$\begin{aligned} \mathcal{L}_d &= \frac{1}{2}\lambda^T Q\lambda - 1^T\lambda \\ s.t.\ \sum_k^N &\lambda_k t_k = 0 \quad \& \quad \lambda_k \in [0, \frac{C}{N}]\ \forall k \end{aligned} \tag{2.22}$$

Class Balanced SVM

The class balanced SVM assigns separate slack weights to each class. Let the class probabilities be estimated by $p = \hat{\mathbb{P}}(y = 1) = \frac{n^+}{N}$ and $q = \hat{\mathbb{P}}(y = -1) = \frac{n^-}{N}$. Use i for positive label indexing, j for negative label indexing, and k for indexing all examples.

$$\begin{aligned} \mathcal{L}_p &= \frac{1}{2}||w||^2 + Cq\sum_i^{n^+}\xi_i + Cp\sum_j^{n^-}\xi_j \\ s.t. \quad & t_k y_k \geqslant 1 - \xi_k \quad \& \quad \xi_k \geqslant 0\ \forall k \end{aligned} \tag{2.23}$$

$$\begin{aligned} \mathcal{L}_d &= \frac{1}{2}\lambda^T Q\lambda - 1^T\lambda \\ s.t.\ \sum_k^N &\lambda_k t_k = 0 \quad \& \quad \lambda_i \in [0, Cq]\ \forall i \quad \& \quad \lambda_j \in [0, Cp]\ \forall j \end{aligned} \tag{2.24}$$

The only difference in the dual form is that the dual coefficients are constrained to a bounding box that is specific to each class.

Sequential Minimal Optimization

Optimization of SVM requires solving the dual loss defined above (equation 2.21). Unfortunately, quadratic programming is a computationally complex task. A major advancement which

reduced the difficulty of the task was the theoretical finding that the SVM quadratic programming loss could segment the task into subproblems and optimize the quadratic loss over batches of examples and be guaranteed to converge to the optimal solution (Osuna et al. (1997)). This was taken to the extreme of pairs of samples in the famous Sequential Minimal Optimization (SMO) (Platt (1998)). When examining pairs of examples the quadratic programming task has a closed form solution at every step. It has the additional benefit that it keeps kernel calculations to a minimum. The speed of the SMO is determined by how well it selects the pairs of examples to optimize over. The candidate pairs are referred to as the working set.

Working Set Selection

In the original SMO, a heuristic measure of loss was used in order to select which examples would be investigated. A formal presentation of the ideal working set was developed (Keerthi et al. (2001)). It has been proven that the optimal pair can be selected by utilizing second order information. However, that would be onerous and defeat the purpose of speeding up the algorithm. A fast methodology which only uses second order information in order to select the second example while using heuristics for the first has proven to be the fastest strategy (Fan et al. (2005)).

Graphical Processing Units (GPUs) are capable of efficiently running hundreds of operations in parallel. In the domain of SVM optimization, GPUs have been integrated into the working set selection in order to improve speed (A. Athanasopoulos (2011)). However, there exists an entirely different optimization strategy – dual coordinate descent (Hsieh et al. (2008)). After the introduction of the Wolfe dual, we implement and evaluate a Wolfe dual coordinate descent SVM optimization. We follow up with theoretical and experimental evidence that the coordinate descent optimization is more well suited to utilize GPUs than SMO.

2.3 Frank-Wolfe Algorithm

The Frank-Wolfe algorithm, also known as the conditional gradient method, can optimize any loss over a convex parameter domain. One such convex parameter domain is the set of box constraints defined over λ for the SVM model. But, the joint constraints of all the examples is not a cube. There are interdependencies in the constraints which create a domain with a larger number of facets. The convex parameter domain can more clearly be defined by a larger representation of the Wolfe dual (equation 2.28) which assigns a parameter to each constraint. The Wolfe dual parameters are represented by α. When the loss is differentiable the constraints equal the subgradient. The Frank-Wolfe algorithm minimizes the loss $\mathcal{L}$ by taking the centroid of the constraints. The prediction is guaranteed to stays within the constraint domain $\mathcal{D}$ given that the domain is convex and the solution is a convex combination of constraints.

$$min_{\alpha \in \mathcal{D}} \mathcal{L}(\alpha) \tag{2.25}$$

The Frank-Wolfe algorithm minimizes a convex function over convex parameter space (equation 2.25) by using the first order Taylor series expansion in order to approximate the optimal solution with respect to the original loss. The algorithm requires a subroutine which discovers the constraint which will minimize the loss most given the current solution. The structural SVM is a special case of Frank-Wolfe where the constraint is easily calculated by structural hinge.

The convex combination coefficient γ, $\gamma \in [0, 1]$ provide the parameter update rule.

$$\alpha^k = (1 - \gamma)\alpha^{k-1} + \gamma s = \alpha^{k-1} + \gamma(s - \alpha^{k-1}) \tag{2.26}$$

This is also equal to the current solution plus the search direction, $\gamma(s - \alpha^{k-1})$. The search direction can be optimized with the 1st order Taylor expansion or a Trust Region

Algorithm 1 Frank-Wolfe Algorithm

$\alpha^{(0)} = \vec{0}$ or any $\alpha^{(0)} \in \mathcal{D}$
$k = 0$
repeat
  $\gamma = 2/(2 + k)$
  $s = argmin_s \langle \nabla \mathcal{L}(\alpha^k), s \rangle$
  $\alpha^{k+1} = (1 - \gamma)\alpha^k + \gamma s$
  $k = k + 1$
until dual gap convergence: $\langle \nabla \mathcal{L}(\alpha^{k+1}), \alpha^{k+1} - s \rangle \leqslant \epsilon$

optimization. The successive losses are estimated by the first order Taylor expansion.

$$\mathcal{L}(\alpha^{(k)}) \geqslant \mathcal{L}(\alpha^{(k-1)}) + \langle \nabla \mathcal{L}(\alpha^{(k-1)}), \gamma(s - \alpha^{(k-1)}) \rangle \tag{2.27}$$

The Frank-Wolfe algorithm finds the constraint which maximizes the impact of the first order Taylor expansion. The optimal constraint at each iteration is equal to the subderivative if the loss is differentiable.

A simple form of the algorithm is presented in algorithm 1. In it, the step size, $\gamma \in [0, 1]$, can leverage a simple strategy which amounts to the centroid of the iterative constraints, but a line search to find the optimal γ could be used instead. For structural SVM, the line search can be solved analytically at each step.(Lacoste-Julien et al. (2013)) In figure 2.2, the initial estimate oscillates toward one of two constraints which are nearest the equilibrium.

Wolfe dual

When optimizing a structural SVM loss, the Frank-Wolfe algorithm can be optimized with a dual coordinate descent algorithm (Lacoste-Julien et al. (2013)). The structural SVM loss can be represented with a Wolfe dual loss. The Wolfe dual form includes a constraint for every potential gradient direction. This generates an exponential explosion in the number of constraints. However, the number of constraints with a non-zero dual coefficient remains low. Represent each examples contribution to all possible constraints by the matrix $\mathcal{S}$.

$$\begin{aligned} \mathcal{L}_d =& \frac{1}{2}\alpha^T \mathcal{S}^T Q \mathcal{S}\alpha - \Delta^T \alpha \\ & s.t. \sum_{m=1}^{|\mathcal{Y}|} \alpha_m = C \end{aligned} \tag{2.28}$$

Let w represent a linear weight vector; X represent a matrix of features; t, binary labels in $\{-1, 1\}$; α, the Lagrangian coefficients for corresponding to slack associated with an individual data point; and C represent a set of valid constraints where $col_i(C) = \mathbf{s}^{(i)}$. The variables which SSVM optimizes are characterized by their relation to the dual representation of a linear function. (equation 2.29).

$$w = (t \odot \lambda)^T X = (t \odot \mathcal{S}\alpha)^T X \tag{2.29}$$

In structural models, independent examples can be modeled independently or jointly. Typically in structural SVM, if an example can separated from the others, it is. Whereas the coordinate descent optimization averages the margin of all examples together at every step.

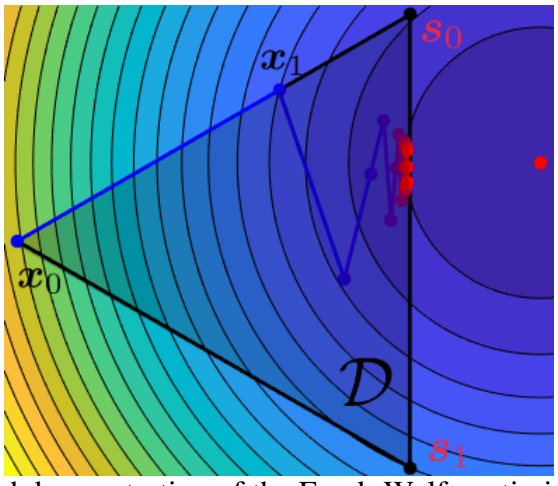

FIGURE 2.2: Simplified demonstration of the Frank-Wolfe optimization (Pedregosa et al. (2018))

2.4 Structural Support Vector Machines

Structural SVM extends the soft margin SVM constraints to structural slack variables (Tsochantaridis et al. (2005); Taskar et al. (2004)). The primal structural SVM loss looks identical to the SVM primal form of the loss (equation 2.30). However, the slack variables are generalized hinge loss (equation 2.31). There are an exponential number of potential outputs $|\mathcal{Y}|$. In classification, the number of constraints can be either N or 2^N and in bipartite ranking there are $N!/(n^-! \times n^+!)$ while in multiclass classification there can be either K^N or $\sum_i^N K$.

$$\begin{aligned} & min \frac{1}{2}||w||^2 + C\xi \\ s.t.\ & \langle w, \partial\Psi(X, t, t_m)\rangle \geqslant \Delta_m - \xi \quad \forall m \end{aligned} \tag{2.30}$$

The joint feature function $\Psi(X, t)$ captures the interdependent relationship between inputs, X, and outputs, t and is specific to a given task. The loss is a non-negative real valued number $\Delta_i \in \mathbb{R}^+$ and it represents the optimization metric. $\lambda \in \mathbb{R}^+$ is the L2 regularization weight.

$$\xi = |max_m(\Delta(t, t_m) + \langle w, \Psi(X, t_m)\rangle) - \langle w, \Psi(X, t)\rangle|_+ \tag{2.31}$$

The structural slack includes an inner max function which is referred to as the maximization oracle or augmented loss. The vector $\mathbf{s}$ required by the Frank-Wolfe algorithm is the same value as is derived by the maximization oracle for structural slack variables (Lacoste-Julien et al. (2013)). The number of potential interaction of all the inputs and outputs within an example structure is exponentially larger than the number of data points within the example. For instance, a single structural slack for $\{0, 1\}$ loss has been shown to be the same value as the sum of all the slacks in binary SVM at a given solution – they differ by a constant factor of two (Joachims (2005)).

$$\xi_m = |\Delta_m - \langle w, \partial\Psi(X, t, t_m)\rangle|_+ \tag{2.32}$$

The constraint s in the Frank-Wolfe algorithm directly corresponds to the subgradient. In the coordinate descent optimization s is a one-hot vector where as if each example is independent s is an N-hot vector.

$$\partial\Psi(X, t, t_m) = (t \odot s_m\mathcal{S})X \tag{2.33}$$

The subgradient of the ξ component equals $-\partial\Psi(X, t, t_i')$. (Lacoste-Julien et al. (2013))

$$\langle w, \partial\Psi(X, t, t_m)\rangle = \langle w, \Psi(X, t_m)\rangle) - \langle w, \Psi(X, t)\rangle \tag{2.34}$$

Thus this maximization oracle is hidden within the internal mechanism of the subgradient. The slack can be rewritten concisely as in equation (2.32). Figure 2.3 compares the

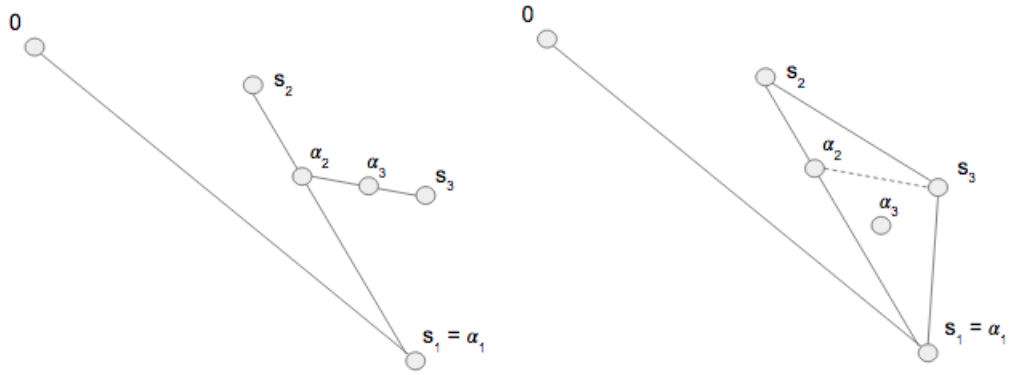

FIGURE 2.3: Frank-Wolfe optimization: Coordinate Descent (left) and Structural Support Vector Machine (right).

iterative search space within the constraints for the classical Frank-Wolfe and the SSVM in the scenario where both are using the one-hot variation of s. The are an exponential number of constraints $s \in \mathcal{D}$ with respect to the number of data points. SSVM is a quadratic programming algorithm which introduces a Lagrangian coefficient α^i for each

successively uncovered constraint, $\mathbf{s}^{(i)}$. It optimizes a convex upper bound of the loss f over the simplex of currently active constraints.

The optimal solution is sparse with respect to the number of active constraints ($\alpha_i \neq 0$). Most are never calculated and integrated into the optimization. The dual coefficients in structural SVM, α, are extremely sparse. Although for complex output structures the resulting λ may be sparse, for the $\{0, 1\}$ loss and rank metric losses, structural SVM produces a dense vector of Lagrangian coefficients λ. The equivalence of Lagrangian coefficients is presented in equation 2.29. The dense λ is not a burden for linear functions which enables linear SVM and linear SSVM to scale in linear time complexity and efficiently handle extremely large sparse feature sets (Joachims (2006); Joachims et al. (2009)).

2.5 Gradient Boosting

Gradient boosting can apply to either log-loss or exponential loss. This dissertation focuses on logit-boost. A gradient step for these losses is the residual weighted expectation of the features. The residual for the logistic regression is $r_i = 1 - p_i$ if $t_i = 1$ and $r_i = p_i$ otherwise. The gradient update rule is weighted linear combination of features $\nabla\mathcal{L} = \sum_i r_i x_i$. A model with the same gradient would be represented by the residual weighted expectation of predictions.

Rather than the linear combination of features found in linear function optimization each step can be replaced by a residual sample weighted decision tree – more precisely, a decision stump. Gradient boosting combines the decision function outputs of each step in order to make a prediction. The loss is then used to determine the residual at each step which is in turn used to optimize the next tree.

Let p represent a decision tree's output. Where p is the probability that $t = 1$ for a given example in a $p \in [0, 1]$. Let η represent a regularization coefficient $\eta \in (0, 1)$. Gradient boosting feeds a regularized probability to return a decision function score y which would be the equivalent of the linear function score $y = \langle w, x \rangle$ where $p = \sigma(y)$, given sigmoid function (σ). Below is the regularized logit function.

$$y = \frac{log((1-\eta)p + \eta\frac{1}{2}))}{log(1 - ((1-\eta)p + \eta\frac{1}{2}))} \tag{2.35}$$

2.6 Hyperparameter Optimization

Most models require some choice of hyperparameters. There are several common ways to try to select the best hyperparameters. They all involve evaluating the output of the training algorithm algorithm on validation data set. Often the hyperparameters are manually selected or evaluated over a grid search of potential values. However, in this dissertation, a Bayesian strategy is employed.

2.6.1 Bayesian Sequential Model-Based Optimization

Bayesian model-based optimization methods build a probability model of the output metric in order to select the next set of hyperparameters to evaluate. SMBO is a Bayesian optimization that uses prior observations to determine which parameter combinations are promising, worth exploring, or to be avoided.

Algorithm 2 SMBO Algorithm with Expected Improvement

Input: prediction scores $\hat{Y}$, labels Y, Θ^0
repeat
  randomly partition data into cross validation folds
  repeat
    fit the predictive model on the training cv folds given Θ as hyperparameters
    evaluate $\mathcal{L}(f)$ on validation cv fold
  until cv folds enumerated
  model the expected $\mathcal{L}$ with Θ as features and historical $\mathcal{L}$ as labels
  select next Θ by expected improvement
until AUC variance of last 5 samples $\leqslant \epsilon$

The optimization used in experiments utilizes a Gaussian process regression. The inputs are the range of allowed hyperparameters and the scale in which they are optimized. For instance, regularization parameters are scaled 10^x and the Gaussian process optimizes over x. Hyperparameters are initialized with common defaults. For each set of hyperparameters, AUC or MRR is averaged over a 3-fold cross validation on the training set. Gaussian process regression models the mean and variance of AUC over the hyperparameter space.

Expected Improvement is used to select the next hyperparameter to evaluate. Expected Improvement balances exploration and expectation of AUC by searching high variance and high mean value hyperparameter sets (Snoek (2012)). The Bayesian hyperparameter selection considered converged when the variance of returned AUC of the last 5 samples is less than a given tolerance ($\epsilon < 1e-3$). Once converged, a final Gaussian Process is fit to all the data and the maximum mean valued hyperparameter set is chosen.

Expected Improvement

Given the optimal observed loss, $\mathcal{L}^*$, and Gaussian distribution, $\mathcal{N}$, the below utility function can be maximized by integrating over the parameter space (equation 2.36).

$$max\{0, \mathcal{L}^*(\Theta) - \mathbb{P}(\mathcal{L}|\Theta)\}$$

$$EI = \int_{-\infty}^{\mathcal{L}^*} (\mathcal{L}^*(\Theta) - \mathbb{P}(\mathcal{L}|\Theta))\mathcal{N}_{\mathcal{L}}(\mu(\Theta), K(\Theta, \Theta))\partial\Theta \tag{2.36}$$

The expected improvement utility function makes sure to search hyperparameters with a large variance and expected value.

2.6.2 Gaussian Process Regression

A Gaussian process regression predicts both mean μ and variance σ^2. Using an identity matrix, I, and a Matern kernel matrix, K that are defined by T, the number of prediction points, and N, the number of current observations. K_{TN} represents that Matern kernel matrix of size $T \times N$. Given the observations $\mathcal{L}$ and a kernel matrix defined over the hyperparameter space, the formulas for the Gaussian process regression are defined below. The process makes a large number of predictions which are randomly spaced over the hyperparameter domain.

$$\mu_T = K_{TN}(K + \sigma_{\mathcal{L}}^2 I)^{-1}\mathcal{L} \tag{2.37}$$

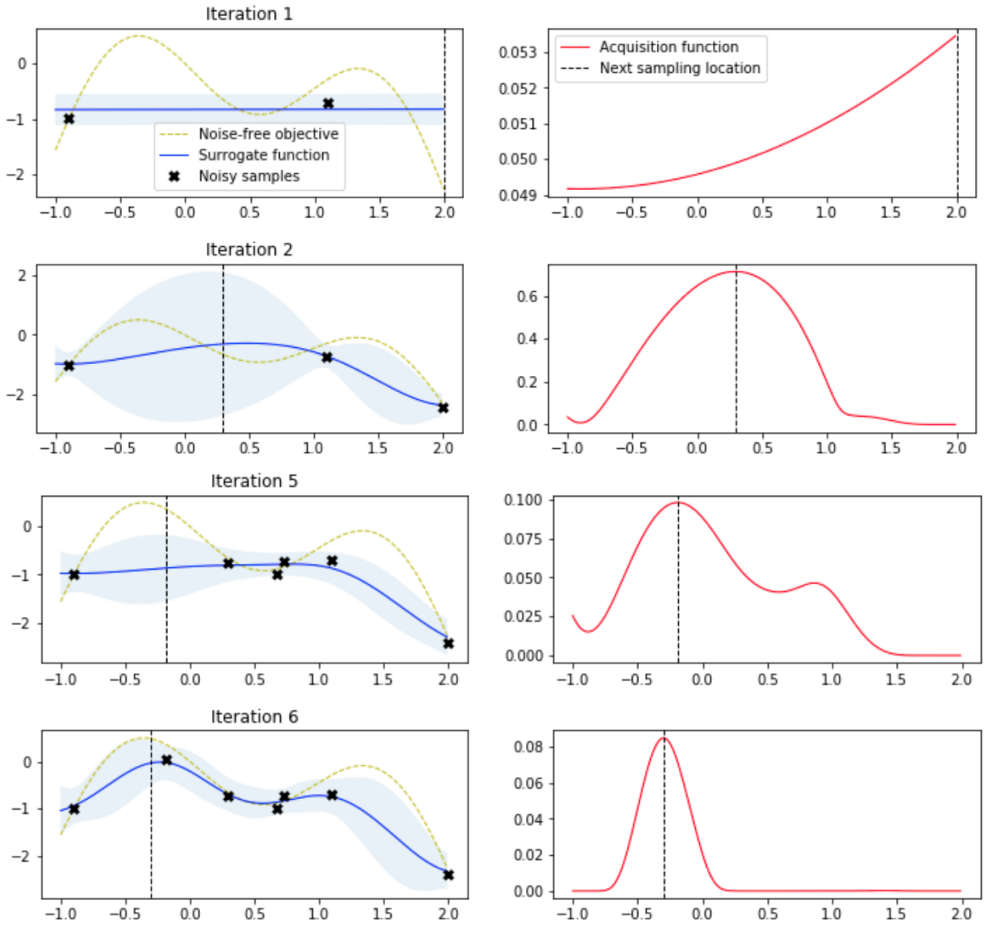

FIGURE 2.4: Simplified demonstration of the Gaussian Process optimization (Krasser (2018))

$$\sigma_T^2 = K_T - K_{TN}(K_N + \sigma_{\mathcal{L}}^2 I)^{-1} K_{NT} + \sigma_{\mathcal{L}}^2 I \tag{2.38}$$

The Gaussian process outputs feed into the expected improvement formula which selects a hyperparameter sample which is fit using the chosen algorithm on the cross validation layers which outputs a $\mathcal{L}$ and the SMBO algorithm continues. In figure 2.4, the acquisition is pictured on the right with the sample value as a dashed line on the left graphic across the range of expected values. The variance and mean the Gaussian Process Regression are in blue.

CHAPTER 3

PAIRWISE LOSSES AND COORDINATE DESCENT

AUC is equal to the number of correctly ordered pairs of positive and negative labels (Cortes and Mohri (2003)). Optimizing a 0-1 convex loss over pairs of instances is a convex upper bound of 1 - AUC. Rather than transforming the data, it is possible to directly optimize over the pairwise relationships. In fact the gradient (and subgradient) calculations can be computed much faster when exploiting relationships between the pairwise derivatives and non-transformed data X.

3.1 Related Work for AUC Optimization

Bayesian consistency is considered a valuable property of classification surrogate losses. But, just because a loss is 0-1 classification consistent does not mean that the application of the loss to pairwise features is AUC consistent. The pairwise logistic and exponential are AUC consistent while the pairwise hinge is not (Gao and Zhou (2015); Zhou and Liu (2010)). This has served as motivation for methods which use stochastic primal updates for bipartite ranking that are designed for online learning (Gao (2013); Ying et al. (2016); Liu (2018)). Similar methods have sacrificed being an upper bound of AUC loss in order to improve run time (Li et al. (2014)).

The OPAUC, one of the commonly referenced online learning methodologies is evaluated in the bipartite ranking linear function experiments. Using the published guidelines for hyperparameter selection the OPAUC did not outperform logistic regression or Frank-Wolfe coordinate descent which are already possible to optimize in the online setting. The algorithm was evaluated on the data sets which it published results on and validated those experiments. However, those data sets were trivially easy to perform well on AUC. That body of research achieved similar performance to logistic regression in those experiments – at most half a point of AUC higher. But, in our extended body of research logistic regression performs much better.

Additionally, experiments conducted in order to validate those methodologies compare to standard logistic regression and not balanced logistic regression. In 75% of the bipartite ranking experiments the standard logistic regression and balanced logistic regression produce equal AUC. In the other 25%, balanced logistic regression wins by two points of AUC, which is a greater improvement than those methodologies show.

It has been established that balanced 0-1 loss is a tighter bound on AUC loss than 0-1 loss, but that this does not hold for surrogate losses (Kotlowski et al. (2011)). However, the experimental results in this dissertation show that empirically balanced surrogate losses improve AUC. Theoretically, applying the novel optimization algorithm proposed in this work suggests that a balanced univariate is the optimal univariate loss.

The novel ranking and classification algorithm presented in this dissertation build upon the foundation that Frank-Wolfe coordinate descent can optimize any structural SVM problem (Lacoste-Julien et al. (2013)). It does not always achieve the same results, however it often does, and provides learning time improvements in terms of integrating GPUs and in terms of kernel functions for structural SVM losses.

The structural SVM optimization has a slow run time for the pairwise hinge loss when fitting kernels. Various strategies have been proposed to resolve it (Chen et al. (2017); Kuo et al. (2014)). The presentation on kernels contained here reveals the heart of the problem

and the easiest way to resolve it. This provides a more elegant and more efficient solution than existing strategies.

Several research papers worked on the discovery of and proof of correctness of algorithms which produce the next iteration of Frank-Wolfe constraint, s, for discontinuous rank metrics (Yue et al. (2007); Agarwal and Narasimhan (2013); Narasimhan (2013)). Those methodologies are a subprocess of the structural SVM optimization and do not alter the fundamental optimization procedure – the trust region optimization presented in this dissertation does. The novelty of this is demonstrated by it being one of the few to integrate ensembles of decision trees into the Frank-Wolfe algorithm.

3.2 Pairwise Loss

Rather than generating pairwise features and labels it is possible to evaluate pairwise losses and even calculate their subgradients with respect to the outputs alone. This saves time in predictions and can also save time in terms of learning. Some algorithms have more time savings in terms of learning than others.

$$\xi_{ij} = max\{0, 1 - y_i + y_j\} \tag{3.1}$$

$$\xi_{ij} = log(1 + exp\{-y_i + y_j\}) \tag{3.2}$$

$$\xi_{ij} = exp\{-y_i + y_j\} \tag{3.3}$$

The primal loss is then defined over a quadratic number of losses.

$$\begin{gathered} min_w \frac{1}{2}||w||^2 + \frac{C}{N^2}\sum_i^{n^+}\sum_j^{n^-}\xi_{ij} \\ s.t.\ \langle w,\phi(x_i) - \phi(x_j)\rangle \geqslant 1 - \xi_{ij} \quad \& \quad \xi_{ij} \geqslant 0 \end{gathered} \tag{3.4}$$

$$\rho_{ij} \in [0, \frac{C}{N^2}] \quad \forall\ i, j \tag{3.5}$$

$$w = \sum_i^{n^+}\sum_j^{n^-}\rho_{ij}(x_i - x_j) \tag{3.6}$$

$$y_k = \sum_i^{n^+}\sum_j^{n^-}\rho_{ij}(x_i - x_j)^T x_k = \sum_i^{n^+}\sum_j^{n^-}\rho_{ij}(K_{ik} - K_{jk}) \tag{3.7}$$

Each has a quadratic number of constraints and corresponding Lagrangian coefficients ρ. However, the subdifferentials directly with respect to the univariate features. The pairwise logistic has a gradient and so gradient descent can be used. However, the pairwise logistic residuals require calculating the pairwise distance of all decision function outputs

of the opposite classes. After taking the sigmoid function applied to those distances and summing over all pairs containing that example (eq. 3.8), the gradient is then a linear function (eq. 3.9).

$$r_i = \sum_j \sigma(y_j - y_i) \tag{3.8}$$

$$\nabla \xi_i = r_i x_i \tag{3.9}$$

The hinge loss has a subgradient but no gradient. Each data sample has a linear subdifferential and it contributes to the loss if a negative sample's prediction score is within the margin of a positive sample's prediction score. Thus, the final subgradient is the sum of all data points of the opposite class that are within the margin of the wrong side of another reference data point. The formula for counting this is detailed below.

$$c_{ij} = c_{ji} = \begin{cases} 1 & \text{if } 1 + y_j > y_i \quad \forall\, t_j = -1 \;\&\; t_i = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

The positive and negative examples no longer need to be indexed independently.

$$\begin{aligned} \mathcal{S}_{m,j} &= \sum_{\forall i} c_{ij} \quad t_i = 1 \;\&\; t_j = -1 \\ \mathcal{S}_{m,i} &= \sum_{\forall j} c_{ij} \quad t_i = 1 \;\&\; t_j = -1 \end{aligned} \tag{3.11}$$

Given that the number of swaps to transform the current ordering to a perfect AUC is equal to $\sum_j c_j$. The loss Δ is an the exact measure of AUC loss given the margin, $\Delta = \sum_j c_j$. The resulting structural slack for hinge AUC can be written as in equation 4.12. The subdifferentials matrix $\mathcal{S}$ is designed so that selecting a column m and taking the product with the feature set X produces the subderivative.

$$\partial \Psi(X, t, t_m) = \sum_k \mathcal{S}_{m,k}\; t_k x_k \tag{3.12}$$

$$\Delta(t, t_m) = \sum_k \mathcal{S}_{m,k} \tag{3.13}$$

$$\partial\Psi(X, t, t_m) = (t \odot s_m \mathcal{S})X \tag{3.14}$$

3.2.1 Multiclass Classification

Multiclass algorithms must rank the positive class above the false classes. In the binary setting this is a simple matter of the margin, ty, being greater than zero. However, the problem is more complicated in the multiclass setting. The tightest possible hinge upper bound of multiclass accuracy is what is referred to as the Winner-Take-All loss (equation 3.15) presented in Crammer and Singer (2001).

$$\xi = |1 + max_{k \neq t} y_k - y_t|_+ \tag{3.15}$$

In this scenario, only the highest rank false class is penalized and it is only penalized if it is within a margin of 1 of the true class score. Evaluation with respect to all other classes was introduced by Westin and Watkins (1999) and is equal to evaluating the AUC of each example with respect to the other classes.

$$\xi = \sum_{k \neq t} |1 + y_k - y_t|_+ \tag{3.16}$$

The AUC multiclass optimization (equation 3.16) is believed to have better generalization properties but tends to be slower (Doğan et al. (2016)). The coordinate descent algorithm inspected in this dissertation can optimize the AUC multiclass loss efficiently.

3.3 Coordinate Descent Experiments

In existing GPU speed ups of binary SVM optimization each point is already being evaluated in order to determine the working set. After the evaluation step, the coordinate descent only requires straight forward follow up calculations. That presents an opportunity for GPU SVM optimization. The performance of CPU coordinate descent and SMO are compared on the benchmark data sets. Then the CPU and GPU are compared on large data sets and synthetic data sets in order to understand at what point the trade off of writing data to the GPU is worthwhile.

It has been established that the Wolfe dual coordinate descent can optimize the various hinge losses. Given that there are theoretical benefits from maximizing AUC, these experiments will evaluate the time and accuracy performance of optimizing within example AUC for multiclass classification with Frank-Wolfe coordinate descent (FWCD).

Evaluation. These experiments inspect the following metrics: 0-1 accuracy, balanced accuracy, and F-1 score for the binary labels. Accuracy and mean reciprocal rank (MRR) were evaluated in the multiclass setting. The fit time of a single round of learning is reported alongside the total learning time which includes the hyperparameter optimization.

Statistical significance. Significance is determined by the paired difference test. For every one of the 30 trials a pairwise difference in each target metric is calculated. The pairwise difference is either accepted or rejected as significantly different from zero. The pairwise difference is more informative than a difference in means because the data set causing underlying variance in the two models is accounted for, which is appropriate given it is shared by both models. The trust region method is notated as a statistically significant improvement at the various confidence intervals: * for 90%, ** for 95%, *** for 99%, and **** for 99.9%. Statistically significant lower performance is notated with parentheses (*), (**), (***), (****).

Implementations. LibSVM is used for the SMO optimization. (Chang and Lin (2011))

The Frank-Wolfe coordinate descent (FWCD) was implemented in python. The GPU variant utilizes pytorch.

Data sets. The data used are publicly available from the UCI data repository (Dheeru and Karra Taniskidou (2017)) and are often used for comparing algorithms. Complete information of the number of features in the data set and percent of positive labels are presented in Table 5.3.

3.3.1 Binary Classification Experiments

The FWCD linear fit times are smaller for all but the Ion and Yeast data sets which have trivial fit times for the SMO algorithm. GP fit time tracks similarly. The methods produce similar results, but the FWCD is less likely to run for a very long time.

Table 3.1 is a comparison of the Frank-Wolfe dual coordinate descent algorithm (FWCD) and sequential minimal optimization (SMO) for linear function SVM optimization. If convergence was too slow for the SMO method a single split with default parameters was run and the run time is entered as N/A.

The FWCD outperformed the SMO optimization when using radial basis function kernels on all except the Boston data set. The SMO did converge more quickly, so it is potentially influence by stopping criteria tolerance. Only when an algorithm was faster and more accurate can we confidently conclude superior performance.

Data Set	Method	0-1	p-q	F-1	Learn	Run
	FWCD	87.3	84.1	80.2	0.49	14.6
Ion	SMO	86.2	82.5	78.1	0.01	3.6
N = 351		***	****	****		
	FWCD	92.5	49.9	0.0	0.73	17.2
Boston	SMO	92.6	50.	0.0	0.97	25.5
N = 506		()	()			
	FWCD	85.3	85.8	84.5	0.39	9.8
Credit	SMO	85.2	85.3	83.9	149.3	530.0
N = 690			*	**		
	FWCD	77.	71.9	62.4	0.16	9.1
Pima	SMO	77.1	72.1	62.7	5.64	98.1
N = 781		()	()	()		
	FWCD	74.	67.1	53.9	0.45	14.0
Student	SMO	74.4	67.	53.5	124.2	516.5
N = 1008		()				
	FWCD	68.4	50.1	0.006	0.59	9.8
Yeast	SMO	68.5	50.	0.0	0.01	1.9
N = 1484		(*)				
	FWCD	92.7	92.2	90.7	0.40	N/A
Spam	SMO	93.4	92.9	91.6	198.6	N/A
N = 4601						
	FWCD	95.4	81.7	74.9	0.92	N/A
Pg-Blox	SMO	95.1	85.6	76.3	268.6	N/A
N = 5473						
	FWCD	92.1	67.7	47.7	0.561	17.6
IJCNN	SMO	92.5	65.8	45.7	9.444	68.3
N = 49998		()				

Table 3.1: Linear Function SVM for Binary Classification optimized with Frank-Wolfe and Sequential Minimal Optimization

Data Set	Method	0-1	p-q	F-1	Learn	Run
	FWCD	95.	94.4	93.	0.096	35.3
Ion	SMO	94.5	93.5	92.1	0.0	27.2
N = 351		***	***	***		
	FWCD	93.1	51.3	2.8	0.114	7.2
Boston	SMO	93.3	51.2	3.5	0.0	5.2
N = 506		(**)		()		
	FWCD	83.1	83.3	80.1	0.109	28.8
Credit	SMO	65.7	64.9	53.4	0.015	6.9
N = 690		****	****	****		
	FWCD	76.5	71.4	61.6	0.11	15.8
Pima	SMO	66.3	56.3	24.9	0.01	4.3
N = 781		****	****	****		
	FWCD	71.3	61.3	39.7	0.122	8.6
Student	SMO	67.4	50.1	1.5	0.02	4.9
N = 1008		****	****	****		
	FWCD	73.1	64.3	46.2	0.271	14.3
Yeast	SMO	71.8	60.4	33.8	0.021	4.0
N = 1484		***	***	***		
	FWCD	89.7	88.5	85.4	1.563	64.4
Spam	SMO	87.7	87.	84.	0.406	42.0
N = 4601		**				
	FWCD	96.3	88.2	81.4	2.168	41.4
Pg-Blox	SMO	93.1	69.3	52.9	0.379	9.0
N = 5473		****	****	****		
	FWCD	92.	71.9	48.4	131.67	376.6
IJCNN	SMO	96.2	82.2	73.1	8.80	83.1
N = 49998		(****)	(****)	(****)		

Table 3.2: Radial Basis Function SVM for Binary Classification optimized with Frank-Wolfe and Sequential Minimal Optimization

3.3.2 *Graphical Processor Parallelization*

When the number of examples exceeds 20,000 data points the developed GPU coordinate descent optimizations outperforms its identical CPU optimization. The data is generated by inputing N and fixing $F = 100$. The features X are randomly generated. A weight vector is randomly generated, $w_i \in [-\frac{1}{2}, \frac{1}{2}]$. Mean predictive score is $\mu = \bar{x}w$ and r is randomly uniform generated. Labels were then calculated $t = (Xw + r > \mu + .5)$.

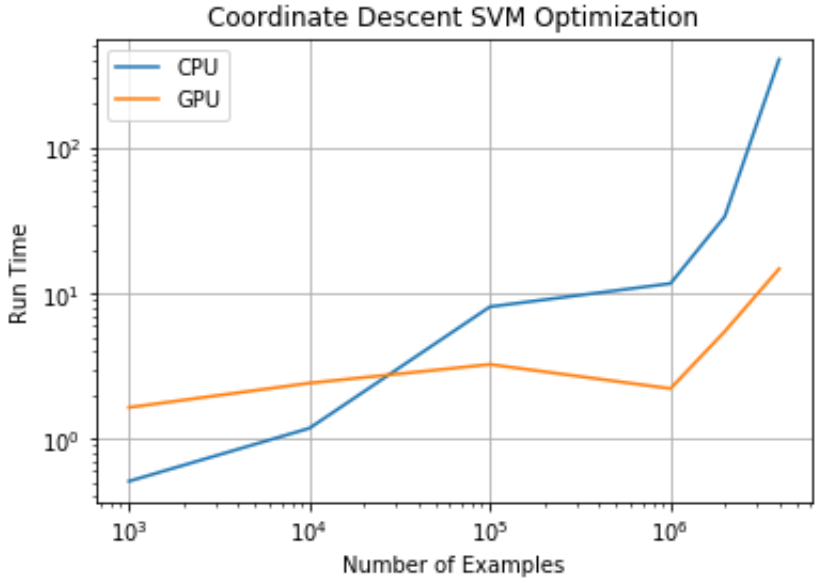

FIGURE 3.1: Comparison of CPU and GPU run times for Coordinate Descent SVM Optimization

Data Set	Split	Design	Accuracy	AUC	Time
IJCNN	Train	CPU	91.0	85.6	1.0
N = 5e+4		GPU	91.3	85.8	1.4
F = 22	Test	CPU	91.0	85.7	1.0
p = 10%		GPU	91.5	85.8	1.4
Real-Sim	Train	CPU	99.6	99.8	62.5
N=7.2e+4		GPU	99.6	99.8	11.3
F = 2.1e+4	Test	CPU	95.6	98.5	62.5
p = 31%		GPU	94.9	98.0	11.3

Table 3.3: Demonstration of GPU Speed Up on Real Data

N	Design	Accuracy	AUC	Time
1e+3	CPU	83.4	92.9	**0.51**
	GPU	83.5	92.9	1.64
1e+4	CPU	88.1	96.1	**1.18**
	GPU	88.0	96.1	2.41
1e+5	CPU	86.4	94.7	8.07
	GPU	87.0	95.2	**3.24**
1e+6	CPU	84.8	93.4	11.65
	GPU	84.8	93.4	**2.21**
2e+6	CPU	84.9	93.5	33.70
	GPU	84.9	93.5	**5.46**

Table 3.4: Demonstration of GPU Speed Up on Synthetic Data

3.3.3 Multiclass Classification Experiments

These experiments compare the various optimization strategies of closely related hinge losses. The one-vs-one, the WTA, and the AUC are all separate losses. Then there are different optimizations of the WTA loss: Crammer-Singer and Structural SVM. Statistical significance is evaluated within a kernel. The WTA-CS only has a linear optimization implementation easily accessed.

Implementations. LibSVM the underlies one-vs-one (OVO) method (Chang and Lin (2011)). The Winner-Take All (WTA CS) solver uses a Liblinear algorithm (Fan et al. (2008)). The WTA SSVM uses the structural SVM is a cutting plane solver (Tsochantaridis et al. (2005); Joachims (2005)) . The AUC Frank-Wolfe coordinate descent (FWCD) was implemented in python.

Table 3.5 is a comparison of multiclass SVM optimizations: One-vs-One (OVO), Crammer-Singer (CS), Structural SVM (SSVM), Frank-Wolfe Dual Coordinate Descent (FWCD). Linear FWCD typically performs worse, although there are some exceptions. RBF FWCD however typically performs better.Sometimes RBF OVO befores best and fastest. The RBF kernels outperform the linear kernels. SSVM is too slow to to be worthwhile to run on Segmentation, Abalone, and Page-Blocks. FWCD linear did the worst at Iris and Glass. WTA CS performed very poorly on Abalone and Page-Blocks. WTA SSVM underperformed with respect to WTA CS, which is expected to produce similar solutions. The loses are accuracy and mean reciprocal rank (MRR). Fit is the learn time for the algorithm. Learn is the total time to optimize the hyperparameter model.

Data Set	Kernel	Method	0-1	MRR	Fit	Run
		OVO	95.1	97.4	0.055	4.1
			****	****		
		WTA CS	91.8	95.8	0.545	9.5
	Linear		****	****		
		WTA SSVM	91.2	95.4	0.316	35.1
			****	****		
Wine		AUC FWCD	96.7	98.3	0.168	5.4
		OVO	70.4	83.6	0.0	6.0
N = 178			****	****		
F = 13		WTA SSVM	72.9	84.9	0.199	138.9
K = 3	RBF		****	****		
		AUC FWCD	96.9	98.4	0.167	13.8
		OVO	96.7	98.4	0.0	3.0
			(****)	(****)		
		WTA CS	96.5	98.2	0.019	4.4
	Linear		(****)	(****)		
		WTA SSVM	92.8	96.4	0.028	31.0
			(****)	(****)		
Iris		AUC FWCD	82.	90.9	0.166	5.8
		OVO	95.9	98.	0.0	4.8
N = 150			()	()		
F = 4		WTA SSVM	94.7	97.3	0.051	68.0
K = 3	RBF					
		AUC FWCD	95.3	97.7	0.164	10.4
		OVO	99.2	99.6	0.0	2.9
			(****)	(****)		
		WTA CS	98.2	99.1	0.31	9.0
	Linear		(****)	(****)		
		WTA SSVM	97.	98.4	0.114	27.1
			(****)	(****)		
Glass		AUC Coor	85.5	91.6	0.179	7.8
		OVO	99.	99.4	0.0	29.5
N = 214			(****)	(****)		
F = 10		WTA SSVM	99.	99.5	0.848	46.0
K = 6	RBF		(****)	(****)		
		AUC FWCD	90.7	94.7	0.18	27.2

	Linear	OVO	96.1	97.9	14.339	175.9
			(****)	(****)		
		WTA CS	94.3	96.9	1.882	45.8
			(****)	(****)		
Segment		AUC FWCD	92.3	95.9	0.585	10.1
N = 2310	RBF	OVO	94.8	97.1	0.089	58.3
F = 19			***	**		
K = 7		AUC FWCD	96.3	97.9	1.866	38.7
	Linear	OVO	26.3	49.6	7.125	57.7
			(****)	(****)		
		WTA CS	18.6	39.6	1.605	195.5
			****	****		
Abalone		AUC FWCD	23.7	44.8	2.028	22.5
N = 4167	RBF	OVO	27.	50.1	0.342	24.5
F = 10			()	(**)		
K = 21		AUC FWCD	26.8	49.5	2.225	203.9
	Linear	OVO	96.7	98.2	173.5	969.7
			(****)	(****)		
		WTA CS	42.5	68.3	0.318	112.4
			****	****		
Pg-Blox		AUC FWCD	89.9	93.	2.475	34.2
N = 5473	RBF	OVO	91.8	95.3	1.192	14.5
F = 10			****	****		
K = 5		AUC FWCD	96.8	98.2	4.768	65.3

Table 3.5: Comparison of Linear Function and Radial Basis Function Performance on Multiclass Classification with SVM optimized with a variety of Methodologies

3.4 Racial Bias

Structural loss can do more than improve the rank metrics – it can reduce racial bias in these key applications. Despite omitting racial features from society, machine learning algorithms with enough data will always be able to segment racial groups as long as there exists racial segmentation within society. Rather than avoid the bias issue, it can be confronted directly with well designed losses.

The recidivism data set comes from the Propublica report on the COMPAS algorithm (Larson et al. (2016)). It spurred a discussion on racial bias in machine learning. An anecdotal comparison was given, where a black woman and a white man both committed petty larceny and the sentencing was much more severe for the black woman. The white man then went on to commit a more serious crime. An in depth analysis of AUC of the COMPAS algorithm and the racial bias in false positives (Dressel and Farid (2018)), but this can be improved by developing a novel metric.

Here, a novel metric is presented which captures the essence of the anecdotal analysis. A measure of racism is the number of pairwise misorderings of black non-reoffenders and non-black reoffenders. A normalized version is presented in table 3.6 as 'racism'. A Frank-Wolfe coordinate descent (FWCD) with altered loss to penalize racist misorderings is presented in table 3.6 as FWCD'. Surprisingly, it improves the training AUC compared to standard pairwise hinge optimizing.

Data Set	Split	Method	AUC	Racism
Violent	Train	FWCD	67.	55.
		FWCD *race*	**71.1**	**40.6**
		Log Reg	69.8	51.7
	Test	FWCD	65.3	57.5
		FWCD *race*	64.4	**49.8**
		Log Reg	**66.4**	56.7
Non-Violent	Train	FWCD	72.7	46.
		FWCD *race*	74.1	**39.7**
		Log Reg	**74.3**	51.7
	Test	FWCD	72.7	45.2
		FWCD *race*	70.4	**43.7**
		Log Reg	**72.8**	46.1

Table 3.6: Reducing racism with structural loss

CHAPTER 4

NOVEL AUC OPTIMIZATION ALGORITHMS

This chapter introduces relationships between the convex parameter domains of the 0-1 and balanced hinge and logistic loss Lagrangian dual coefficients λ, the convex parameter domains of the pairwise hinge and logistic Lagrangian dual coefficients ρ, and the Wolfe dual coefficients for pairwise hinge α. Their domains are shown to be equivalent and the univariate losses are shown to be an upper bound on each iteration of the Frank-Wolfe algorithm given sample weights equal to the pairwise subdifferential. This enables a trust region optimization of the pairwise hinge (algorithms 4 and 5). The algorithm is presented as three cumulative versions. The versions have different properties which are discussed after their presentation. The final version was used in the experiments. A non-convex algorithm which finds the optimal bias for 0-1 or balanced accuracy is also presented (algorithm 6).

4.1 Dual Coefficient Mappings

The following theorems establish the equivalence of the proposed loss and the pairwise loss. The proposed loss dual coefficient domain is a superset of the pairwise dual domain. The proposed algorithm successively narrows the balanced loss domain toward the pairwise domain.

The Frank-Wolfe algorithm can optimize decomposable losses as independent parts or holistically. Theorems 1 and 2 establish the relationships between the Frank-Wolfe optimizations and the Lagrangian method for balanced 0-1 and pairwise SVM. These theorems allow the results of theorems 3 and 4 to apply to the Frank-Wolfe optimization of pairwise loss.

Theorems 3 and 4 show that SVM dual loss is equal to the pairwise SVM dual loss and that the pairwise dual coefficient domain is a subset of the balanced dual coefficient domain. These theorems imply that if the balanced loss solution is in the pairwise domain then it is the optimal solution and that pairwise regret bounds can be established with respect to the balanced solution.

Combining all four theorems enable us to successively tighten our boundaries until the convex combination of univariate solutions is nearly within the convex domain or moving toward the pairwise domain would not improve the underlining margin. The proposed strategy enables optimization of the dual loss without dual variables. This enables the integration of probabilistic output algorithms, decision trees and random forest, can be integrated into the Frank-Wolfe optimization.

Theorem 1 (Bijective Map between Independent Example Frank-Wolfe and Lagrangian Method Dual Coefficient Domains for Binary SVM)**.** *Let λ represent Lagrangian coefficient vector of the univariate losses (equation 2.21) and α represent the Frank-Wolfe coefficient domain of the pairwise losses (equation 2.28). There exists a bijective function between α and λ.*

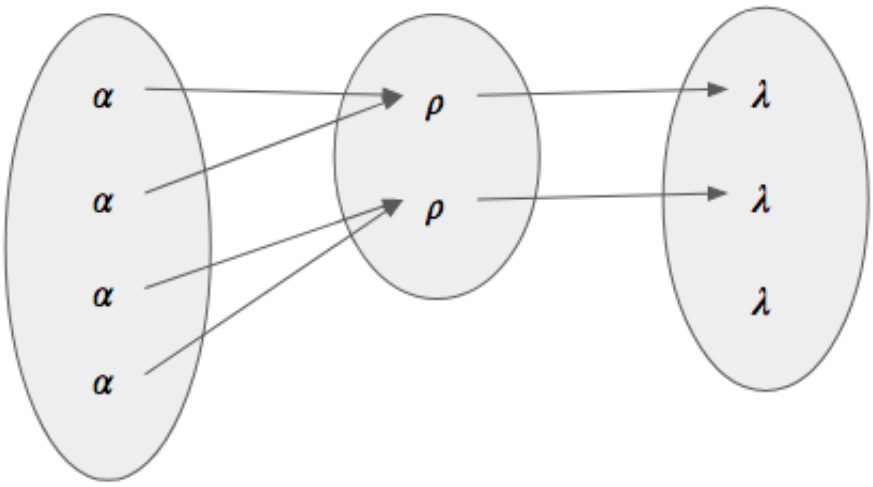

FIGURE 4.1: Coefficient Domain Mappings: Holistic Pairwise Frank-Wolfe onto Pairwise Lagrangian into Balanced 0-1 Lagrangian

Proof. In binary classification SVM, the gradient per example is binary, $\frac{\partial \mathcal{L}}{\partial y} \in \{0, 1\}$. Thus, there are only two Wolfe dual coefficients per example. Recall the coefficient constraint for the Frank-Wolfe algorithm (equation 2.28). Applied to an individual example in binary classification the constraints can be defined as blow.

$$\alpha_{i0} + \alpha_{i1} = C \quad \forall\ \alpha_i \tag{4.1}$$

Recall the box constraints for the Lagrangian method applied to the SVM optimization $\lambda_i \in [0, C]$ (eq. 2.19). The bijective function can be proved for an individual example and will hold for the collection of all examples because the examples are independent. For every example i the function f can be defined as below.

$$\begin{aligned} f(\alpha_i) &= \alpha_{i1} = \lambda_i \\ f^{-1}(\lambda_i) &= (\lambda_i = \alpha_{i1}, C - \lambda_i = \alpha_{i0}) \end{aligned} \tag{4.2}$$

Every value of λ_i is uniquely mapped to a valid α and all valid values of α are mapped onto, thus the function is bijective. □

Corollary 1.1 (Bijective Map between Independent Example Frank-Wolfe and Lagrangian Method Dual Coefficient Domains for Pairwise SVM)**.** *Let ρ represent Lagrangian coefficient*

vector of the univariate losses (equation 3.1) and α represent the Frank-Wolfe coefficient domain of the pairwise losses (equation 2.28). There exists a bijective function between α and ρ.

Proof. The pairwise SVM can be optimized as a preprocess transformation of variables changing it into a binary SVM problem. Where ρ represents the transformed feature set Lagrangian coefficients and α represents the Frank-Wolfe dual coefficients.

$$\begin{aligned} f(\alpha_i j) &= \alpha_{ij1} = \rho_{ij} \\ f^{-1}(\rho_{ij}) &= (\rho_{ij} = \alpha_{ij1}, C - \rho_{ij} = \alpha_{ij0}) \end{aligned} \tag{4.3}$$

□

Theorem 2 (Surjective Map from Holistic Frank-Wolfe onto Lagrangian Method Dual Coefficient Domains for Binary SVM)**.** *Let λ represent Lagrangian coefficient vector of the univariate losses (equation 2.21) and α represent the Frank-Wolfe coefficient domain of the pairwise losses (equation 2.28). There exists a surjective function from α onto λ.*

Proof. Given the following constraints.

$$\begin{aligned} &\lambda \in [0, C]^N \quad s.t. \lambda_k \geqslant 0 \\ &\alpha \in [0, C]^M \quad s.t. \sum_k \alpha_k = C \ \alpha_k \geqslant 0 \end{aligned} \tag{4.4}$$

Recall the matrix of subdifferentials $\mathcal{S}$. It will be shown that the below function $f : \alpha \to \lambda$ is surjective.

$$\lambda = \mathcal{S}\alpha \tag{4.5}$$

Below is an algorithm which maps any feasible λ to a feasible α. Thus the target domain λ is fully mapped, thus if it can also be shown that no one α maps to two λ then the map $f : \alpha \to \lambda$ is surjective. The proof that each α only maps to a single λ is a product of the linear transformation definition $\lambda = \mathcal{S}\alpha$ which does not map to multiple values for each element of λ. The map can be shown to be non-injective by contrapositive. Two distinct

Algorithm 3 $\lambda \rightarrow$ (holistic) α

order all lambda
repeat
binary vector = ($\lambda > 0$)
$z = argwhere_z(Col_z(\mathcal{S}) =$ binary vector$)$
λ_x = smallest non-zero element in λ
$\alpha_z = \lambda_x$
$\lambda_k = \lambda_k - \lambda_x \; \forall \; k \; s.t. \; \lambda_k > 0$
until all $\lambda_k = 0 \; \forall k$ done
if $\sum_m \alpha_m < C$ allocate the rest to α_0

α can be shown to produce the same λ. Examine the case of two data points. The holistic subdifferential matrix can be written as below.

$$\mathcal{S} = |0 \; 0; 1 \; 0; 0 \; 1; 1 \; 1]$$

The following two distinct αs produce the same λ.

$$\alpha = [1 \; 0 \; 0 \; 1]^T$$

$$\widetilde{\alpha} = [0 \; 1 \; 1 \; 0]^T$$

$$\mathcal{S}\widetilde{\alpha} = \mathcal{S}\alpha$$

□

Corollary 2.1 (Surjective Map from Holistic Frank-Wolfe onto Lagrangian Method Dual Coefficient Domains for Pairwise SVM)**.** *Let ρ represent Lagrangian coefficient vector of the univariate losses (equation 2.21) and α represent the holistic Frank-Wolfe coefficient domain of the pairwise losses (equation 2.28). There exists a surjective function from α onto ρ.*

Proof. The pairwise SVM can be optimized as a preprocess transformation of variables changing it into a binary SVM problem. Where ρ represents the transformed feature set Lagrangian coefficients and α represents the Frank-Wolfe dual coefficients. □

Theorem 3 (Injective Mapping of Pairwise SVM into Balanced SVM Dual Lagrangian Coefficients)**.** *There exists an injective mapping of the pairwise SVM Lagrangian coefficients, ρ, with regularization coefficient $\frac{C}{N^2}$ into the balanced binary SVM Lagrangian coefficients, λ with regularization coefficient $\frac{C}{N}$.*

Proof. Start by deriving the $\{0,1\}$ SVM constraints associated with $\{0,1\}$ pairwise hinge loss. Let ρ represent the Lagrangian coefficients from the pairwise loss model. Given $y_i = 1$ and $y_j = -1$, we generate pairwise labels and features $y_k = 1$, $x_k = x_i - x_j$. This utilizes some arbitrary indexing structure such as $k = i(n^-) + j$. In pairwise hinge, the bounding box constraint is the only non-trivial constraint because the linear constraint resulting from the intercept cancels out with equal positive and negative pairwise examples. The dual representation of the linear function can be decomposed. Allow two forms of indexing with $\rho_p = \rho_{ij}$.

$$\begin{aligned} w &= \sum_{p}^{n^+n^-} \rho_p t_p x_p = \sum_{p}^{n^+n^-} \rho_p (x_i - x_j) = \sum_{i}^{n^+}\sum_{j}^{n^-} \rho_{ij}(t_i x_i + t_j x_j) \\ &= \sum_{i}^{n^+}(\sum_{j}^{n^-} \rho_{ij}) t_i x_i + \sum_{j}^{n^-}(\sum_{i}^{n^+} \rho_{ij}) t_j x_j = \sum_{i}^{n^+} \lambda_i t_i x_i + \sum_{j}^{n^-} \lambda_j t_j x_j \end{aligned} \tag{4.6}$$

This yields the function which maps ρ to λ and aligns with the standard dual representation. The resulting bounding boxes with respect to Lagrangian coefficients for individual data points α can be calculated.

$$0 \leqslant \lambda_i = \sum_{j}^{n^-} \rho_{ij} \leqslant \frac{C}{N^2} n^- = \frac{C}{N} q$$

$$0 \leqslant \lambda_j = \sum_{i}^{n^+} \rho_{ij} \leqslant \frac{C}{N^2} n^+ = \frac{C}{N} p$$

Combining the box constraint for the pairwise parameters with the previous function shows that the pairwise coefficient domain falls within the balanced 0-1 loss coefficient

domain. The set of constraints for the balanced hinge loss are well established and listed below.

$$0 \leqslant \lambda_i \leqslant \frac{C}{N} q$$

$$0 \leqslant \lambda_j \leqslant \frac{C}{N} p$$

These parameter bounds equal the balanced binary loss bounds, which shows that the map does not leave the binary domain.

This mapping can be shown to be injective by contradiction. Assume that this map is non-injective. This implies that changing ρ to can yield the same λ. In order to fix an entry λ_i, two ρ entries in column i must offset each others changes. Thus, two λ_j are changed. Therefore, the function is injective.

This function is not bijective because it is not surjective. Examine the λ with all zeros except for two elements. If those elements exceed $\frac{C}{N}$ then they constitute a valid binary SVM coefficient but an invalid pairwise coefficient. □

Corollary 3.1 (Bias Constraint)**.** *The linear constraint for Binary SVM decreases the size of the unmapped λ set while preserving the injective function.*

Proof. The mapping of rho variables to lambda variables by column and row sums of the rho matrix implies that the sum of positive lambdas must equal the sum of negative lambdas.

$$\sum_{i}^{n^+} \sum_{j}^{n^-} \rho_{ij} = \sum_{i}^{n^+} \lambda_i = \sum_{j}^{n^-} \lambda_j$$

Subtracting the sum of the negative coefficients from the sum of positive coefficients shows that the pairwise loss maintains the linear constraint in binary losses.

$$\sum_{i}^{n^+} \lambda_i - \sum_{j}^{n^-} \lambda_j = \sum_{i}^{n^+} t_i \lambda_i + \sum_{j}^{n^-} t_j \lambda_j = \sum_{k}^{N} \lambda_k t_k = 0$$

Incorporating the linear constraint decreases the balanced domain of λ but the pairwise domain only maps to a subset of the balanced λ domain with a linear constraint. □

Theorem 4 (Equal Dual Losses)**.** *The Balanced 0-1 and Pairwise Dual Losses are Equal over the entire Pairwise Coefficient Domain given Regularization weights $\frac{C}{N}$ and $\frac{C}{N^2}$ respectively.*

$$\mathcal{L}_{dual\ balanced\ 0-1} = \mathcal{L}_{dual\ pairwise} = \frac{1}{2}\lambda Q\lambda - \mathbb{1}^T\lambda \quad s.t.\ \lambda \in \mathcal{D}_{\rho\to\lambda} \tag{4.7}$$

Proof. The definition of the margin as the sum of subdifferentials (eq. 3.13) can be combined with the map of Frank-Wolfe coefficients to the univariate domain (Theorem 2).

$$\Delta^T\alpha = (\mathbb{1}_n^T\mathcal{S})\alpha = \mathbb{1}_n^T\lambda \tag{4.8}$$

The Frank-Wolfe coefficients can be transformed for the quadratic component also. Substituting the above linear component into the structural SVM dual loss shows that the two dual losses are equal as claimed above. However, theorem 3 reveals that the pairwise domain is a subdomain of the balanced loss. For all solutions in the pairwise domain, the two losses are equal.

$$\frac{1}{2}(\mathcal{S}\alpha)^T Q\mathcal{S}\alpha - \Delta^T\alpha = \frac{1}{2}\lambda Q\lambda - \mathbb{1}^T\lambda \tag{4.9}$$

The pairwise dual coefficient domain maps to a restriction of the balanced 0-1 dual coefficient domain (Theorem 3). Specifically, outside the restricted domain.

$$\Delta^T\alpha \neq \mathbb{1}^T\lambda \tag{4.10}$$

□

This says that a projection back to the valid domain which accounts for the loss contour would be the optimal loss.

4.2 Algorithms

This section introduces a novel slack which is an upperbound of the pairwise structural slack. The proposed loss is then defined with the novel slack variable. The proposed algorithms optimize the proposed loss.

4.2.1 Upper Bound of Structural Pairwise Hinge

The sum of univariate losses are an upper bound for the respective pairwise loss. This is a property of all convex functions.

$$\xi(y_i) + \xi(-y_j) \geqslant \xi(y_i - y_j) \tag{4.11}$$

The structural SVM slack at each iteration of the Frank-Wolfe algorithm is upper bounded by a subdifferential weighted univariate hinge. The weight c_j is the subdifferential for example j of the pairwise hinge. The structural loss at iteration k is defined below.

$$\xi_m = |\sum_{k=1}^{N} \mathcal{S}_{m,k} - \langle w, \sum_{k=1}^{N} \mathcal{S}_{m,k}\ t_k x_k \rangle|_+ \tag{4.12}$$

Because the margin Δ and the subdifferential scale equally per each example, the weight can be shifted out of the loss and more closely resemble the 0-1 hinge loss.

$$\xi_m = |\sum_{k=1}^{N} \mathcal{S}_{m,k}(1 - t_k y_k)|_+ \tag{4.13}$$

The structural slack can be upper bounded by a subdifferential weighted sum of univariate losses (equation 4.14).

$$\xi_m \leqslant \sum_{k=1}^{N} \mathcal{S}_{m,k}|1 - t_k y_k|_+ = \tilde{\xi}_m \tag{4.14}$$

When the prediction of the univariate is initialized at zero and the current Frank-Wolfe prediction is maintained, the gradients are equal. The two have equivalent Hessians. These

observations combined with the convex dual coefficient domain equivalence presented in the next section enable the subgradient weighted univariate to serve as a trust region algorithm.

Three cumulative variations of a trust region optimization of the Frank-Wolfe optimization of pairwise hinge are presented. The first two are guaranteed to converge to the optimal solution. The third does not have the same guarantee, however, empirically it converges to a nearly identical AUC. The third is what enables the gradient boosting approach to be applied. The resulting optimization has the auxiliary benefit that it also centers proposed predictions. This produces high accuracy or balanced accuracy in conjunction with an rank metric optimized list.

4.2.2 Trust Region Frank-Wolfe Ranking Algorithm

The proposed primal loss is defined below.

$$\widetilde{\mathcal{L}}_p = \frac{1}{2}||w||^2 + \widetilde{\xi} \tag{4.15}$$

The proposed slack is an upper bound of the pairwise structural slack and the two solutions point to the same constraint when the proposed loss is initialized at the origin. Optimizing the proposed slack from the origin produces the same first order Taylor optimization.

$$argmin_{\lambda_s}\langle\nabla\widetilde{\mathcal{L}}_d(0), \lambda_s\rangle = \mathcal{S}(argmin_s\langle\nabla\mathcal{L}_d(\alpha^k), s\rangle) \tag{4.16}$$

The proposed loss and the pairwise loss have the same Hessian matrix. But, the proposed loss can integrate the Hessian at each step.

$$\nabla^2\widetilde{\mathcal{L}}_d(\lambda) = \nabla^2\mathcal{L}_d(\alpha) \quad \forall\ \lambda\ \&\ \alpha \tag{4.17}$$

The proposed loss is an appropriate choice for a Trust Region given that the first order and second order Taylor series of the two losses align.

Algorithm 4 Trust Region Frank-Wolfe for Pairwise Losses: Version 1

$s = argmin_s \langle \nabla \mathcal{L}(0), s \rangle$
$\omega = s^T \mathcal{S}$
$\lambda^{(1)} = argmin_\lambda \frac{1}{2} \lambda^T Q \lambda - 1^T \lambda$ s.t. $\lambda_i \in [0, C\omega_i] \forall i$
$k = 1$
repeat
  $\gamma = \frac{2}{k+2}$
  $s = argmin_s \langle \nabla \mathcal{L}(\lambda^{(k)}), s \rangle$
  $\omega = s^T \mathcal{S}$
  $\lambda' = argmin_\lambda \frac{1}{2} \lambda^T Q \lambda - 1^T \lambda$ s.t. $\lambda_i \in [0, C\omega_i] \forall i$
  $\lambda^{(k+1)} = (1 - \gamma)\lambda^{(k)} + \gamma \lambda'$
  $k = k + 1$
until $\nabla \mathcal{L}_d(\lambda^{(k)})^T (\lambda^{(k)} - \omega)$ decreases by less than ϵ

The first variant tracks the Frank-Wolfe solution at each iteration and then solves a univariate upper bound of the current solution. That current solution then informs discovery of the next constraint. The next constraint combines with the current weights for the next solution.

In version 1, the current sample weight vector is no longer combined updated with the next unconvered Frank-Wolfe constraint. Instead, a given iteration's constraints are used as the weights for the local model and the dual outputs are combined with the current Frank-Wolfe solution in order to find the next iteration of Frank-Wolfe solution.

This algorithm is applicable to any primal loss with a dual form: hinge loss, logistic loss, and exponential loss. The sample weighted SVM optimization is presented here (on lines 3 & 8). The algorithm successively tightens the balanced univariate boundary so that it must be closer to the pairwise domain at every iteration. The main difference is that if moving toward the pairwise domain increases the margin (a tight upper bound on AUC) then the proposed algorithm stays outside the pairwise domain.

Although the logistic regression has a dual form which can track Lagrangian coefficients, storing functions or combining linear weights are feasible in variation three. The sample weighted logistic regression has a proportional gradient and the same Hessian and is

Algorithm 5 Trust Region Frank-Wolfe for Pairwise Losses: Version 2

Input: features, X; labels, t; and classification algorithm, A
$y = 0$
$\omega = argmax_{\omega} \langle \nabla \xi(y), \omega \rangle$
optimize A given X,t, and sample weights ω
$z = A(X)$, output label probability
$y^{(1)} = logit(z)$, transform probability into decision function
$k = 2$
repeat
  $\gamma = \frac{2}{k+2}$
  $\omega = argmax_{\omega} \langle \nabla \xi(y), \omega \rangle$
  optimize A given X, t, and sample weights ω
  $z = A(X)$
  $y' = logit(z)$
  $y^{(k+1)} = (1 - \gamma) y^{(k)} + \gamma y'$
  $k = k + 1$
until $\xi(y^{(k)})$ decreases by less than ϵ

therefore a worthwhile candidate for an approximation to the true loss. The solution is guaranteed to be within the domain of feasible losses and therefore provides an appropriate optimization for a trust algorithm.

Non-linear functions. This work proposes using the logit function to produce a decision function from a normalized probability. The decision function can then be incorporated into each iteration of the proposed trust region algorithm.

This version no longer requires tracking the dual coefficients. But, it produces the same result. The constraint, s, can be calculated from the decision function score, y, alone. The logit transformation is used to convert decision tree probabilities into decision functions scores.

4.2.3 Non-Convex Bias

No matter what the data configuration looks like the optimal bias for the non-convex 0-1 classification or balanced classification losses can be directly optimized. We present the algorithm for 0-1 loss, and we use the balanced loss in our applications because it improves

F-1 score and other multivariate output metrics more so than 0-1 loss.

Algorithm 6 Non-convex bias optimization

Input: prediction scores y, labels t
v = argsort(t), descending
u = cumulative sum($t[v]$)
i = $argmax_i(u[i])$
bias = $\frac{n^-}{n} y[v[i]] + \frac{n^+}{n} y[v[i+1]]$

4.2.4 Theoretical Results

Feasible constraints. Although the coefficient domains are equivalent, the margins are different. The feasible constraints are those which can be achieved by finding the possible maximum margin for any given set of parameters in the feasible domain and the linear combinations thereof. At this point, it is not possible to make definitive claims about the domains of feasible constraints. However, the number of constraints in the ranking problem is much higher and therefore there is a greater probability that a greater number of distinct constraints will be active than in the classification scenario. This discrepancy in active search space is resolved by applying weights according to the pairwise subdifferentials is resolved by allowing sample weights of the univariate to reflect the active constraint proposed by each iteration of Frank-Wolfe or structural SVM.

Implicit constraint. There is one constraint that is not articulated in the literature but is independent of any specific features and labels. $\sum_i \lambda_i > 0$ which also means the Wolfe coefficient associated with the zero vector of joint features cannot be equal to one. This is intuitively the case because in both classification and ranking if $\sum_i \lambda_i = 0$ the prediction will be zero features. This will necessitate active margins $\Rightarrow \sum_i \lambda_i > 0$.

Kernel methods. Calculating with respect to kernel is useful when the number of features exceeds the number of examples. It is often used with a radial basis function which produces an infinite dimensional feature in the standard space. A kernel matrix K has elements $K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle$. A radial basis kernel operator produces elements

$K_{ij} = exp\{-\frac{1}{\sigma^2}||x_i - x_j||^2\}$. The structural kernel elements are calculated in terms of $\langle \partial\Psi(\phi(x), y, t), \partial\Psi(\phi(x), y', t)\rangle$. If it were not done in this manner, then the equivalent alternate Wolfe constraint/example decompositions would not produce equivalent results. Furthermore, this enables the structural loss hinge loss which derives the most violated constraint $\partial\Psi$, ($\partial\Psi(\phi(x), y, t) = \Psi(\phi(x), y) - \Psi(\phi(x), t)$). Last, if it were not done in this manner it would not align with the pairwise construction of kernel functions of features $\phi(x_i) - \phi(x_j)$. The pairwise construction of kernel functions of features must be done in that way otherwise the transitive properties of distances will no longer hold.

Given that the structural kernel is a linear combination of the univariate kernel, we can efficient represent the kernel in terms of λ. Despite its simplicity, this has not been done before (however more complex solutions with less gains in efficiency can be found in the literature). This means that the kernel matrix need not be ever extended to the structural constraints when using the block coordinate Frank-Wolfe optimization. Instead, we can use the structural constraints to project the large α to the smaller λ. This minimizes storage demands and enables efficient prediction $\langle w, \phi(x)\rangle$, which is useful but also necessary at each iteration for line search optimization and most violated constraint calculations.

Density of Lagrangian coefficients. In the block coordinate Frank-Wolfe and the structural SVM (also known as a fully corrective Frank-Wolfe) the entire kernel matrix must be calculated. The very first constraint is dense with respect to λ and requires that all components of a kernel matrix be calculated. The block coordinate Frank-Wolfe needs away steps in order to become sparse. The algorithm proposed in the next section does not suffer from this weakness.

Convergence. The proposed algorithm minimizes the same dual loss or a similar surrogate dual loss. The proposed algorithm searches a larger dual coefficient domain but can do so while only implicitly deriving dual coefficients. At each iteration of the proposed algorithm the solution is pulled towards the pairwise domain, thus converging to a pairwise solution. The difference however is that if a solution outside of the domain is

unable to be improved upon by moving towards the pairwise domain, then the expanded domain is used.

CHAPTER 5

RANKING EMPIRICAL RESULTS

This chapter reviews the ranking experimental results. It covers the experimental designs, statistical evaluations, and sourced implementations. It reviews the bipartite ranking performance of linear functions, kernel functions, and non-linear ensemble methods. Then it covers collaborative filtering applications.

5.1 Experimental Design

The experiments are focused the following problem types on bipartite ranking and collaborative filtering. The bipartite ranking tasks are split into three groups: linear methods, kernel methods, and non-linear methods. For each experiment, 30 trials are run. Each trial uses a distinct random split of data points with 67% for training and 33% for testing. The collaborative filtering experiments are split into list based targets and binary targets. All of the datasets come from the LETOR experiments and have 5 predefined folds.

Evaluation. We inspected the following metrics: 0-1 accuracy, balanced accuracy, AUC, F-1 score, average precision, discounted cumulative gain, and the number of positives ranked above the highest scoring negative example. Training times are also examined. The fit time of a single round of learning is reported alongside the total learning time which

includes the hyperparameter optimization.

Statistical significance. Significance is determined by the paired difference test. For every one of the 30 trials a pairwise difference in each target metric is calculated. The pairwise difference is either accepted or rejected as significantly different from zero. The pairwise difference is more informative than a difference in means because the data set causing underlying variance in the two models is accounted for, which is appropriate given it is shared by both models. The trust region method is notated as a statistically significant improvement at the various confidence intervals: * for 90%, ** for 95%, *** for 99%, and **** for 99.9%. Statistically significant lower performance is notated with parentheses (*), (**), (***), (****).

Implementations. RankBoost was implemented by the Lemur project. (LEMUR (2019)) Structural SVM is a cutting plane solver. (Tsochantaridis et al. (2005); Joachims (2005)) The logistic regression solver uses a Liblinear algorithm. (Fan et al. (2008)) LibSVM is the underlying SVM algorithm used in the trust region optimization. (Chang and Lin (2011)) The Random Forest implementation is from SciKit-Learn and the Gradient Boosting algorithm used is XGBoost. (Chen and Guestrin (2016)) The block coordinate Frank-Wolfe was implmented in python.

Data sets. The data used are publicly available (from the UCI data repository, Dheeru and Karra Taniskidou (2017)) and often used in the bipartite ranking and collaborative filtering research papers.

Non-linear methods. Although the kernel methods outperform the linear methods, the non-linear methods perform the best on these data sets. The trust region approach optimizes a Random Forest at every iteration. The default depth is 6 and number of trees is 20. Typically, the trust algorithm converges within 5 to 15 iterations. So, we included an ensemble of ensemble methods as baseline. A bag of 15 Random Forests and a bag of 15 XGBoost classifiers.

Collaborative filtering. In the following collaborative filtering experiments the the

linear form of the novel ranking algorithm is compared to the structural SVM linear model and the non-linear form of the novel ranking algorithm is compared to Rank Boost.

Hyperparameter Optimization Specifics

The hyperparameter optimization requires a transformation – the Gaussian process regression optimizes over x but the hyperparameter is the transformation of x and bounds on the values for x for each hyperparameter. The linear functions optimize a regularization value C.

Parameter	Transformation	Bounds
C	10^x	[-5, 5]

In additional to the regularization term, SVM with RBF kernels have one additional model hyperparameter, σ^{-2}.

Parameter	Transformation	Bounds
C	10^x	[-5, 5]
σ^{-2}	10^x	[-4, 2]

Although there are more hyperparameters that what were used in the following optimizations of ensemble learners (Random Forest and XGBoost), these two variables have the largest impact. Furthermore, adding variables slows down the hyperparameter optimization.

Parameter	Transformation	Bounds
Number of Ensemble Learners	Int(x)	[$\frac{1}{10}$n, $\frac{1}{4}$n]
Max Depth	Int(x)	[1, f]

Bagged ensemble methods produces superior AUC because independent classifiers increase AUC (Long (2008)). The bagged ensemble methods then have one additional component, the number of ensemble learners.

Parameter	Transformation	Bounds
Number of Ensemble Learners	Int(x)	[$\frac{1}{10}$n, $\frac{1}{4}$n]
Max Depth	Int(x)	[1, f]
Number of Estimators per Ensemble	Int(x)	[1, 20]

5.2 Results

The bipartite ranking and collaborative filtering experiments show that the proposed optimization works. In fact, the proposed optimization produces the best solution in seven on the eight data sets. It also produces an efficient kernel optimization and reveals that efficient strategies already exist which improve upon structural SVM and published improvements thereof.

5.2.1 Bipartite Ranking

The linear function results largely show the experimental success of balanced logistic regression which matches the theoretical claims of and experiments of (Kotlowski et al. (2011)). It could be seen as a confirmation that the methodology does not worsen performance. The proposed optimization improves RBF kernel results, but even more so, it improves the non-linear ensemble results. As noted before, this is the first Frank-Wolfe design which enables non-linear ensemble optimization. The size of the data sets is reported as part of the non-linear results table.

The linear method experiments included logistic regression, balanced logistic regression, OPAUC (one-pass AUC optimization), Frank-Wolfe coordinate descent, SSVM, and the proposed Trust algorithm. Results show that balanced logistic regression outperformed the standard logistic regression. Balanced logistic regression was the best performing baseline algorithm and the proposed Trust algorithm more or less tied it on every experiment. On 1 of 8 data sets, the FWCD algorithm performed significantly better than the balanced logistic regression. The regularization coefficient, C, is tuned with hyperparameter optimization.

The kernel method experiments compared SVM, Frank-Wolfe coordinate descent, and the proposed Trust algorithm. On 5 of 8 data sets, the proposed Trust algorithm outperformed FWCD and in 6 of 8 experiments the the Trust algorithm outperformed SVM. The Trust method is a hybrid of the two yet outperforms the others. The regularization coefficient,

C, and the inverse covariance for the radial basis function were tuned with hyperparameter optimization.

The non-linear ensemble method experiments compared Random Forest, XGBoost, RankBoost, and the proposed Trust region algorithm. The detailed results for each of the 8 data sets are presented in the following slides. The hyperparameter optimization tunes the depth of the trees and the number of trees. There is theoretical evidence that bagging independent classifiers will improve AUC (Long). Bagging random forests merely produces a larger forest but bagging XGBoost and the Trust algorithm were notated E-XGB and E-Trust for and ensemble of XGBoost classifiers and ensemble of Trust region rankers. For ensemble of ensembles methods, the number of ensembled learners is also a hyperparameter. For an idea of scale, the RF and XGBoost would have around 100 trees where the Trust algorithm has closer to 150 trees. The E-XGBoost and E-Trust have a 20x increase in the number of trees.

Linear functions

The proposed linear optimization shows minor differences from balanced logistic regression – at most 0.003. This is because the chosen optimization is a logistic regression for each step of the proposed Frank-Wolfe algorithm. The first step of the proposed algorithm uses sample weights that equal the balanced class weights. The balanced logistic and the proposed optimization both outperform the 0-1 logistic regression, although sometimes barely sometimes by more than 2 points. The structural SVM (SSVM) and Frank-Wolfe coordinate descent (FWCD) typical perform worse. This is not a complete surprise given that it is known that logistic regression and balanced logistic regression are AUC consistent.

Data	Step	AUC	AP	F1	0-1	p-q	Top	Fit	Run
Ion	Log 01	89.	88.6	76.8	86.3	81.6	24.733	0.0	3.0
		()	(**)		(****)				
	Log Bal	88.9	88.3	77.4	85.7	82.2	25.1	0.0	3.2
		()	()	()	(***)		()		
	FWCD	90.1	56.4	34.	61.	57.2	25.433	0.004	2.4
		(****)	****	****	****	****	()		
	OPAUC	84.	83.	42.1	74.2	63.7	20.033	0.01	4.2
		****	****	****	****	****	****		
	SSVM	89.7	89.7	37.6	72.9	61.8	27.033	0.012	25.0
		(**)	(****)	****	****	****	(****)		
	Trust	88.6	88.1	77.1	84.3	82.2	24.8	0.006	3.3
Boston	Log 01	78.8	6.1	0.5	93.3	49.9	0.067	0.0	4.0
		****	****	****	(****)	****	()		
	Log Bal	82.	18.5	28.6	76.4	77.1	0.0	0.0	3.7
				()	(****)				
	FWCD	80.9	10.4	6.6	47.6	50.	0.033	0.05	4.2
		****	***	***	****	****	(****)		
	OPAUC	52.1	0.0	0.0	0.0	0.0	0.1	0.01	22.723
		****	****	****	****	****	()		
	SSVM	76.3	15.5	10.8	51.4	55.8	0.1	1.628	140.1
		**	**	****	****	****	()		
	Trust	81.9	18.7	27.9	73.7	78.1	0.033	0.011	3.4

Credit	Log 01	92.1	90.9	84.3	85.7	85.9	21.433	0.0	3.2
		(***)	(****)	*	**	*	(*)		
	Log Bal	92.1	90.7	84.8	85.9	86.2	20.467	0.0	3.1
		(**)	(**)	()		()	()		
	OPAUC	50.	0.0	0.0	0.0	0.0	0.9	0.03	37.15
		****	****	****	****	****	****		
	FWCD	77.1	65.7	34.3	51.	50.9	13.533	0.039	3.2
		****	****	****	****	****	****		
	SSVM	79.6	77.8	59.8	70.4	68.5	11.867	1.128	92.3
		****	****	****	****	****	****		
	Trust	91.9	90.5	84.7	86.1	86.2	19.867	0.021	3.3
Pima	Log 01	82.7	70.4	63.	77.2	72.4	3.2	0.0	3.2
			(**)	****	(****)	****	(**)		
	Log Bal	82.6	70.1	66.6	75.1	74.5	2.867	0.0	3.4
				()	(****)	(*)	()		
	OPAUC	59.1	26.7	26.9	18.8	27.2	0.9	0.02	12.4
		****	****	****	****	****	***		
	FWCD	81.1	34.2	25.5	50.	50.	4.733	0.059	3.3
		****	****	****	****	****	(***)		
	SSVM	78.8	66.	51.	34.3	50.	3.867	1.436	75.3
		****	****	****	****	****	(*)		
	Trust	82.7	70.2	66.4	73.9	74.3	2.833	0.018	3.9
Student	Log 01	76.7	61.7	55.3	74.6	67.9	4.167	0.0	3.8
		***	()	****	(****)	****	(*)		
	Log Bal	76.6	61.4	61.5	72.7	71.4	3.6	0.0	3.9
		****		()	(****)	()	()		
	OPAUC	55.4	25.7	29.	35.7	36.	1.233	0.03	6.1
		****	****	****	****	****	***		
	FWCD	70.	27.	24.1	49.5	50.	5.467	0.119	3.8
		****	****	****	****	****	(***)		
	SSVM	68.	51.1	50.5	65.4	63.1	1.867	1.85	80.8
		****	****	****	****	****	**		
	Trust	76.8	61.6	61.4	71.4	71.2	3.5	0.01	3.9

Yeast	Log 01	72.8	48.	29.7	67.1	55.1	1.7	0.0	3.5
			(****)	****	()	****	()		
	Log Bal	72.8	47.7	59.3	68.	69.6	1.633	0.0	2.7
		****	***	*	(****)		()		
	OPAUC	69.2	34.2	10.7	68.2	52.3	0.133	0.04	4.2
		****	****	****	(***)	****	****		
	FWCD	71.9	21.7	22.2	51.	50.	0.9	0.0	3.7
		****	****	****	****	****	***		
	SSVM	72.8	47.9	49.8	37.9	53.9	1.6	0.14	24.0
			()	****	****	****	(****)		
	Trust	72.8	47.8	59.6	66.6	69.7	1.6	0.01	2.6
Spam	Log 01	97.	94.6	90.4	92.6	91.9	29.933	0.04	3.4
		(****)	(***)		(**)	****	()		
	Log Bal	97.	94.5	90.9	92.7	92.5	26.167	0.035	3.3
		(***)		(****)	(****)	(****)			
	OPAUC	50.4	0.0	0.0	0.0	0.0	0.567	0.204	34.612
		****	****	****	****	****	****		
	FWCD	91.5	45.7	28.3	50.	50.	25.233	0.113	4.1
		****	****	****	****	****			
	SSVM	86.1	79.	64.8	59.6	65.4	23.167	1.636	104.4
		****	****	****	****	****			
	Trust	97.	94.6	90.5	92.4	92.3	29.2	0.214	4.9
Pg-Blox	Log 01	94.7	78.7	67.9	94.5	77.9	20.233	0.049	3.4
		****	(**)	()	(****)	****	(****)		
	Log Bal	96.4	78.5	71.1	92.6	91.	13.8	0.031	2.5
		**	(***)	(****)	(****)	****	(*)		
	FWCD	96.2	73.8	16.	52.5	52.9	8.533	0.161	4.4
		**	****	****	****	****	****		
	OPAUC	52.2	2.1	2.3	3.1	2.8	0.267	0.137	28.6
		****	****	****	****	****	****		
	SSVM	92.9	69.9	53.4	79.5	80.7	12.3	1.902	45.0
		****	****	****	***	****			
	Trust	96.5	77.9	66.7	90.4	91.7	12.533	0.123	3.9

Table 5.1: Bipartite Ranking Linear Testing Results

Kernel functions

Frank-Wolfe dual coordinate descent can do well with the right hyperparameters. However, it seems to overfit when using an SMBO for training.

Data	Step	AUC	AP	F1	0-1	p-q	Top	Fit	Run
	SVM	97.6	83.5	66.8	85.2	81.1	26.967	0.0	17.159
		()	**		(**)	()	()		
Ion	FWCD	97.8	51.1	27.6	52.3	51.1	28.233	0.049	28.888
		()	****	****	****	****	()		
	Trust	97.6	95.7	73.8	77.2	80.5	24.9	0.044	37.188
	SVM	70.9	24.8	22.	91.7	58.7	0.8	0.0	50.141
		**			(***)		()		
Boston	FWCD	76.9	14.8	6.1	53.	50.	1.1	0.06	70.641
		(**)	****	****	****	****	(***)		
	Trust	74.2	28.6	23.9	89.8	59.8	0.767	0.056	67.442
	SVM	74.2	59.8	57.4	68.1	66.8	3.067	0.014	42.746
			*	***	*	***	(*)		
Credit	FWCD	72.2	23.8	22.5	51.7	50.	3.067	0.084	44.684
		****	****	****	****	****	(**)		
	Trust	74.3	65.3	67.3	69.7	70.	1.8	0.316	57.919
	SVM	79.1	49.9	35.2	71.1	61.5	4.0	0.002	38.187
		***	****	****	()	****	(*)		
Pima	FWCD	76.9	33.8	27.6	48.9	50.	3.933	0.037	38.192
		****	****	****	****	****	()		
	Trust	79.9	66.8	63.6	70.7	71.3	2.833	0.055	37.16
	SVM	59.4	36.3	22.4	66.8	54.	0.733	0.1	16.666
		(*)		****	(****)		()		
Student	FWCD	54.6	17.5	23.8	49.4	50.	0.667	0.073	32.973
		**	****	***	****	****	()		
	Trust	57.2	39.4	38.9	60.7	55.7	0.5	0.914	31.367
	SVM	76.5	24.4	16.8	70.	54.9	2.367	0.023	10.994
			****	****	(**)	****	()		
Yeast	FWCD	74.	23.8	22.5	51.5	50.	1.267	0.148	7.313
		****	****	****	****	****	*		
	Trust	76.9	55.3	60.2	68.3	70.6	1.9	0.208	9.048
	SVM	95.5	91.2	87.8	90.4	89.9	9.367	542	108.973
		(***)	(***)	(***)	(****)	(***)	*		
Spam	FWCD	93.2	38.3	24.5	51.4	50.	19.433	2.389	121.21
		**	****	****	****	****	()		
	Trust	94.3	89.9	85.6	88.3	88.4	13.433	8.56	187.575

Pg-Blox	SVM	94.5	82.2	71.8	95.2	80.1	21.833	162	70.414
		****	(****)	(****)	(****)	****	(**)		
	FWCD	96.6	37.7	8.7	52.9	50.	38.7	3.103	167.886
		(****)	****	****	****	****	(****)		
	Trust	96.2	76.	67.	93.	83.	16.833	3.764	183.832

Table 5.2: Bipartite Ranking Radial Basis Function Kernel Testing Results

Non-linear functions

Bagged XGBoost is the best peforming baseline. This is not surprising given the well established success of XGB for optimizing 0-1 in conjunction with the fact that there exists a functional lower bound of AUC in terms of 0-1 loss. Bagging ensemble learners improved the AUC from 5.3 to 33 points. This is substantial given that random AUC is 50 and thus a 33 point improvement is 66% of the total possible score. The case where the AUC increases only 5.3 is when the AUC of the baseline learner is already 93.8 which is only 6.2 away from a perfect prediction. The improvement in AUC with independent model has been studies theoretically (Long (2008)), but previous work was limited to weak learners. Experimentally, there is greater success with strong learners.

The non-linear models have two novel methods to compare. The random forest trust region algorithm (Trust) and an ensemble of said method (E-Trust).

Data	Step	AUC	AP	F1	0-1	p-q	Top	Fit	Run
	RF	89.9	83.7	88.4	92.1	90.5	10.2	0.032	4.37
		****	****	()	(**)		****		
		****	****	()	(*)		****		
	RankBoost	79.8	75.	52.6	35.6	49.9	7.6	0.44	25.41
		****	****	****	****	****	****		
		****	****	****	****	****	****		
	XGB	89.6	83.4	87.5	91.7	89.7	16.2	0.018	4.19
Ion		****	****		()	***	****		
N = 351		****	****		()	***	****		
F = 34	E-XGB	95.5	94.3	86.9	90.9	89.7	26.2	0.754	14.71
p = 36%		****	****	**		****	***		
		****	****	***	**	****	****		
	Trust	97.	95.9	88.	91.2	91.	29.0	0.12	2.72
		****	****			()	**		
	E-Trust	97.2	96.3	88.1	91.6	90.8	30.0	1.12	45.06
	RF	50.6	5.1	3.3	92.3	50.6	0.167	0.036	14.6
		****	****	****	(**)	****	**		
		****	****	****	(****)	****	**		
	RankBoost	81.	25.9	12.9	6.9	50.	0.2	0.611	31.2
		****	****	****	****	****	**		
		****	****	****	****	****	*		
	XGB	54.1	8.6	11.9	92.	53.8	0.333	0.01	14.6
Boston		****	****	***	()	**			
N = 506		****	****	****	(****)	****			
F = 13	E-XGB	87.1	30.5	31.8	90.1	64.4	0.167	0.422	29.9
p = 7%		(*)		(****)	***	(****)	***		
			*	(***)		(***)	**		
	Trust	85.7	31.8	19.5	91.7	56.4	0.467	0.111	3.6
		****		**	(***)	***	()		
	E-Trust	87.4	33.	24.6	90.8	59.7	0.367	1.571	59.6

Credit N = 690 F = 53 p = 44%	RF	86.5	79.5	85.5	86.9	86.8	6.667	0.032	5.4
		****	****		**	*	****		
		****	****	*			****		
	RankBoost	92.5	91.5	62.2	45.1	49.9	26.767	0.762	27.0
		****	*	****	****	****	(***)		
		****	()	****	****	****	(****)		
	XGB	86.5	78.4	85.	86.2	86.3	5.7	0.022	5.0
		****	****	***	****	***	****		
		****	****	***	***	***	****		
	E-XGB	93.3	91.6	84.9	86.2	86.1	22.067	1.156	20.9
				***	****	****	()		
			()	***	***	***	(**)		
	Trust	93.5	91.9	85.8	87.3	87.1	21.933	0.12	2.7
		(*)	(****)		(*)	()	(***)		
	E-Trust	93.3	91.4	85.9	87.	87.	18.133	1.469	47.3
Pima N = 781 F = 8 p = 35%	RF	67.6	51.8	53.9	74.8	67.7	3.133	0.04	4.6
		****	****	****	**	****	***		
		****	****	****	**	****	***		
	RankBoost	83.2	71.5	51.4	34.6	50.	4.8	0.806	25.3
		()	(**)	****	****	****			
		(****)	(****)	****	****	****			
	XGB	70.7	53.7	61.	75.8	71.	2.1	0.009	3.2
		****	****	****	()	****	****		
		****	****	****		****	****		
	E-XGB	82.3	69.8	66.7	74.9	74.4	4.967	0.503	15.7
		****		()	***	()			
		*	()	()	***	()			
	Trust	83.2	70.5	66.4	75.8	74.4	5.3	0.13	2.8
		(****)	(***)	()		()	()		
	E-Trust	82.6	69.6	66.1	76.	74.2	5.1	1.716	50.5

Student N = 1008 F = 32 p = 32%	RF	58.	32.4	25.3	71.	56.8	1.567	0.04	4.2
		****	****	****	()	****	****		
		****	****	****	(**)	****	****		
	RankBoost	74.2	58.4	48.2	30.5	48.	3.067	1.21	28.3
				****	****	****	**		
		**	***	****	****	****	****		
	XGB	63.	42.9	43.5	73.	62.3	1.733	0.016	4.8
		****	****	****	(****)	****	****		
		****	****	****	(****)	****	****		
	E-XGB	73.1	57.7	55.3	66.9	65.2	4.067	1.22	28.0
		***	**	***	***	****			
		***	***	***	**	***	****		
	Trust	74.6	58.9	57.3	70.	68.3	4.733	0.12	2.8
			***		()	()	***		
	E-Trust	74.8	59.6	57.4	69.4	68.2	6.0	2.382	50.2
Yeast N = 1484 F = 8 p = 31%	RF	63.1	42.6	41.9	73.3	62.5	1.833	0.035	4.3
		****	****	****	(***)	****	****		
		****	****	****	****	****	***		
	RankBoost	68.6	44.4	47.8	31.4	50.	1.067	1.616	30.4
		****	****	****	****	****	****		
		****	****	****	****	****	****		
	XGB	67.8	46.2	54.1	74.4	67.4	1.7	0.027	4.9
		****	****	****	(****)	****	****		
		****	****		****	*	****		
	E-XGB	78.7	60.1	58.6	72.2	69.9	4.1	1.554	25.4
		****	**	****	()	****	*		
		****	****	(****)	****	(****)			
	Trust	79.6	61.1	61.9	71.8	72.2	5.067	0.14	3.1
		****	****	(****)	****	(****)	()		
	E-Trust	80.7	64.	55.2	75.6	68.2	4.7	1.287	49.0

	RF	93.5	89.2	92.4	94.1	93.5	15.167	0.051	5.0
		****	****	(****)	(****)	(****)	****		
		****	****	****	****	****	****		
	RankBoost	95.6	94.7	56.5	39.2	49.8	79.467	19.086	75.7
		****	****	****	****	****	****		
		****	****	****	****	****	****		
	XGB	93.8	89.4	92.6	94.3	93.8	18.433	0.292	6.3
Spam		****	****	(****)	(****)	(****)	****		
N = 4601		****	****	***	**	****	****		
F = 57	E-XGB	98.6	97.9	93.7	95.	94.9	127.933	11.697	89.3
p = 39%		(****)	(****)	(****)	(****)	(****)	**		
		(****)		(****)	(****)	(****)	****		
	Trust	97.7	96.9	91.	92.8	92.7	147.667	0.29	5.2
		****	****	****	****	****	****		
	E-Trust	98.5	98.	93.2	94.6	94.5	202.933	2.625	69.2
	RF	91.1	76.5	86.2	97.4	91.	10.9	0.116	7.2
		****	****		(*)	****	****		
		****	****	****	****	****	***		
	RankBoost	94.6	79.8	18.3	10.1	49.9	33.433	24.635	84.6
		****	****	****	****	****			
		****	****	****	****	****	(*)		
	XGB	92.7	78.2	87.6	97.5	92.8	6.6	0.101	6.3
Pg-Blox		****	****	(**)	(****)	****	****		
N = 5473		****	****	****	****	*	****		
F = 10	E-XGB	99.2	93.1	88.2	97.6	94.6	28.867	5.784	51.8
p = 10%		()	*	(****)	(****)	**	**		
		*	**		***	(****)	()		
	Trust	99.2	93.4	86.8	97.2	94.9	37.067	0.346	5.9
		****		****	****	(****)	(****)		
	E-Trust	99.2	93.5	88.4	97.7	93.2	27.067	3.687	66.7

	RF	86.6	73.1	83.1	97.1	86.5	25.66	1.227	20.9
		****	****	(****)	(****)	****	****		
		****	****	****	****	****	****		
IJCNN	XGB	92.1	81.4	89.	98.	92.1	17.5	4.431	29.3
		****	(**)	(****)	(****)	(**)	****		
N = 49998		****	****	(****)	(****)	(****)	****		
F = 22	E-XGB	99.4	95.9	85.9	97.	95.7	588.76	65.768	347.5
p = 10%		(****)	(****)	(****)	(****)	(****)	(****)		
		(****)	(****)	(**)	****	(****)	(****)		
	Trust	96.5	79.	65.9	90.9	90.8	110.16	3.095	42.0
		****	****	****	****	(****)	****		
	E-Trust	99.1	94.2	85.2	97.3	89.1	349.8	25.712	192.7

Table 5.3: Bipartite Non-Linear Testing Results

Additional Run-time Results

The radial basis function for SSVM AUC is so slow that it is not worthwhile. That is why various adaptations have been developed. In order to demonstrate its issues we run a special set of experiments with only a single random trial without any hyperparameter tuning.

Data	**FWCD**	**SSVM**	**SVM**	**Trust**
Ion	0.01	226.	0.00	0.02
Boston	0.01	16.	0.00	0.02
Credit	0.02	118.	0.01	0.08
Pima	0.02	5.	0.01	0.05
Student	0.09	4e+4	0.02	0.26
Yeast	0.02	9e+3	0.02	0.14

Table 5.4: Bipartite run time (seconds) for RBF kernel optimizations

Comparing function types

Although the proposed methodology performs best in its class, it is informative to inspect the best method across function types. Typically, the non-linear function performs the best. And, it never falls far below the best top performer. Kernel methods, on the other

hand, sometimes performs near the best but occasionally performs much worse than the other function types.

This table shows the best AUC for each class of algorithm for each dataset. An * indicates that the proposed trust region algorithm achieved the best performance. Across all methods the Trust Region algorithm produced the best AUC on 5 of 8 data sets. In the three where it did not, it was within 0.2 (Ion), 0.0 (Pima), and 0.1 (Spam) of the top AUC. The ensemble methods typically outperform the others, which suggests that proposed extend FWCD for pairwise hinge to enable boosting is a valuable contribution.

Data	**Linear**	**Kernel**	**Ensemble**
Ion	89.7	**97.8**	97.2*
Boston	82.0	76.9	**87.4***
Credit	92.1	74.3*	**93.5***
Pima	82.7*	79.9*	**83.2**
Student	**76.8***	59.4	74.8*
Yeast	72.8*	76.9*	**80.7***
Spam	97.0	95.5	**98.6**
Page Blocks	96.6	96.1*	**99.2***

Table 5.5: Comparison of best AUC performer of each function type

5.2.2 Collaborative Filtering

The trust region for Frank-Wolfe (Trust Linear: linear function; Trust Boost: non-linear function) reduces run time and improves Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) on LETOR Datasets.

5.2.3 Ordered Lists

The proposed trust algorithm performed slightly worse than the SSVM (simplex) however it was able to optimize more efficiently and was more stable with respect to hyperparameters.

Data set	Algorithm	MAP	NDCG	Fit	Total
TD 2003	**Trust Linear**	26.73	52.28	1.84	20.4
	Simplex	23.57	48.39	1.76	75.0
	stat sig	**	*		
	Trust Boost	25.38	48.82	24.5	N/A
	Rank Boost	22.51	48.49	74.3	N/A
	stat sig	**			
TD 2004	**Trust Linear**	22.14	52.46	7.67	41.6
	Simplex	22.78	52.36	2.26	65.8
	stat sig				
	Trust Boost	24.51	53.87	21.9	N/A
	Rank Boost	24.84	54.90	169.8	N/A
	stat sig				
HP 2003	**Trust Linear**	74.90	85.12	5.34	43.39
	Simplex	76.05	85.79	0.96	74.16
	stat sig				
	Trust Boost	79.86	87.33	47.48	N/A
	Rank Boost	72.89	83.48	588.58	N/A
	stat sig	****	***		
HP 2004	**Trust Linear**	69.05	80.62	2.13	26.77
	Simplex	68.99	82.01	0.38	57.65
	stat sig		(*)		
	Trust Boost	71.32	81.69	39.3	N/A
	Rank Boost	74.95	85.60	171.2	N/A
	stat sig	(***)	(**)		
NP 2003	**Trust Linear**	70.37	83.05	6.066	31.66
	Simplex	70.06	83.86	0.91	71.94
	stat sig				
	Trust Boost	70.27	83.15	49.8	N/A
	Rank Boost	74.12	85.07	591.5	N/A
	stat sig	(**)	(**)		
NP 2004	**Trust Linear**	69.43	82.99	0.124	20.27
	Simplex	69.56	84.39	0.36	56.94
	stat sig		(*)		
	Trust Boost	64.96	79.24	38.9	N/A
	Rank Boost	61.35	77.48	166.4	N/A
	stat sig				

Table 5.6: Collaborative Filtering Testing Results

Data set	Algorithm	MAP	NDCG	Fit	Total
OHSUMED	Trust	44.61	66.73	0.12	4.8
	Simplex	45.13	67.25	0.27	49.1
	stat sig				
MQ 2007	Trust	53.64	67.17	1.76	5.9
MQ 2008	Trust	65.21	75.14	0.43	1.57

Table 5.7: List Ranking Testing Results

CHAPTER 6

MULTIVARIATE OUTPUT REGRESSION

Given pairwise relationships between outputs, a Laplacian output kernel can be used to minimize mean squared error. These pairwise relationships come in many forms, a relatable example is the links in social networks. Another used in this research is graphical similarities based on inverse distances. Integrating network information in this way is an efficient approach used in are Gaussian Conditional Random Fields (GCRFs) presented in (Radosavljevic et al. (2010)) and Network Lasso from (D. Hallac and Boyd. (2015)). Alternatively, it is possible to learn link weights. Methods of this nature include Glasso (Friedman et al. (2008)), which is for predicting relationships, or Sparse GCRF (SpGCRF) (Wytock and Kolter (2012)), which learns a network in a Glasso manner while trying to optimize predictions in a GCRF manner.

For the discussion on Gaussian conditional random fields (GCRF) the variables α, β, and Q are different than in the classification sections. The linear weight vector for combining unstructured predictors, α, the linear weight vector for combining graph Laplacians β and the precision matrix Q. GCRF integrates a smoothing operator that uses a Laplacian output kernel, L, where L is the Laplacian of a similarity matrix, S.

6.1 Gaussian Conditional Random Fields

Gaussian Conditional Random Fields have been used for a broad set of applications ranging from climate models, energy forecasting, healthcare, speech recognition, and computer vision (Radosavljevic et al. (2010), Radosavljevic et al. (2014), Djuric et al. (2015), Wytock and Kolter (2013), Guo (2013), Gligorijevic et al. (2015), Polychronopoulou and Obradovic (2014), Khorram et al. (2014), Tappen et al. (2007), Wang et al. (2014)). Yet, the GCRF method has draw backs. Including the fact that it has cubic time complexity per iteration of gradient descent with respect to the number of nodes.

GCRF works by training input learners and feeding those input learners as well as graph structures into a cost function which returns the optimal linear combination of input learners, $\vec{\alpha}$, and the optimal linear combination of graph structures, $\vec{\beta}$, which together produce a prediction. The best combination of input predictions is projected through the graph structure and predictions that are deemed similar by the optimal graph are pulled closer to one another.

Originally, the model was restricted so that $\vec{\alpha}$ were weighted average coefficients. Additionally, all network links were restricted to be positive, which limits the network to pulling values toward one another. The last restriction we will discuss is that GCRF requires the network weights to be symmetric. We address and remove these three restrictions in the next chapter.

The GCRF model is fast and convex. It also has no hyperparameters. The GCRF model is convex and differentiable because the probability function can be mapped to a Multivariate Normal (MVN) distribution given specified conditions. Once in MVN form the method can be optimized using gradient descent. Below we present the mapping for GCRF to Gaussian Normal Form (GNF) to MVN. We present the formulation from Glass et al. (2016) because it is the most concise. This section covers the relationships briefly and a more detailed proof can be found at the end of the chapter. A conditional random

field with quadratic feature and interaction functions define the probability below.

$$P(y|x) = \frac{1}{Z} exp\{-\sum_{i=1}^{N}\sum_{k=1}^{K} \alpha_k (y_i - R_k(x))^2 - \sum_{l=1}^{L}\sum_{i\ j} \beta_l S_{i,j}^l (y_i - y_j)^2\} \tag{6.1}$$

This can be equivalent to the Gaussian normal form given the following conditions.

$$P(y) = \frac{1}{Z} exp\{-\frac{1}{2} y^T Q y + b^T y + c\} \tag{6.2}$$

The condition on the precision matrix.

$$Q = \sum_k \alpha_k I + \sum_j \beta_j \cdot Laplacian(S_j) \tag{6.3}$$

The condition on the Guassian normal form linear vector.

$$b = R\alpha. \tag{6.4}$$

The Gaussian normal form is equivalent to the multivariate normal distribution defined below.

$$P(y|x) = \frac{1}{2\pi|\Sigma|^{1/2}} exp\{-\frac{1}{2}(y-\mu)^T Q (y-\mu)\} \tag{6.5}$$

The one condition is the prediction is equal to the product of the covariance matrix, Q^{-1}, and the linear component, b.

$$\mu = Q^{-1} b. \tag{6.6}$$

The only remaining constraint is that Q is positive semi-definite, which is a bound on convexity, but also a by-product of the multivariate normal form and Gaussian normal form conditions. As long as we satisfy the positive semi-definite constraint, we have a convex model. The first order derivatives can be straightforwardly derived and bounded gradient descent approaches applied. We maximize

$$\mathcal{L}_{GCRF} = -(y^T Q y - 2 y^T Q \mu + \mu^T Q \mu) - log(|Q^{-1}|) \tag{6.7}$$

with respect to α and β by taking steps in the directions given by the following two gradients and contraints.

$$\frac{\partial l}{\partial \alpha_i} = -(y-\mu)^T(y-\mu) + 2(R_i-\mu)^T(y-\mu) + tr(Q^{-1}) \tag{6.8}$$

$$\frac{\partial l}{\partial \beta_i} = -(y+\mu)^T 2L_i(y-\mu) + tr(Q^{-1}L_i) \tag{6.9}$$

$$\alpha_i > 0 \ \forall i \quad \& \quad \beta_j > 0 \ \forall j. \tag{6.10}$$

Equivalent Conditions for Conditional Random Fields and Gaussian Normal Form

Below the detailed equivalence between the conditional random fields with quadratic interaction and the Guassian normal form is shown.

$$\sum_{i=1}^{N} A(\alpha, y_i, \mathbf{x}) + \sum_{j\sim i} I(\beta, y_i, y_j) = -\sum_{i}^{N}\sum_{k}^{K} \alpha_k (y_i - R_{i,k}(X))^2 - \sum_{i\sim j}\sum_{l}^{L} \beta_l S_{l,ij}(y_i - y_j)^2$$

The symbol $\sim$ indicates that i is connected to j. One can equivalently go over all pairs knowing the no connection is synonymous with a similarity of zero.

$$= -\sum^{N}\sum^{K} \alpha_k (y_i - R_{i,k}(X))^2 - \frac{1}{2}\sum_{i}^{N}\sum_{j}^{N}\sum_{l}^{L} \beta_l S_{l,ij}(y_i - y_j)^2$$

Then expand out the quadratic feature functions. This will enable group summations of independent linear and quadratic components.

$$= -\sum^{N}\sum^{K} \alpha_k y_i^2 + \sum^{N}\sum^{K} 2\alpha_k y_i R_{i,k}(X) - \sum^{N}\sum^{K} \alpha_k (R_{i,k}(X))^2 +$$

$$+\sum_{i}^{N}\sum_{j}^{N}\sum_{l}^{L} \beta_l S_{l,ij} y_i y_j - \frac{1}{2}\sum_{i}^{N}\sum_{j}^{N}\sum_{l}^{L} \beta_l (S_{l,ij} + S_{l,ji}) y_i^2$$

Given the following equivalence, the equivalence to the Gaussian normal form can be established.

$$\sum_{i}^{N}\sum_{j}^{N}\sum_{l}^{L}\beta_l(S_{l,ij}+S_{l,ji})y_i^2=\sum_{i}^{N}\sum_{l}^{L}(\sum_{\forall j}S_l,ij+\sum_{\forall j}S_l,ji)\cdot\beta_l y_i^2$$

The probability for the Gaussian normal form.

$$P(y)=-\frac{1}{2}y^TQy+y^Tb+c=\sum_{i}^{N}\sum_{j}^{N}Q_{ij}y_iy_j+\sum_{i}^{N}y_ib_i+c$$

The final requirement for equivalence are given below.

$$\sum_{i}^{N}\sum_{j}^{N}Q_{ij}y_iy_j=(\sum^{K}\alpha_k)\sum^{N}y_i^2+\sum_{i}^{N}\sum_{l}^{L}(\sum_{\forall j}S_l,ij+\sum_{\forall j}S_l,ji)\cdot\beta_l y_i^2-\sum_{i}^{N}\sum_{j}^{N}\sum_{l}^{L}\beta_l S_{l,ij}y_iy_j$$

Sparse GCRF

The Sparse GCRF method produces a unique function for each node in a graph. That graph is modelled through time. It works by having a single set of features for the entire graph at each time step. As opposed to GCRF, this method learns similarities. These similarities are required to be symmetric. The convergence time for this method is very inconsistent and depends on the data input to the model. A setup with outputs = 80, time steps = 1, and features = 4 has taken anywhere from 0.2 seconds to 44 seconds. The approach scales cubically with the number of outputs.

6.1.1 Parameter Domain

In research published in Glass et al. (2016), GCRF is extend to allow negative parameters or input negative link weights. This research expands on the model of Radosavljevic et al. (2010) by allowing a broader range of both linear combinations of unstructured predictors and network structures. As mentioned, the previous implementation could only take a weighted average of both unstructured predictors and link weights and also required that all link weights be positive.

Precision Matrix Decomposition

The convexity requirement of GCRF is to maintain positive definiteness of the precision matrix. The original GCRF proposed in Radosavljevic et al. (2010) is expanded in the case with one similarity network. In the presented formulation, the boundary conditions for positive definiteness are written as a set of linear equations of our parameters. This is the ideal for constrained gradient descent and allows negative parameter values into the convex optimization space for this problem. Because the focus is on the case where there is only one similarity network, the new model is referred to as Unimodal GCRF (UmGCRF). Begin by diagonalizing Q. We know $L = UDU^T$ where $UU^T = I$ and D is a diagonal matrix because L is a symmetric real valued matrix.

$$\begin{aligned} Q &= \sum_k \alpha_k I + \beta L \\ &= \sum_k \alpha_k I + \beta U D U^T \\ &= U((\sum_k \alpha_k) \cdot U^T I U + \beta D) U^T \\ &= U(\sum_k \alpha_k I + \beta D) U^T \end{aligned} \tag{6.11}$$

Then, $Q = U \Lambda U^T$ where Λ is diagonal matrix, with diagonal elements:

$$\lambda_i = \sum_k \alpha_k + \beta d_i \quad \forall i$$

Exact bounds for convexity for the uni-modal case are provided. The parameter search space is show to be convex by establishing that covariance matrix is Positive Definite subject to our boundary conditions. This is the main claim necessary for GCRF convexity.

Theorem 5 (GCRF Precision Matrix Positive Semi-Definiteness)**.** *The precision matrix, Q, is positive definite is positive definite so long as cumulative unstructured predictor*

weights $\sum_i \alpha_i$ and the weighted eigenvalues of the similarity network Laplacian sum to greater than or equal zero.

$$\mathbf{Q} \succeq 0 \Leftrightarrow \begin{cases} \sum_k \alpha_k + \beta d_0 \geqslant 0 \\ \sum_k \alpha_k + \beta d_{n-1} \geqslant 0 \end{cases} \tag{6.12}$$

Proof. Start with a theorem established in Ayres (1967): Given a real symmetric matrix, Q, $\exists\ U$ such that $Q = U\Lambda U^T$ and $Q \succ 0 \Leftrightarrow \lambda_i > 0\ \forall i$ where λ_i are the diagonal entries in Λ. Next, substitute λ_i with a function in terms of our parameters: $\lambda_i = \sum_k \alpha_k + \beta d_i$. Since D is a diagonalized matrix we know that d_i are in ascending order, d_0 being lowest and d_{n-1} being highest. As a result,

$$\beta d_{n-1} \geqslant ... \geqslant \beta d_0 \text{ if } \beta \geqslant 0$$
$$\beta d_{n-1} \leqslant ... \leqslant \beta d_0 \text{ if } \beta \leqslant 0$$

Since $\sum_k \alpha_k$ effects each diagonal equally, $\forall \beta$, each diagonal entry in Λ is in between λ_{n-1} and λ_0. Thus, only the outermost constraints are required to ensure positive definiteness. □

With linear boundary conditions, the optimization can be done with an interior point bounded gradient descent. Below are graphical representations of the new parameter search space. Previously, all searches were restricted to the first quadrant. In the proposed case, if $d_0 = 0$ then we search the entire first quadrant and additional space.

In order to demonstrate the additional modeling capacity of the new parameter search space, we walk through a simple example case. In this case there are only two targets. The right panel in Figure 6.2 shows the smoothing behavior of traditional GCRF pulling updated predictions towards each other, but away from their true value. The left panel shows the behavior possible as a result of negative links or negative betas. We can now push values away from each other, and in this toy case toward their true values.

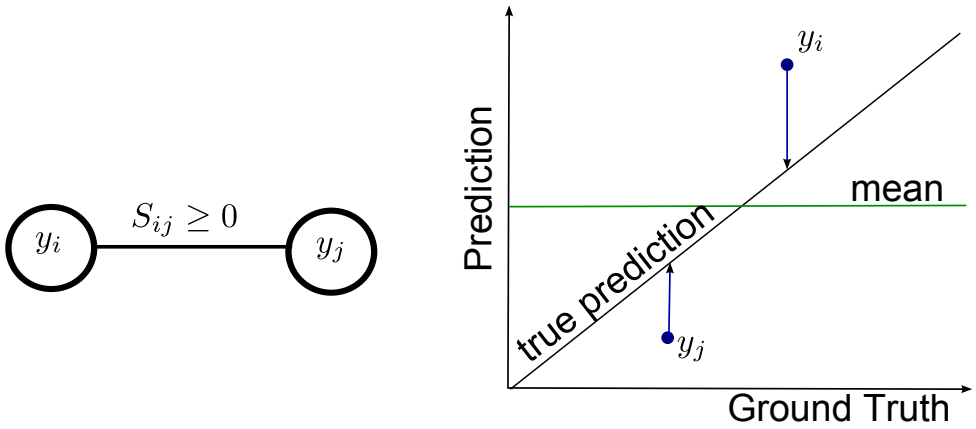

FIGURE 6.1: GCRF attractive force

GCRF gravitate unstructured predictions y_i and y_j closer to the target 45° line when the similarity S_{ij} has positive influence. However, the similarity might have negative influence on the GCRF prediction. Figure 6.2 (right panel) represents the situation when positive link alienate predictions from the ground truth (blue lines that are gravitating toward mean of predictions). While, on the left panel, the negative weights ($\beta S_{ij} \leqslant 0$) would push predictions from each other and thus towards true solution.

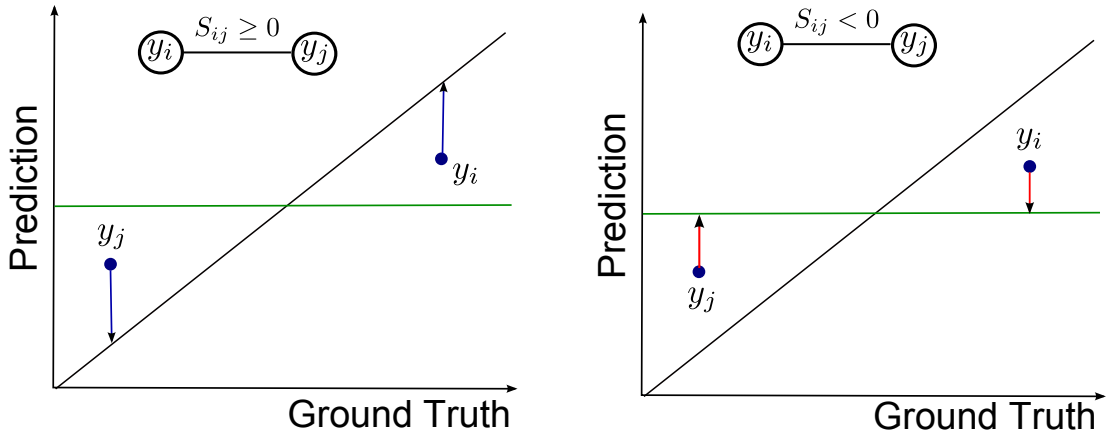

FIGURE 6.2: GCRF repulsive force

Assume that there are two unstructured predictors with weights α_1 and α_2. Similarly to the above case, it is possible to see that the necessary and sufficient conditions for positive definite Q is that $\alpha_1 + \alpha_2 + \beta > 0$ and $\alpha_1 + \alpha_2 > 0$. The second condition $\alpha_1 > -\alpha_2$ is

interesting. The exact values of α do not matter, only the ratio of α's matters. So figure 6.3 (right) shows the search space of α's on the projected unit circle (the ratio between α_1 and α_2). The original GCRF restricts α's to the first quadrant of the search space (all parameters are greater than 0), whereas the the optimal search space allows more possible combinations.

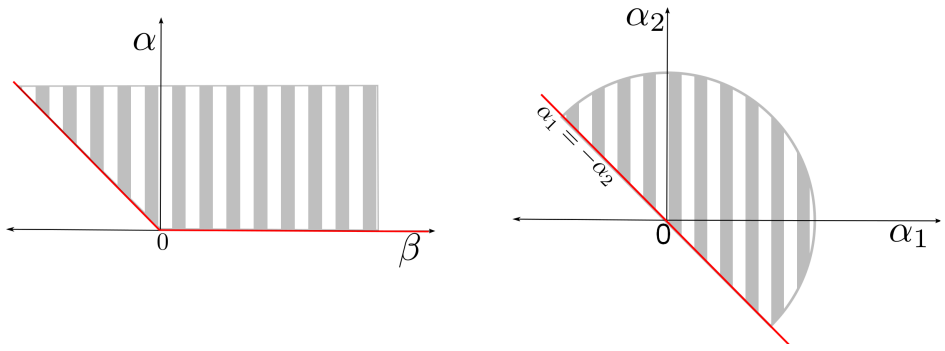

FIGURE 6.3: Extended feasible parameter space for GCRF

In order to expand the power and scope of our model we will include an additional rank one similarity network that will serve as a bias term for network weights. This will enable the model to incorporate networks with non-negative weights. These network weights can be the result of other research or Gaussian kernels or cosine similarity. By introducing an intercept Matrix $J = \vec{1}\vec{1}^T$, one can shift similarity weights so that they are centered around any chosen point. Note here that the optimization for this shift is convex. This bias term can be maintained throughout all the advances discussed in this and the next section. For the below equation, know that the Laplacian of $\frac{1}{n}J$ is $I - \frac{1}{n}J$.

$$Q = \sum_k \alpha_k I + \beta L + \text{ç} I - \text{ç}\frac{1}{n}J \tag{6.13}$$

Laplacian matrices have been studied in depth. In Merris (1998), it was proven that all Laplacian matrices have an eigenvector of $v = \vec{1}/|\vec{1}|$ and an associated $\lambda = 0$. In Ding and Zhou (2007), it was shown that the perturbation of a matrix by a rank one matrix that has a basis in the original matrix only contributes to the the eigenvalue associated with that

basis. Since it is known that $v = \vec{1}/|\vec{1}| \in U$ this implies $U^T \frac{1}{n} JU = \frac{1}{n}(U^T\vec{1})(U^T\vec{1})^T = D_j$, a matrix of zeros with a one on the diagonal entry associated with $col(U) = v$. With this established, one can diagonalize Q.

$$\begin{aligned} Q &= \sum_k \alpha_k I + \beta L + \text{ç} I - \text{ç}\frac{1}{n} J \\ &= \sum_k \alpha_k I + \beta U D U^T + \text{ç} I - \text{ç}\frac{1}{n} J \\ &= U((\text{ç} + \sum_k \alpha_k) \cdot U^T I U + \beta D - \frac{\text{ç}}{n} U^T J U) U^T \\ &= U(\sum_k \alpha_k I + \text{ç} I + \beta D - \text{ç} D_J) U^T \end{aligned} \tag{6.14}$$

Since the bias term can now be included, ç$/\beta$, all weights can be shifted into the non-negative space and still map the original values in addition to a broader range of values.

$$I(\beta, y_i, y_j, \text{ç}) = -\sum_{l=1}^{L} \sum_{i \sim j} (\beta_l S_{ij}^l + \text{ç})(y_i - y_j)^2 \tag{6.15}$$

Having strictly non-negative weights makes notation for the following notation simpler but it is not necessary that weights be non-negative. For Laplacian matrices with non-negative similarity weights, it is known that $\vec{1}/|\vec{1}|$ is associated with the lowest eigenvalue, λ_0. Then, $Q = U \Lambda U^t$ where Λ is diagonal matrix, with diagonal elements:

$$\begin{aligned} \lambda_i &= \sum_k \alpha_k + \text{ç} + \beta d_i \text{ if } i \neq 0 \\ \lambda_0 &= \sum_k \alpha_k + \beta d_0 \end{aligned} \tag{6.16}$$

Theorem 6 (Systemic Repulsion)**.** *All of the predictions are able to be pushed apart by ç and the precision matrix remains positive semi-definite.*

$$\mathbf{Q} \succeq 0 \Leftrightarrow \begin{cases} \sum_k \alpha_k + \beta d_0 \geqslant 0 \\ \sum_k \alpha_k + \text{ç} + \beta d_1 \geqslant 0 \\ \sum_k \alpha_k + \text{ç} + \beta d_{n-1} \geqslant 0 \end{cases} \tag{6.17}$$

Proof. We use the theorem established in Ayres (1967) to state $Q \succeq 0 \Leftrightarrow \lambda_i > 0 \forall i$ where λ_i. Substitute lambda for a function with respect to the models parameters (eq.2). Since D is a diagonalized matrix we know that d_i are in ascending order, d_0 being lowest and d_{n-1} being highest. As a result,

$$\beta d_{n-1} \geqslant ... \geqslant \beta d_0 \text{if } \beta \geqslant 0$$
$$\beta d_{n-1} \leqslant ... \leqslant \beta d_0 \text{if } \beta \leqslant 0$$

Thus λ_{n-1} through λ_1 are greater than zero so long as:

$$\textstyle\sum_k \alpha_k + \text{ç} + \beta d_1 > 0$$
$$\textstyle\sum_k \alpha_k + \text{ç} + \beta d_{n-1} > 0$$

This leaves us with a final constraint $\sum_k \alpha_k + \beta d_0 > 0$ and we have established bounds on positive definiteness for Q. □

When experimenting with UmGCRF, we found that a regularization component was helpful to reduce testing error. So in the final version, we apply a ridge regularization to ç.

Parameter Ratio Corrollary

The important information provided by $\sum \alpha$ and $\sum \beta$ are ratios. Thus, the dimensionality of the parameters can be reduced.

$$\mu_i = 2\alpha \sum_{j=1}^{n} \frac{U_{i,j} c_j}{2(\alpha + \beta d_j)} = \sum_{j=1}^{n} \frac{U_{i,j} c_j}{(1 + \frac{\beta}{\alpha} d_j)} \tag{6.18}$$

6.1.2 Gradient Iteration Time Complexity

Previous approaches required matrix inversion in order to calculate the first order derivatives. This slowed down gradient descent at each iteration by $O(n^3)$. Additionally, the previous

methods inferred μ at every iteration as input to calculate the first order derivatives, a cost of $O(n^2)$.

$$\frac{\partial l}{\partial \alpha_i} = -(y-\mu)^T(y-\mu) + 2(R_i-\mu)^T(y-\mu) + tr(Q^{-1}) \tag{6.19}$$

$$\frac{\partial l}{\partial \beta_i} = -(y+\mu)^T 2L_i(y-\mu) + tr(Q^{-1}L_i) \tag{6.20}$$

$$l(\alpha,\beta) = -\frac{1}{2}(y-\mu)^T Q(y-\mu) - log|Q^{-1}|^{\frac{1}{2}} \tag{6.21}$$

Only the trace of the inverted matrix is needed, so solving for the eigenvalues directly makes the same calculation $O(n)$. This is side-stepped the need to infer μ for the first order derivatives. Te eigen-decomposition of the Laplacian of the Similarity matrix occurs before gradient descent, then each iteration only takes $O(n)$ operations.

$$Tr(Q^{-1}) = Tr(U\Lambda^{-1}U^T) = Tr(\Lambda^{-1}) = \sum_{\forall i} \lambda_i^{-1} \tag{6.22}$$

$$Tr(Q^{-1}L) = Tr(U\Lambda^{-1}U^T UDU^T) = Tr(\Lambda^{-1}D) = \sum_{\forall i} \lambda_i^{-1} d_i \tag{6.23}$$

$$log(|Q^{-1}|^{\frac{1}{2}}) = \frac{1}{2} log(\prod_{\forall i} \lambda_i^{-1}) = -\frac{1}{2}\sum_{\forall i} log(\lambda_i) \tag{6.24}$$

Below are the linear time computations. They replace operations that can occur outside gradient descent with x's. Also, note that $C = U^T R$ is where C comes from and occurs outside the gradient update procedure.

$$\frac{\partial l}{\partial \alpha} = \frac{-1}{2}(x_1 + 2(C_i \odot \lambda^{-1})C\alpha + \alpha^T C^T \Lambda^{-2} C\alpha) - 1^T\lambda^{-1}) \tag{6.25}$$

$$\frac{\partial l}{\partial \beta} = \frac{-1}{2}(x_2 - \alpha^T C^T((d \odot \lambda^{-2})1^T \odot C)\alpha) + \frac{1}{2}d^T\lambda^{-1} \tag{6.26}$$

$$l = -\frac{1}{2}(p^T\lambda - 2\alpha^T R^T y + 4\alpha^T C^T \Lambda^{-1} C\alpha + log(1^T\lambda^{-1})) \tag{6.27}$$

Linear Time Gradients

Here the process in order to derive gradients which can be computer in a linear number of steps is presented in greater detail.

$$\begin{aligned}
\frac{\partial l}{\partial \alpha_i} &= \frac{-1}{2}(y^T y + 2R_i^T(\mu - y) + \mu^T\mu) + \frac{1}{2}Tr(Q^{-1}) \\
&= \frac{-1}{2}(y^T y - 2R_i^T y + 2R_i^T\mu + \mu^T\mu) + \frac{1}{2}Tr(Q^{-1}) \\
&= \frac{-1}{2}(y^T y - 2R_i^T y + 2R_i^T Q^{-1}b + (Q^{-1}b)^T Q^{-1}b) + \frac{1}{2}Tr(Q^{-1}) \\
&= \frac{-1}{2}(y^T y - 2R_i^T y + 2R_i^T U\Lambda^{-1}U^T b + b^T U\Lambda^{-2}U^T b) + \frac{1}{2}Tr(\Lambda^{-1}) \\
&= \frac{-1}{2}(y^T y - 2R_i^T y + 2R_i^T U\Lambda^{-1}U^T R\alpha + \alpha^T R^T U\Lambda^{-2}U^T R\alpha) + \frac{1}{2}Tr(\Lambda^{-1})
\end{aligned} \tag{6.28}$$

$$
\begin{aligned}
l(\epsilon|\alpha,\beta) &= -\frac{1}{2}\epsilon^T Q\epsilon - log(Z) \\
&= -\frac{1}{2}(y-\mu)^T Q(y-\mu) - log|Q^{-1}|^{\frac{1}{2}} \\
&= -\frac{1}{2}(y^T Qy - 2y^T Q\mu + \mu^T Q\mu) - \frac{1}{2}tr(Q^{-1}) \\
&= -\frac{1}{2}(y^T U\Lambda U^T y - 2y^T QQ^{-1}b + (Q^{-1}b)^T QQ^{-1}b) - \frac{1}{2}1^T\lambda^{-1} \\
&= -\frac{1}{2}(\tilde{y}^T\Lambda\tilde{y} - 2y^T QQ^{-1}b + (Q^{-1}b)^T QQ^{-1}b) - \frac{1}{2}1^T\lambda^{-1} \\
&= -\frac{1}{2}(\tilde{y}^T\Lambda\tilde{y} - 2y^T b + b^T Q^{-1}b) - \frac{1}{2}1^T\lambda^{-1} \\
&= -\frac{1}{2}((\tilde{y}\cdot\tilde{y})^T\lambda - 2y^T(2R\alpha) + (2R\alpha)^T Q^{-1}(2R\alpha)) - \frac{1}{2}1^T\lambda^{-1} \\
&= -\frac{1}{2}((\tilde{y}\cdot\tilde{y})^T\lambda - 2y^T(2R\alpha) + 4\alpha^T R^T U\Lambda^{-1}U^T R\alpha) - \frac{1}{2}1^T\lambda^{-1} \\
&= -\frac{1}{2}((\tilde{y}\cdot\tilde{y})^T\lambda - 4y^T R\alpha + 4\alpha^T C^T\Lambda^{-1}C\alpha) - \frac{1}{2}1^T\lambda^{-1} \\
&= -\frac{1}{2}((\tilde{y}\cdot\tilde{y})^T\lambda - 4y^T R\alpha + 4(C\alpha\cdot C\alpha)^T\Lambda^{-1}) - \frac{1}{2}1^T\lambda^{-1}
\end{aligned}
\tag{6.29}
$$

Time Series Optimization

Optimizations which do not share information over time can use specific designs in order to improve efficiency. GCRF and SpGCRF are designed to predict multiple outputs over multiple time steps. Both assume independence across time conditional on the joint probability of the outputs given the inputs. If one does not code GCRF with this type of application in mind, the method will be far less efficient. By introducing some specific data structures we can greatly reduce the space and time used during GCRF optimization.

$$\mu = Q^{-1}b = Q^{-1}R\alpha$$

$$\left|\begin{array}{ccc} Q_i^{-1} & 0 & 0 \\ 0 & Q_{ii}^{-1} & 0 \\ 0 & 0 & Q_{iii}^{-1} \end{array}\right|\left|\begin{array}{c} X \\ Y \\ Z \end{array}\right| = \left|\begin{array}{c} Q_i^{-1}X \\ Q_{ii}^{-1}Y \\ Q_{iii}^{-1}Z \end{array}\right|$$

The same reformulation of GCRF into block diagonal components can be applied to UmGCRF. Our eigenvector matrix is reduced to a rectangle with respect to the number of observations and predictors: $U_{(nxn)} \rightarrow U_{(nxm)}$. Similarly, the eigenvalue matrix is reduced in size : $\Lambda_{(nxn)} \rightarrow \Lambda_{(nxm)}$.

$$\left|\begin{array}{ccc} U_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & U_p \end{array}\right|\left|\begin{array}{ccc} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_n \end{array}\right| = \left|\begin{array}{c} U_1 \\ U_2 \\ U_2 \end{array}\right|\left| \begin{array}{c} [\lambda_1 : \lambda_m]^T\vec{1}^T \ ... \ [\lambda_{n-m} : \lambda_n]^T\vec{1}^T \end{array}\right|$$

$$\mu = \left|\begin{array}{c} U_1 \\ U_2 \\ U_2 \end{array}\right|\left| \begin{array}{c} [\lambda_1 : \lambda_m]^t\vec{1}^t \ ... \ [\lambda_{n-m} : \lambda_n]^t\vec{1}^t \end{array}\right| C\alpha$$

For faster iteration convergence, 2nd order methods are used. Once again, naive would perform this task much more slowly. BFGS approximations were used in practice, but an exact implementation of Newton-Raphson's method is achievable and runs even more quickly. Below are the solutions for all $H_{i,j}$ as functions of α and β, all of which have been reduced to $O(n)$ calculations.

$$\frac{\partial^2 l}{\partial\alpha_i\partial\alpha_j} = -16\alpha^T C^T\Lambda^{-3}C\alpha + 8\alpha^T C^T\Lambda^{-2}(c_i + c_j) + c_i^T\Lambda^{-2}c_j - 2(1^T\lambda^{-2})$$

$$\frac{\partial^2 l}{\partial\alpha_i\partial\beta_j} = -16\alpha^T C^T D_j\Lambda^{-3}C\alpha - 8\alpha^T C^T D_j\Lambda^{-2}c_i - 2(d_j^T\lambda^{-2})$$

$$\frac{\partial^2 l}{\partial\beta_i\partial\beta_j} = -16\alpha^T C^T D_i D_j\Lambda^{-3}C\alpha - 2(d_i \cdot d_i)^T\lambda^{-2}$$

$$l = -\frac{1}{2}(p^T\lambda - 2\alpha^T R^T y + 4\alpha^T C^T\Lambda^{-1}C\alpha + log(1^T\lambda^{-1}))$$

6.1.3 Nested Network Optimization

As discussed above, structure based regression algorithms suffer speed losses and unsurprising increase demand on memory. Gaussian Conditional Random Field (GCRF) models are one of the most time and memory efficient approaches to structured regression, but still have limitations. The established speed ups for the GCRF method allow for exact solutions on networks of up to a hundred thousand nodes and ten million links. Using multi-scale networks, the exact solution for networks of millions of nodes and trillion of links can be solved with linear time complexity as presented in Glass and Obradovic (2017). We walk through the intuitiveness of using multiple scales of networks on a real-life health-informatics application. The time and memory demands from using this approach are logarithmic compared to naive implementations.

Some similarity networks can be built by interweaving more than one network. In the motivating example, we predict admissions by disease for various hospitals in California. Two networks were obtainable. One compares diseases to diseases and another compares hospitals to hospitals. Intuitively, we can visualize each hospital containing a network of disease nodes. The diseases interact each other within and across hospitals. A network with networks inside nodes is a multi-scale network. Multi-scale networks can be modeled as a Kronecker product of networks. A $\otimes$ B signifies that B is nested within A. An example to illustrate our case is when we have a network of disease similarities (6.4 on the right), and we also have a network of hospital similarities (6.4 on the left). Each node on the left has a graph within it which looks like the graph on the right.

In figure 6.4, the **top left** shows an outer network (hospitals in our example), the **top right** shows an inner network, located inside a node of the outer network (diseases in our example), the **bottom left** shows a single link from the outer network (a relationship between hospitals in our example), and the **bottom right** shows two joined inner networks (a network of the relationships between all diseases admission rates at two different hospitals

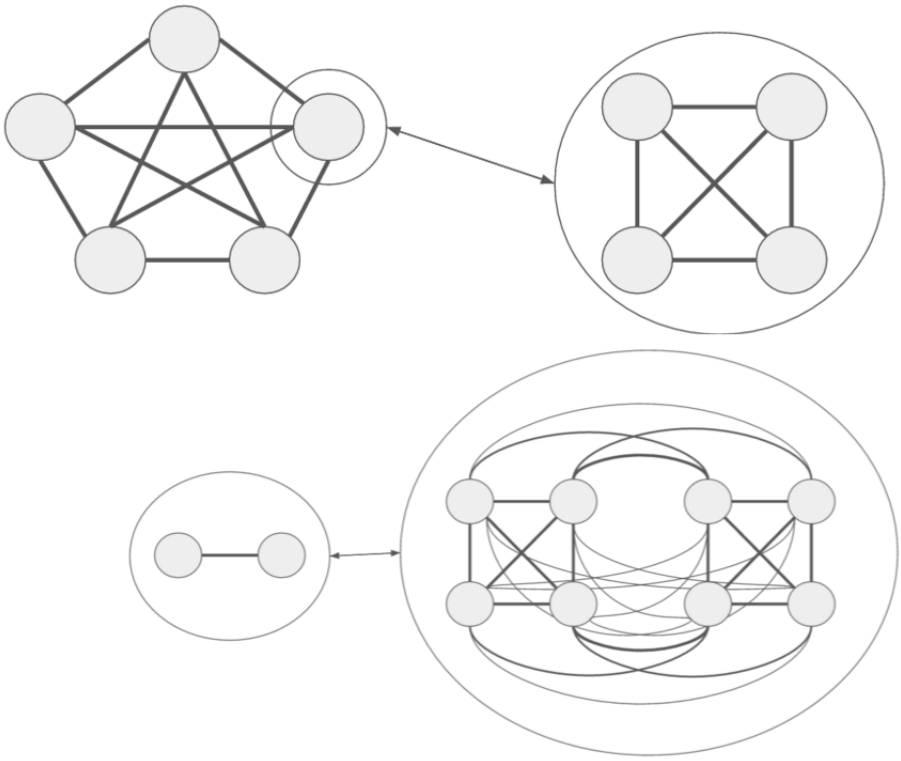

FIGURE 6.4: Example of nested network structures: two networks and their outer product of link information

our example).

Structured regression methods are expected to achieve higher accuracy compared to unstructured regression methods. The drawback of structured methods is usually computational complexity. On problems similar to our motivating multi-scale problem, GCRF-MSN computes faster than VAR. When using structured methods, one must choose from many network weight design choices including: robust weight updating, sparse weight learning, or weights as prior information. GCRF as introduced by Radosavljevic et al. (2010) uses network weights as prior information. Using network weights as a prior is by far the fastest approach. The multi-scale structured regression approach described in this paper finds the optimal solution to the GCRF problem. When using multi-scale networks the proposed method can handle trillions of links in minutes while alternatives require weeks or months.

We introduce an implementation of GCRF which operates on nested network similarities, otherwise known as a Kronecker product of matrices. This method is much faster and requires less memory than standard approaches. It also provides a framework for incorporating various types of network information into a structured regression. In order to develop this model, we also formally introduce two properties of Laplacians of Kronecker products currently absent from literature.

Laplacian of Kronecker Product

Despite the vast literature on Kronecker products and Laplacian matrices Schäcke (2013); Zumstein (2005); Lovasz (2004), there is not a reference which contains the formula for the Laplacian of the Kronecker products of matrices. To define the Laplacian of a Kronecker product we use the following notation: similarity network, S, has a diagonal sum matrix, $D(S)$, each entry of which is denoted $d_i^{(1)}$. The Laplacian of S is $L(S)$, and the standard formula for the Laplacian is written as $L(S) = D(S) - S$. The Laplacian of a Kronecker product is given by equation 6.30.

$$L(S_1 \otimes S_2) = D(S_1) \otimes D(S_2) - S_1 \otimes S_2. \tag{6.30}$$

Efficient Decomposition

With this new formula for the Laplacian of the Kronecker of two adjacency matrices we can formalize the eigendecomposition of the Laplacian with respect to its input matrices. In order to do this, we use a regularized Laplacian. A regularized Laplacian is of the form,

$$\mathcal{L}(S) = I - D(S)^{-1/2} \;\; S \;\; D(S)^{-1/2}. \tag{6.31}$$

It has specific bounds on the sum of its eigenvalues and the rows and columns of the S matrix are now normalized. We denote the unnormalized network weight matrix, S_0. One can obtain a regularized Laplacian by normalizing S_0. Perform the following operation

$$S = D(S_0)^{-1/2} \;\; S_0 \;\; D(S_0)^{-1/2}. \tag{6.32}$$

With our regularized Laplacian, we have a Laplacian where the eigenvectors are determined entirely by the eigenvectors of the input adjacency matrix, S.

$$\begin{aligned}\mathcal{L}(S) &= I - S \\ &= I - U\Lambda_S U^T \\ &= U(I - \Lambda_S)U^T.\end{aligned} \tag{6.33}$$

Applying our definition of the Laplacian of a Kronecker to our normalized adjacency matrices, we get

$$\mathcal{L}(S_1 \otimes S_2) = I_1 \otimes I_2 - S_1 \otimes S_2. \tag{6.34}$$

One can perform eigen-decomposition on each adjacency matrix in relatively little time. It is then possible to exploit known properties of the Kronecker product of decomposed matrices, as seen below:

$$\begin{aligned}&= I_1 \otimes I_2 - (U_1\Lambda_{S_1}U_1^T) \otimes (U_2\Lambda_{S_2}U_2^T) \\ &= I_1 \otimes I_2 - (U_1 \otimes U_1^T)(\Lambda_{S_1} \otimes \Lambda_{S_2})(U_1 \otimes U_2)^T.\end{aligned} \tag{6.35}$$

Finally, to get the formula written in a way which yields computational speed up for GCRF, we must projecting the identity matrix onto the new orthonormal basis. The projection of an identity matrix onto a new orthonormal basis produces an identity matrix, thus

$$\mathcal{L}(S_1 \otimes S_2) = (U_1 \otimes U_1^T)(I_1 \otimes I_2 - \Lambda_{S_1} \otimes \Lambda_{S_2})(U_1 \otimes U_2)^T. \tag{6.36}$$

Nested Network GCRF Precision Matrix Formula

We have diagonalized the Laplacian of a Kronecker product of networks, by only operating on the component networks before the Kronecker product was taken. Diagonalizing in this manner is magnitudes more efficient than a naive approach. Revisiting GCRF's formulation for the precision matrix, Q, we can see that the Kronecker product can be

tucked inside the Laplacian transformation.

$$\begin{aligned} Q &= \sum_k \alpha_k I + \beta L(\otimes_{\forall j} S_j) \\ &= (U_1 \otimes U_1^T)((\sum_k \alpha_k + \beta)I - \beta \Lambda_{S_1} \otimes \Lambda_{S_2})(U_1 \otimes U_2)^T \end{aligned} \tag{6.37}$$

GCRF provides a closed form solution for the partition function, but it is computed in $O(n^3)$ operations. In Glass et al. (2016), the GCRF learning process was reduced from $O(n^3 \cdot I)$ to $O(n^3 + n \cdot I)$. The remaining bottleneck can be addressed with GCRF-MSN in $O(h^3 + d^3 + n \cdot I)$ such that $n = h \cdot d$, where h represents the outer graph and d the inner graph, which in our example are h, the number of hospitals, and d, the number of diseases. The speed up varies according to the size of the subnetworks. The optimally efficient case of GCRF-MSN can be seen by minimizing $h^3 + d^3$ such that $n = h \cdot d$. This results in $h = d = \sqrt{n}$. Then, one can see that $h^3 + d^3 = 2(\sqrt{n})^3 = 2n^{3/2}$, which implies the entire GCRF learning process would be $O(n^{3/2} + n \cdot I)$. If we nest more than two networks the speed ups are even more dramatic.

The current method is also more efficient at storing information. In our motivating example on HCUP data, the network changes over time. This is often referred to as an evolving network. We end up with a unique network for every year spanning 9 years. Predicting 200 diseases across 500 hospitals means predicting 100,000 different node values each month.

So, each monthly network has 10 billion links. GCRF-MSN can optimize parameters while not storing every link. It needs only 312,500 links. What is more, we can capture the interaction between these networks without executing the Kronecker product. The space requirement of GCRF-MSN is only logarithmic compared to traditional GCRF or other network methods. We manage to expand upon recent advances in speed and modeling capacity from Glass et al. (2016). Here, we show that we can extend this definition to networks built on the combination of multiple scales of networks via the

Kronecker product. The new equivalent set of constraints are given by the following lemma. This enables us to utilize improvements introduced in Glass et al. (2016) for multiscale networks (GCRF-MSN).

Lemma 7 (Nested Network Positive Semi-Definiteness)**.**

$$\mathbf{Q} \succeq 0 \Leftrightarrow \begin{cases} \sum_k \alpha_k + \beta \prod_{l=1}^{L} d_{l,0} \geqslant 0 \\ \sum_k \alpha_k + \beta d_{j,0} \prod_{l \neq j}^{L} d_{l,(n-1)} \geqslant 0 \ \ \forall j \\ \sum_k \alpha_k + \beta \prod_{l=1}^{L} d_{l,(n-1)} \geqslant 0 \end{cases} \tag{6.38}$$

Proof. It was proven previously that only the smallest and largest eigenvalues must be greater than zero for all eigenvalues of Q to be guaranteed to be greater than zero Glass et al. (2016). We know the multiplication of ordered positive numbers maintains ordinal relationships. If all the links are positive, we know the diagonal of the Laplacian has all non-negative entries in which case the max and min can be derived.

$$Max(\Lambda_{\otimes_{l=1}^{L} S_j}) = \prod_{l=1}^{L} Max(\Lambda_{S_l})$$

$$Min(\Lambda_{\otimes_{l=1}^{L} S_j}) = \prod_{l=1}^{L} Min(\Lambda_{S_l}).$$

If some of the links are negative, there may be a negative diagonal entry in any one of Λ_{S_i}. With the reasonable assumption that the absolute value of the maximum eigenvalue is larger than the absolute value of the minimum eigenvalue for each matrix S_i, we can exhaustively search all possible minimums relatively efficiently by multiplying the maximum of all the matrices excluding one. We then multiply that value by the minimum eigenvalue of the excluded matrix. Once cycled through all input matrices, we have produced all possible minimums

$$PossibleMin(\Lambda_{\otimes_{l=1}^{L} S_j}) = d_{j,0} \prod_{l \neq j}^{L} d_{l,(n-1)}.$$

Since we can compute the precision matrix's eigenvalues as linear equations with respect to the Kronecker product of the diagonal matrices of the network structures, we can compute the maximum value of an eigenvalue and minimum value of an eigenvalue and then have linear bounds on positive semi-definiteness. □

Laplacian of Kronecker Product

To define the Laplacian of a Kronecker product we use the following notation: similarity network, S, has a diagonal sum matrix, $D(S)$, each entry of which is denoted $d_i^{(1)}$. The Laplacian of S is $L(S)$, and the standard formula for the Laplacian is written as $L(S) = D(S) - S$. We then can formulate $d_i^{(1)}$ as a summation of the entries of S, denoted $s_{i,j}$,

$$d_i^{(1)} = \sum_{j=1}^{n_1} s_{i,j}^{(1)}.$$

Lemma 8 (Laplacian of a Kronecker product)**.** *The Laplacian of a Kronecker product is*

$$L(S_1 \otimes S_2) = D(S_1) \otimes D(S_2) - S_1 \otimes S_2.$$

Proof. The Kronecker product can be concisely represented via block matrices. The Kronecker of diagonal matrices is particularly clean.

$$D(S_1) \otimes D(S_2) = \begin{vmatrix} d_1^{(1)} D(S_2) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & d_{n_1}^{(1)} D(S_2) \end{vmatrix}.$$

When one fully carries out the multiplication of the above block matrix, it is clear that the Kronecker product of diagonal matrices is diagonal, and the exact diagonals are easy to position using ceiling and modulo indexing

$$= \begin{vmatrix} d_1^{(1)} d_1^{(2)} & 0 & 0 & 0 & 0 \\ 0 & d_1^{(1)} d_2^{(2)} & 0 & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & d_{n_1}^{(1)} d_{n_2-1}^{(2)} & 0 \\ 0 & 0 & 0 & 0 & d_{n_1}^{(1)} d_{n_2}^{(2)} \end{vmatrix}.$$

The above matrix shows a diagonal matrix with entries that are a product of the diagonal entries of smaller matrices. The diagonal entries of the smaller matrices are computed by summing across each row of the matrix, as formulated above. Now compute the Kronecker product of $S_1 \otimes S_2$, and calculate the diagonal of this matrix, and verify that it equals the above matrix

$$S_1 \otimes S_2 = \begin{vmatrix} s_{1,1}^{(1)} \cdot s_{1,1}^{(2)} & \cdots & s_{1,1}^{(1)} \cdot s_{1,n_2}^{(2)} \\ s_{1,1}^{(1)} \cdot s_{2,1}^{(2)} & \cdots & s_{1,1}^{(1)} \cdot s_{2,n_2}^{(2)} \\ \vdots & \ddots & \vdots \\ s_{n_1,1}^{(1)} \cdot s_{n_2-1,1}^{(2)} & \cdots & s_{n_1,n_1}^{(1)} \cdot s_{n_2-1,1}^{(2)} \\ s_{n_1,1}^{(1)} \cdot s_{n_2,1}^{(2)} & \cdots & s_{n_1,n_1}^{(1)} \cdot s_{n_2,n_2}^{(2)} \end{vmatrix}.$$

Use index k to represent positions in the matrix $S_1 \otimes S_2$ and $D(S_1 \otimes S_2)$. It is not hard to verify

$$d_k = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} s_{i,ceil(k/n_1)}^{(1)} \cdot s_{j,(k \bmod n_1)}^{(2)}.$$

Since $s_{(i,ceil(k/n_1))}^{(1)}$ is independent of j,

$$\begin{aligned} d_k &= \sum_{i=1}^{n_2} s_{i,ceil(k/n_1)}^{(1)} \sum_{j=1}^{n_2} s_{(j,(k \bmod n_1))}^{(2)} \\ &= \sum_{i=1}^{n_2} s_{i,ceil(k/n_1)}^{(1)} d_{(k \bmod n_1)}^{(2)} \\ &= d_{ceil(k/n_1)}^{(1)} d_{(k \bmod n_1)}^{(2)}. \end{aligned}$$

This is the same as the the Kronecker product of diagonals where k maps to the ordering resultant of Kronecker products. □

6.1.4 Experiments

The experiments begin with synthetic evaluation of the proposed methodology. It is followed by evaluation of mean squared error in terms of predicting the number of patients admitted to hospitals in California for a given disease each month. Then synthetic time trials and a time trial of the nested network on admissions to each hospital in California.

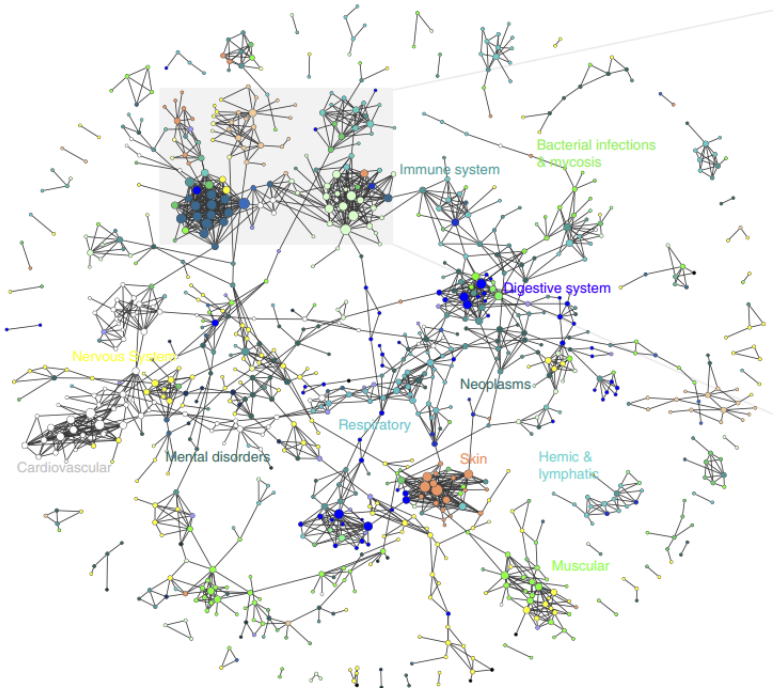

FIGURE 6.5: Disease similarity network generated from PubMed document co-occurrence (Zhou et al. (2014))

Synthetic Experiments

We started the evaluation with a myriad of synthetic datasets. First, we examine a case where we expect UmGCRF and GCRF to perform similarly ($\alpha, \beta > 0$). Next, we look at cases where $\alpha > 0$ & $\beta < 0$ and then $\alpha_1 > 0$ & $\alpha_2 < 0$. The last synthetic experiment is a series of trials used to compare GCRF and UmGCRF performance depending on the

number of negative link weights in the network. We finish this subsection demonstrating performance on the task of predicting the number of people admitted to hospitals for a given disease in California each month.

In the following three experiments, we generated a vector of length 2000 which is then used as an unstructured prediction. We generated a positive valued matrix which represents a uni-model graph with 2000 nodes and 2 million edges. Then, choosing appropriate values for α and β we generated our target vector using the GCRF model. Models were compared in terms of conditional log-likelihood (LL) and R^2.

The synthetic data for this experiment was generated with parameters $\hat{\alpha} = 0.792$ and $\hat{\beta} = 0.61$. The results of regression on such a graph are shown in Table 6.1. In this experiment UmGCRF and GCRF models were nearly identical and almost perfect accuracy.

Model	LL	α	β	R^2
GCRF	-97.346	0.791	0.611	0.999
UmGCRF	-97.350	0.793	0.609	0.999

Table 6.1: Convergence evaluation of standard and proposed GCRF optimizations on standard synthetic data

Comparison of the optimal parameters and loss for the standard GCRF optimization and the proposed GCRF optimization

The next synthetic dataset was generated with parameters $\hat{\alpha} = 1$ and $\hat{\beta} = \frac{-1}{2}$. The results of regression on that graph by UmGCRF vs. GCRF shown in Table 6.2 provide evidence that UmGCRF significantly outperforms GCRF.

Model	LL	α	β	R^2
GCRF	-5.83e+05	0.820	0.573	0.31
UmGCRF	-1.11e+03	0.894	-0.447	**0.56**

Table 6.2: Convergence evaluation of standard and proposed GCRF optimizations on synthetic data where predictions ought to be pushed apart

Comparison of the optimal parameters and loss for the standard GCRF optimization and the proposed GCRF optimization (negative adjacency matrix is the ground truth for synthetic data)

Here, the synthetic data was generated using two unstructured predictors with the optimal parameters $\hat{\alpha_1} = \frac{\sqrt{3}}{2}$ and $\hat{\alpha_2} = \frac{-1}{2}$. The results are shown in table 6.3. UmGCRF significantly outperformed GCRF and captured the optimal set of parameters. In contrast, GCRF could not find the optimal negative parameter α_2 and instead it found a positive value (which is sub-optimal) that affects its accuracy as indicated by a very low R^2.

Model	LL	α_1	α_2	R^2
GCRF	-54.21e+03	0.721	0.693	0.051
UmGCRF	-01.37e+03	0.866	-0.500	**0.78**

Table 6.3: Synthetic experiment where the optimal combination of GCRF parameters is non-convex

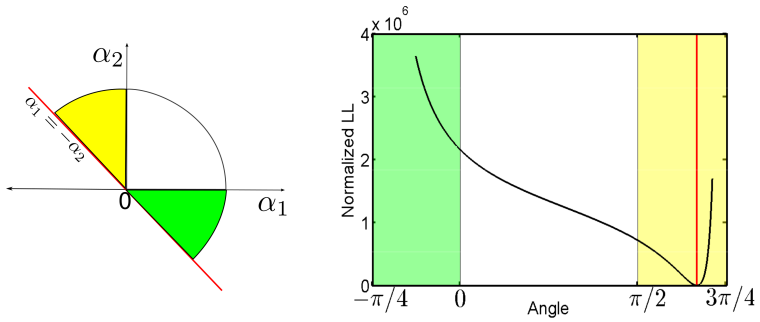

FIGURE 6.6: Extended Parameter Space and Log-Likelihood

That scenario is illustrated at Figure 6.6 where we graphically show why UmGCRF outperformed GCRF on this experiment. In the left panel, the expanded search for α_1 and α_2 is projected onto the unit (half) circle. In the right panel, we plot the angle between the two parameters (x-axis) and the corresponding normalized negative log-likelihood NLL (y-axis). The white region at the right panel corresponds to the first quadrant at the left panel where both parameters α_1 and α_2 are positive and this is the space where GCRF looks for the optimal parameter. Clearly GCRF can not find the optimal parameter (red

vertical line at the yellow region in the right panel) because they are out of its parameter space, while UmGCRF searches for the optimal parameters in all 3 regions (green, yellow, and white) which can be found. The ability to map the log-likehood on a 2-d plot with parameters required a corollary which can be found at the end of the chapter. That corollary also shows why αs were originally only weighted average coefficients although they were reported as any positive linear combination.

The reason that UmGCRF significantly outperformed GCRF in Table 6.2 is that the links (similarity) have negative influence on the predicted value which cannot be captured by GCRF, but were captured by UmGCRF model. In real-life applications, it is not necessary the case that all links have either positive (Table 6.1) or negative (Table 6.2) influence on the prediction. In many cases (as depicted in many real datasets including the one described in the next section) some of the links might have positive influence and some other links might have negative influence. For example, node i might be positively correlated to node j while it has negative correlation with node k. In this case the β parameter will be positive such that it captures both.

Note that, this case can not be captured by the original GCRF. To handle this case usually all negative correlations are removed from the graph before applying GCRF since it can not handle this case otherwise.

To evaluate benefit of UmGCRF in this scenario we generated 7 synthetic graphs with $0\%, 16\%, 35\%, 50\%, 65\%, 84\%$, and 100% of positive inks, respectively. Here, the graph that corresponds to 0% is basically the experiment shown in Table 6.2 while 100% corresponds to the experiments summarized in Table 6.1. The regression results by UmGCRF on these 7 datasets for these experiments are shown in Figure 6.7.

The x-axis in Figure 4 represents the percentage of positive links the graph while the y-axis represents the ratio of Mean Square Error (MSE) of UmGCRF to GCRF. So, if UmGCRF significantly outperforms GCRF then the ratio tends to zero, while the ratio tends to one when both models have equally performance.

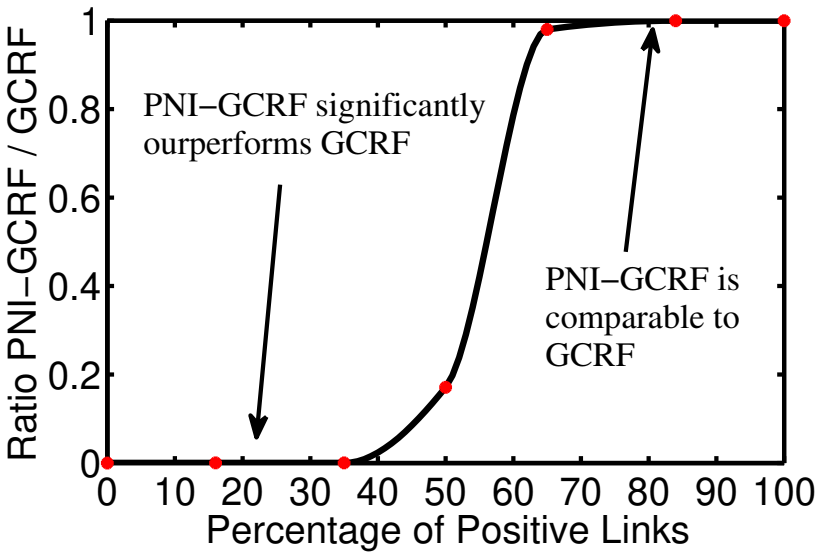

FIGURE 6.7: Ratio of MSE of UmGCRF to GCRF for different datasets with different percentages of positive links

Hospital Admissions

Next, is the evaluation of UmGCRF on the problem of predicting monthly hospital admissions for 189 classes of diseases in California from HCUP data HCUP (2011). The target is to predict the number of admission for each disease for the next month. For each of 35,844,800 inpatient discharge records collected over 9 years in the California HCUP database there are up to 253 diagnosis codes in CCS coding schema. In this study, we constructed monthly disease graphs such that each node represents one disease. But 22 diseases had incomplete information over the time period analyzed, resulting in 231 nodes. In this experiment we use the disease-symptom similarity network built in Zhou et al. (2014). Since this network was build using MeSH terminology, we built a translation table by hand for CCS codes to the specific MeSH terminology used in Zhou et al. (2014). That table is publicly available at http://astro.temple.edu/∼tud25892. The matching is not one-to-one. Sometimes we would map several MeSH terms to a single CCS code. In these cases, we would take the average of similarities. There were also 52 CCS codes with no MeSH term in the network used. This reduced the number diseases in our analysis to 189.

Assume that $x_{t,i}$ is the number of patients diagnosed with disease i during the month t. For each node i, we computed the rate of admission change $y_{t,i} = (x_{t,i} - x_{t-1,i})/x_{t-1,i}$. We used 108 monthly graphs (representing years 2003-2011). After converting this to a rate of change, we have 107 time points. We train on the first 80 months and test on the remaining 27. We slide a window of length $w = 12$. We trained two unstructured predictors: Linear regression (LR) and a Neural Network (NN). They were used as input for both GCRF and UmGCRF. NN had 26 hidden nodes. The algorithm was tested 100 times because the NN is non-convex and yields different results each time. That has an effect of altering the output of GCRF and UmGCRF. Averages are taken to report results.

Algorithm	Training MSE	Testing MSE	Time
SpGCRF	0.0002	0.0577	41 min
LR	0.0253	0.0268	1 sec
NN	[0.0204 to 0.0209]	[0.0257 to 0.0309]	25 sec
GCRF	0.0155	0.0229	3 min
UmGCRF	0.0148	**0.0228**	30 sec

Table 6.4: GCRF regression performance on Hospital admissions data set

Although the improvement in accuracy is marginal, UmGCRF outperforms GCRF, achieving a lower testing error for greater than 80% of trials in an experiment of a thousand trials. This increase in accuracy is due to an improved network model. The network input was disease similarity on a zero to one scale. Upon inspection of parameters, one can see that our learned intercept term tended to be around -.05. That shifts the similarity scale from $[0, 1]$ to $[-0.05, 0.95]$. Of the 100 trials used for the table above, UmGCRF outperformed GCRF 89 times on training data and 71 times on testing Data. UmGCRF also outperformed NN on training Data and on testing Data 95 times as seen in Figure 5. UmGCRF had 17% and 12% improvement in test accuracy over input baselines.

Although the SpGCRF method can be regularized in a way where it does not learn structure, in that case it still performs worse than linear regression. This is most likely due to the fact that SpGCRF ignores at what nodes the feature is observed.

Time Trials

For a comparison, four implementations are examined. SpGCRF was chosen to represent network learning algorithms because they made code available for testing. We also compare the time for the original proposed method, GCRF, and a fast learning approximation method, FF-GCRF Ristovski et al. (2013). The following speed tests were done in Matlab with a single feature per target variable. SpGCRF has widely inconsistent convergence times depending on the data.

Target Size	UmGCRF	FF-GCRF	GCRF	SpGCRF
1,000	4.7 secs	3.3 secs	41 secs	[13 to 200] mins
5,000	43 secs	34 secs	9.2 mins	[28 to 42] hours
10,000	3.9 mins	2.9 mins	1.76 hours	9+ days
20,000	30.2 mins	22.4 mins	17 hours	N/A
40,000	5 hours	3 hours	7.5 days	N/A
100,000	21 hours	16 hours	N/A	N/A

Table 6.5: Speed of different GCRF optimizations

For the first set of experiments, time trials were conducted where the proposed GCRF-MSN was compared to three structured regression alternatives discussed in section 2 (GCRF, UmGCRF, and FF-GCRF). Then, a real world data experiment is run and evaluated in terms of execution time and mean squared error (MSE). The task is to predict monthly admissions for each disease for each hospital in the state of California. The data comes from the California HCUP database, and contains 35,844,800 inpatient discharge records collected over 9 years, and uses the CCS disease coding schema.

Start by comparing MSN-GCRF to previous fast implementations of GCRF as shown at Table 2. The relative speed of UmGCRF, FF-GCRF, and GCRF was investigated in Glass et al. (2016), but that analysis was done on a single static graph over two time steps. In the HCUP dataset, the network changes over time. The speed performance examined below is for optimizing evolving graphs with 100,000 nodes at each time step across 72 time points. Critically, this network can be segmented into two networks on different

scales.

GCRF-MSN	UmGCRF	FF-GCRF	GCRF
10 minutes	1.1 weeks	1 week	2 months

Table 6.6: Run time of proposed GCRF-MSN and other methods on Big Data

Any datasets where structured regression is applicable could alternatively use independent functions for each node. In our motivating data set, this corresponds to giving each disease within each hospital its own time series function. In the table below, we label this scenario "unique". If we use the following variables: h is the number of hospitals, d is the number of diseases, and t is the number of time steps; then we can see that $n = h \cdot d \cdot t$. In the "unique" scenario, we have $h \cdot d$ equations and t data points per equation. The other possible scenarios are detailed in the table below. Possible data segmentations for predicting monthly admissions for each disease at each hospital.

These various data segmentations are useful for understanding the differences between models. VARIMA and GCRF-MSN use a weighted mixture of Unique and All. A VARIMA model would be an ARIMA model in the "All" scenario with a network where every weight was equal to one. Because GCRF inputs unstructured predictions, we train regressions on all the above possible segmentations and then input all of those predictions into GCRF. This is interesting because GCRF can balance the gains of each. Some of the algorithms are very poor performers, but collectively they can produce a much more accurate prediction.

The structure used for VAR and GCRF-MSN boils down to N^2 pairwise relations. Each relation is measured by a weight. These weights are input to both methods as prior information. For the disease similarity network, we use symptom similarity scores

Identifier	Parameter Sets	Data Points
All	1	h $\cdot d \cdot t$
Disease	d	$h \cdot t$
Hospital	h	$d \cdot t$
Unique	$h \cdot d$	t

from Zhou et al. (2014) where biomedical literature was parsed, and binary indicators for correlated symptoms were generated. The relationship weight is calculated by taking the cosine similarities of each symptom vector. This was the approach used in Glass et al. (2016).

The other network prior used was a set of weights based on hospital similarity. For this purpose, a hospital similarity network built in a previous study Polychronopoulou and Obradovic (2014) to capture the overlap in specialization that each hospital has was incorporated. The specialization of each hospital is represented by the distinct rate at which hospitals treated various diagnoses in the previous year. For instance, Cardiovascular Hospitals will only treat a subset of diseases and so will have a large similarity value with one another.

GCRF-MSN has a subquadratic preprocessing step, with linear learning time and inference. UmGCRF has a cubic preprocessing step which takes nearly a day per network, with nine networks, that means over a week. GCRF has cubic complexity at every iteration of gradient descent and takes months to complete learning on a dataset this size. We used the four autoregressive setups: "All", "Disease", "Hospital", "Unique" as outline above for the input learners for GCRF-MSN because those were all computable in around a minute or less. We can see execution time and mean squared error for the HCUP data set for the 3 relevant algorithms.

Table 6.7 presents run time, training and testing error (MSE) of GCRF-MSN, neural networks (NN), and the varima model (VAR(12)). Using a two-sided difference in means, there is a p-value < 0.001 that these differences in error come from similar distributions.

Model	Time	Train MSE	Test MSE (Standard Error)
NN	6.7 hours	0.0214	0.0574 (3.71e-4)
VAR(12)	166 hours	0.0189	0.0548 (3.84e-4)
GCRF-MSN	**10 min**	**0.0178**	**0.0531** (3.08e-4)

Table 6.7: GCRF-MSN Performance on Large Scale Hospital Admissions

GCRF-MSN used a combination of weak learners to outperform the next two most accurate regression methods. This is important because those weak learners were not as accurate but they were much more efficient to compute. VAR utilizes a network structure and learns a unique prediction function for each disease at each hospital, but this prediction function can closely be approximated when only using neighbors prediction functions. The VAR model has particularly reasonable assumptions and nice imputation of missing functions if the network is well built. The time to compute VAR demonstrates the scaling difficulty of VAR as opposed to GCRF-MSN in this scenario. VAR is traditionally one of the most efficient structured approaches taking $O(IN^2)$ time. However, in the case where the structure can be represented as a Kronecker product, GCRF-MSN runs much faster.

6.2 Local Regression

Integrating an output kernel is a convex optimization. It is a transformation of output predictions. It can be optimized as many independent weighted least squares problems. The W below is referred to as the output kernel. In GCRF, the covariance matrix acts an an output kernel.

This optimization is a smoothed output of many local models. When W is row normalized, the mean squared error of the above model can be optimized with a weighted least squares optimization for each local function. The wi vector is the ith row of the W matrix. The Network Lasso has does not combine output models on training data. But uses a graph Laplacian to pull the weights of local models toward each other. Incorporating the strength of the network connections is difficult to integrate can keep convexity. The Network Lasso shifts the non-convex component to be a hyperparameter.

A similarity matrix, S, can use the same similarity matrix as GCRF. Rather than a Laplacian transformation, the algorithm proposed here uses row normalization. When $\gamma = 0$ the algorithm produces the same prediction as linear regression. When $\gamma = 1$ the algorithm produces the same results as locally weighted least squares. S and J are row normalized so their convex combination is also. The novelty comes from the fact that γ is also optimized at each step.

The interpolation comes from the row normalized value W. This methodology does not have the costly gradients of GCRF, it can directly optimize mean squared error, and the local models are learned simultaneously with the graph weights (GCRF learns predictions without a network structure and then combines them together with a graph structure).

The aim of the regression models researched in this dissertation is to minimize mean squared error. In order to integrate a Laplacian output kernel and learn the output kernels influence, GCRF does not minimize mean squared error exactly – the convex loss is altered.

$$Y = (W\Theta \odot X)\mathbb{1} \tag{6.39}$$

However, there are other ways to integrate network information a prediction function. The network lasso uses local models and regularizes the distance between the weights of linear functions. Locally weighted least squares can be interpreted as an output kernel. Traditionally, Lowess uses input features in order to build the similarity, however, the output kernel can come from external data.

6.2.1 Network Lasso

Network Lasso is an extension of traditional regularization techniques that includes regularization across a network of similarities. It refers to all regularization approaches as a Lasso. Regularization is a common approach to lowering the bias of a method. It allows the number of features to exceed the number of examples. It includes a hyperparameter, λ. One of the most common regularization approaches is the ridge regularization. This approach minimizes the sum of squared values of the inputs.

This term introduced in D. Hallac and Boyd. (2015) covers a broad framework of regularization across localized predictions. In this model, the similarities across nodes are incorporated via regularization weights. What is interesting is that this model interpolates for testing on new data in a way that is a naive implementation of our approach. Given local loss, l and edge weight $e_{j,k}$ and penalty $g(x_j, x_k)$.

$$\mathcal{L} = C\sum_{i \in V} l_i(x_i) + \sum_{j,k \in E} e_{j,k} \cdot g(x_j, x_k) \tag{6.40}$$

Network Ridge Regression

Here is an instance of the network lasso for regression with ridge regularization. (D. Hallac and Boyd. (2015))

$$l_i(x_i; \theta_i) = y_i - \theta_i x_i^T + \mu||\theta_i||_2^2. \tag{6.41}$$

The graph structure is a ridge network cost function.

$$g(x_j, x_k) = ||\theta_j - \theta_k||_2^2. \tag{6.42}$$

The weighted L2 norm of the matrix can be efficiently represented by the Laplacian of the weight matrix L_W.

$$\sum_{j,k \in E} w_{j,k} \cdot g(x_j, x_k) = \sum_{j,k \in E} w_{j,k} ||\theta_j - \theta_k||_2^2 = Tr(\Theta L_W \Theta). \tag{6.43}$$

The final loss function takes the form (equation 6.44).

$$\mathcal{L} = Cr^T r + ||\Theta||_2^2 + Tr(\Theta L_W \Theta). \tag{6.44}$$

6.2.2 Differentiable Output Kernel

Similar to Network Lasso, we also the the ideal property where if $\lambda = 0$, the method produces the parameters from linear regression. Somehow we have managed to get some of the best properties from GCRF and Network Lasso and come up with a method with an objective function that minimizes MSE and can be learned faster than either of those two approaches. Adding the following component to the Lowess approach sets it on equal footing with the Network Lasso and GCRF.

$$W = (\gamma S + (1 - \gamma)\frac{1}{N} J) \tag{6.45}$$

S is a row normalized similarity and takes the place of what was being represented as W in the earlier sections. λ is now a parameter that reflects the quality of the information in the graph. The prediction for each output is given by the matrix of parameter weights Θ and the smoothing kernel W.

$$f(\Theta, w_i, x_i) = w_i \Theta x_i^T. \tag{6.46}$$

This is akin to β in GCRF and λ in Network Lasso. In our case, as in GCRF, we can optimize this parameter as part of the convex optimization function. This is much more

convenient then optimizing with a hyperparameter. The update rule for λ is given below. The prediction for every node is

$$\partial l/\partial \lambda = \sum_{i}^{n} e_i(s_i - \frac{\vec{1}^T}{N})\Theta x_i^T. \tag{6.47}$$

Although the optimization is not guaranteed to be convex, the experimental results are promising.

6.2.3 Nested Network Optimization

A Kronecker product is a 2-D expression of the outer product of 2-D matrices, which in a more natural way can be represented as a tensor ($W = H \otimes D$). The interpolation of the differentiable output kernel approach allows us to not calculate the Kronecker product of weights because the parameter set can be left in tensor form. Θ is shaped $[N_1$ by N_2 by $P]$. This is based on two similarities, represented by matrices H and D. H is a matrix of hospitals $[N_1$ by $N_1]$ and D is a matrix of diseases $[N_2$ by $N_2]$. We use different letters H & D for different similarities W_1 & W_2 to avoid overusing subscripts as much as possible. While before we had a parameter matrix, now we will have a parameter tensor, Θ, shaped $[h, d, f]$. The prediction in this setting can be formulated as in equation 6.48.

$$f(\Theta, h_i, d_i, x_i) = (h_i \Theta d_i^T) x_i^T \tag{6.48}$$

6.2.4 Experiments

The experiments are a Sacramento real estate price prediction task and comparisons to the two experiments run for analysis of the GCRF algorithm.

Sacremento Real Estate Value

Sacremento Real Estate value is an openly available dataset. It was the dataset used in the original Network Lasso paper. We recreate their exact experimental conditions and demonstrate a 22% improvement over their model. In the application presented

for network lasso (D. Hallac and Boyd. (2015)), a dense network is calculated via the inverse of the distances between all houses. That network is then pruned and forced to be symmetric. This process is a time consuming one. That preprocessing step will be referred to as a sparse graph structure.

Although the GCRF and network lasso require graph symmetry, the new model does not have this requirement and benefits from asymmetry. Of 10,000 random partitions of the training and testing. The differentiable output kernel outperformed Network Ridge, 8,691 times. or 86.91% of testing sets.

Method	Dense Graph	Sparse Graph
Linear Regression	0.7014	0.7014
Lowess	0.5342	0.5342
Network Ridge	0.7010	0.4935
Differentiable Output Kernel	0.4331	0.4360

Table 6.8: Testing MSE comparing dense and sparse networks.

All of the algorithms presented here perform worse with a dense network. But, this network is not artificially pruned. Pruning a network prohibits learning distances because if the ordinal distances of the observations change when learning a distance, then a pruned network can exhibit discontinuous changes and many more procedures will be required to re-evaluate the matrix entry ordinalities at every step. Also, testing the optimal K requires many separate evaluations. This model was important because it established a baseline for interpolation on testing data. This is essential for cold start predictions which many methods cannot handle well or efficiently. Furthermore, both network lasso and the new method can provide parameters defined over a space or network. This produces an interpretable result.

Monthly Disease Admissions

We revisit the experiments predicting monthly hospital admissions by disease for the state of California as done in Glass et al. (2016). The new model shows a 14% improvement

over GCRF.

Method	Train MSE	Test MSE	Time
SpGCRF	2.5e-4	0.0577	41 min
NN	[0.0204 to 0.0209]	[0.0257 to 0.0309]	25 sec
GCRF	0.0155	0.0229	3 min
Differentiable Output Kernel	0.0236	**0.0210**	5 sec

Table 6.9: Predicting the total number of patients admitted to any hospital in California for each major disease

Now revisit the experiments predicting monthly disease admissions for each hospital in the state of California as done in Glass and Obradovic (2017). This is potentially the most astonishing results yet, finding a 78% improvement over GCRF and a learning time complexity similar to linear regression. It is however, more memory intensive than linear regression – the same burden as GCRF.

Method	Train MSE	Test MSE	Time
NN	0.0214	0.0574	6.7 hrs
MSN-GCRF	0.0178	0.0531	10 min
Differentiable Output Kernel	**0.0008**	**0.0148**	**2.5 min**

Table 6.10: Predicting the number of patients admitted for each disease to each hospital in California

The Kronecker product of two row normalized matrices is also a row normalized matrix. As in the case of GCRF, the Kronecker product does not need to be calculated even though it impacts the model. The Kronecker product is implicitly in the equations and is used for theoretical understanding of expanding the models to more complex graph structures. Below, Θ is a tensor and the two different similarities are referenced by H and D respectively.

The GCRF experiments were revisited and compared to the new algorithm. The proposed algorithm achieves much better mean squared error in much less time. As currently implemented, the optimization is non-convex with no convergence guarantees. It can, however, be cast as a Frank-Wolfe optimization of the L2 regularization of the

parameters with the additional constraint that $\gamma \in [0, 1]$. The method then has convergence guarantees.

γ increases if the local regression is more accurate than the linear regression. Because this is always the case lambda is only increasing. Future work in this area would benefit from taking the convex combination of a sigmoid transformation of the ratio of the sum of squared errors of the weighted local regression and the linear regression. If the gradient updates are isolated per similarity matrix, the J matrix only needs to calculate and store the weight updates for the standard linear matrix. A boosted version would use residual weighted regression trees for each step of of gradient descent. For each local model the residual is weighted by the contribution of that example to that local model.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

Although it is possible to optimize ranking over pairwise examples, it is often more logical to use binary classification because it is much faster and typically performs nearly as well. In this dissertation, the relationship between univariate and pairwise losses are exploited at each step of Frank-Wolfe dual coordinate descent in order to produce unparalleled AUC in top tier run time. At a more basic level the Frank-Wolfe dual coordinate descent is examined in terms of how it can improve the runtime and scalability of binary and multiclass SVM and in order to improve the pairwise hinge RBF kernel scalability.

The optimization of Gaussian conditional random fields is improved. Large speed ups for the case of a single graph and larger memory and speed up savings for special nested graph structures. A simpler yet novel strategy which achieves the same goal is introduced. It runs faster and achieves lower mean squared error. Although it is differentiable, it is not convex. So future work would be to show convergence or to alter the method to achieve convergence. There method can be incorporated into graph neural networks. More interesting gains may be achieved by using decision tree stumps for each step of the optimization producing a network gradient boosting regression.

In this dissertation, multivariate output regression efficiency for Gaussian conditional

random fields is improved and a novel algorithm is presented. The Frank-Wolfe dual coordinate descent algorithm is reviewed and shown to easily integrate GPUs for speed ups. The theoretical properities of univariate and pairwise losses are exploited so that classification algorithms can be used at each iteration of the Frank-Wolfe algorithm. This improves performance for linear and kernel functions. The univariate upper bound of pairwise iteration loss enables a gradient boost extension. The gradient boost extension achieves state of the art ranking metrics without greatly increasing time over classification algorithms. A speed up and memory reduction strategy would be to only use several representative data points or pseudo generated points which fit a local model.

BIBLIOGRAPHY

A. Athanasopoulos, A. Dimou, V. M. I. K. (2011), "GPU Acceleration for Support Vector Machines," in *Proc. 12th International Workshop on Image Analysis for Multimedia Interactive Services*, WIAMIS 2011.

Agarwal, S. and Narasimhan, H. (2013), "A structural svm based approach for optimizing partial auc," in *Proceedings of the 30th International Conference on Machine Learning*, ICML '13, pp. 516–524, New York, NY, USA, ACM.

Ayres, F. (1967), "Theory and Problems of Matrices," in *Theory and Problems of Matrices*.

Bartlett, P. L., Jordan, M. I., and McAuliffe, J. D. (2006), "Convexity, classification, and risk bounds." in *Journal of the American Statistical Association*, pp. 138–156.

Chang, C.-C. and Lin, C.-J. (2011), "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, 2, 27:1–27:27, Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chen, K., Li, R., Dou, Y., Liang, Z., and Lv, Q. (2017), "Ranking support vector machine with kernel approximation." in *Computational intelligence and neuroscience*.

Chen, T. and Guestrin, C. (2016), "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, ACM.

Cortes, C. and Mohri, M. (2003), "AUC optimization vs. error rate minimization," in *Proceedings of the 16th International Conference on Neural Information Processing Systems*.

Crammer, K. and Singer, Y. (2001), "On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines," *Journal of Machine Learning Research*.

D. Hallac, J. L. and Boyd., S. (2015), "Network Lasso: Clustering and Optimization in Large Graphs," in *Proceedings SIGKDD*, SIGKDD, pp. 387–396.

Dheeru, D. and Karra Taniskidou, E. (2017), "UCI Machine Learning Repository," .

Ding, J. and Zhou, A. (2007), "Eigenvalues of rank-one updated matrices with some applications," in *Applied Mathematics Letters*.

Djuric, N., Radosavljevic, V., Obradovic, Z., and Vucetic, S. (2015), "Gaussian Conditional Random Fields for Aggregation of Operational Aerosol Retrievals," *IEEE Geoscience and Remote Sensing Letters*.

Doğan, Ü., Glasmachers, T., and Igel, C. (2016), "A Unified View on Multi-class Support Vector Classification," *Journal of Machine Learning Research*.

Dressel, J. and Farid, H. (2018), "The accuracy, fairness, and limits of predicting recidivism," *Science Advances*, 4.

Fan, R.-E., Chen, P.-H., and Lin, C.-J. (2005), "Working Set Selection Using Second Order Information for Training Support Vector Machines," *J. Mach. Learn. Res.*, 6, 1889–1918.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008), "LIBLINEAR: A Library for Large Linear Classification," *Journal of Machine Learning Research*, 9, 1871–1874.

Friedman, J., Hastie, T., and Tibshirani, R. (2008), "Sparse inverse covariance estimation with the graphical lasso," in *Biostatistics*.

Gao, W. (2013), "One-Pass AUC Optimization." in *Proceedings of the 30th International Conference on Machine Learning*, PMLR 28.

Gao, W. and Zhou, Z.-H. (2015), "On the consistency of auc pairwise optimization." in *In Twenty-Fourth International Joint Conference on Artificial Intelligence*, IJCAI '15.

Glass, J. and Obradovic, Z. (2017), "Structured Regression on Multi-Scale Networks," in *IEEE Big Data Special Issue*, pp. 23–30.

Glass, J., Ghalwash, M., Vukicevic, M., and Obradovic, Z. (2016), "Extending the Modeling Capacity of Gaussian Conditional Random Fields while Learning Faster," in *Proc. Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*.

Gligorijevic, D., Stojanovic, J., and Obradovic, Z. (2015), "Improving Confidence while Predicting Trends in Temporal Disease Networks," in *4th Workshop on Data Mining for Medicine and Healthcare, SIAM International Conference on Data Mining (SDM)*.

Guo, H. (2013), "Modeling short-term energy load with continuous conditional random fields," in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pp. 433–448.

HCUP (2011), "HCUP State Inpatient Databases (SID). Healthcare Cost and Utilization Project (HCUP). 2005-2009. Agency for Healthcare Research and Quality, Rockville, MD." http://www.hcup-us.ahrq.gov//sidoverview.jsp.

Hsieh, C. J., Chang, K. W., Lin, C. J., Keerthi, S. S., and Sundararajan, S. (2008), "A dual coordinate descent method for large-scale linear SVM," in *Proceedings of the 25th international conference on Machine learning*, ICML '08, p. 408.

Joachims, T. (2005), "A support vector method for multivariate performance measures," in *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pp. 377–384, New York, NY, USA, ACM.

Joachims, T. (2006), "Training linear svms in linear time," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pp. 217–226, New York, NY, USA, ACM.

Joachims, T., Finley, T., and John Yu, C.-N. (2009), "Cutting-plane training of structural SVMs," in *Machine Learning*, vol. 77, pp. 27–59.

Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., and Murthy, K. R. K. (2001), "Improvements to Platt's SMO Algorithm for SVM Classifier Design," *Neural Comput.*, 13, 637–649.

Khorram, S., Bahmaninezhad, F., and Sameti, H. (2014), "Speech Synthesis Based on Gaussian Conditional Random Fields," in *Artificial Intelligence and Signal Processing*, pp. 183–193.

Kotlowski, W., Dembczynski, K. J., and Huellermeier, E. (2011), "Bipartite ranking through minimization of univariate loss," in *Proceedings of the 28th International Conference on Machine Learning*, ICML '11, pp. 1113–1120, New York, NY, USA, ACM.

Krasser, M. (2018).

Kuo, T.-M., Lee, C.-P., and Lin, C.-J. (2014), "Large-scale kernel ranksvm." in *Proceedings of the 2014 SIAM international conference on data mining*, SIAM '14, pp. 812–820.

Lacoste-Julien, S., Jaggi, M., Schmidt, M., and Pletscher, P. (2013), "Block-Coordinate Frank-Wolfe Optimization for Structural SVMs," in *Proceedings of the 30th International Conference on Machine Learning*, PMLR 28.

Larson, J., Mattu, S., Kirchner, L., and Angwi, J. (2016), "How We Analyzed the COMPAS Recidivism Algorithm," Propublica.

LEMUR (2019), "The LEMUR project," .

Li, N., Jin, R., and Zhou, Z.-H. (2014), "Top rank optimization in linear time." in *Advancements in Neural Information Processing*, NIPS '14, pp. 1502–1510.

Liu, M. (2018), "Fast Stochastic AUC Maximization with O(1/n) Convergence Rate." in *Proceedings of the 35th International Conference on Machine Learning*, PMLR 33.

Long, P. M. (2008), "Boosting the Area Under the ROC Curve." in *Advances in neural information processing systems*, NIPS '08.

Lovasz, L. (2004), "Discrete Analytic Functions: a survey, in: Eigenvalues of Laplacians and other geometric operators," in *International Press, Surveys in Differential Geometry IX*.

Merris, R. (1998), "Laplacian graph eigenvectors," in *Linear Algebra and its Applications*.

Narasimhan, H. (2013), "SVMtight pAUC: A New Support Vector Method for Optimizing Partial AUC Based on a Tight Convex Upper Bound," in *Proceedings of the SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13.

Osuna, E., Freund, R., and Girosi, F. (1997), "Support vector machines: Training and applications, Tech. Rep. AIM-1602, MIT." Articial Intelligence Laboratory.

Pedregosa, F., Askari, A., Negiar, G., and Jaggi, M. (2018), "Step-Size Adaptivity in Projection-Free Optimization," in *arXiv*.

Platt, J. (1998), "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," MSR-TR-98-14.

Polychronopoulou, A. and Obradovic, Z. (2014), "Hospital pricing estimation by Gaussian conditional random fields based regression on graphs," in *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 564–567.

Radosavljevic, V., Vucetic, S., and Obradovic, Z. (2010), "Continuous Conditional Random Fields for Regression in Remote Sensing." in *Proccedings of European Conference on Artificial Intelligence (ECAI)*, pp. 809–814.

Radosavljevic, V., Vucetic, S., and Obradovic, Z. (2014), "Neural Gaussian Conditional Random Fields," in *Proceeding of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*.

Ristovski, K., Radosavljevic, V., Vucetic, S., and Obradovic, Z. (2013), "Continuous Conditional Random Fields for Efficient Regression in Large Fully Connected Graphs," in *AAAI*.

Schäcke, K. (2013), "On the Kronecker Product," .

Snoek, J. (2012), "Practical Bayesian Optimization of Machine Learning Algorithms," in *Advances in neural information processing systems*, NIPS '12.

Tappen, M. F., Liu, C., Adelson, E. H., and Freeman, W. T. (2007), "Learning gaussian conditional random fields for low-level vision," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8.

Taskar, B., Guestrin, C., and Koller, D. (2004), "Max-margin markov networks." Advances in neural information processing systems, pp. 25–32.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005), "Large margin methods for structured and interdependent output variables," Journal of machine learning research, pp. 1453–1484.

Wang, S., Zhang, L., and Urtasun, R. (2014), "Transductive Gaussian processes for image denoising," in *Computational Photography (ICCP), 2014 IEEE International Conference on*, pp. 1–8, IEEE.

Westin, J. and Watkins (1999), "Support Vector Machines for Multi-Class Pattern Recognition," in *European Symposium on Artificial Neural Networks*, ESANN.

Wytock, M. and Kolter, Z. (2012), "Sparse Gaussian conditional random fields," in *NIPS workshop on log-linear models*.

Wytock, M. and Kolter, Z. (2013), "Sparse Gaussian conditional random fields: Algorithms, theory, and application to energy forecasting," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pp. 1265–1273.

Ying, Y., Wen, L., and Lyu, S. (2016), "Stochastic Online AUC Maximization." in *Advancements in Neural Information Processing*, NIPS '16.

Yue, Y., Finley, T. W., Radlinski, F., and Joachims, T. (2007), "A Support Vector Method for Optimizing Average Precision," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, p. 271–278, ACM.

Zhou, X. Z., Menche, J., Barabasi, A.-L., and Sharma, A. (2014), "Human symptoms-disease network," in *Nature Communications*.

Zhou, Z.-H. and Liu, X.-Y. (2010), "On multi-class cost-sensitive learning." in *Computational Intelligence*, pp. 232–257.

Zumstein, P. (2005), "Comparison of Spectral Methods Through the Adjacency Matrix and Laplacian of a Graph," .