Tutorial on SVM and Multiple Kernel Learning (MKL)

Ashesh Jain
IIT Delhi

August 25, 2011

Supervisors:
- Dr. Manik Varma (Microsoft Research India)
- Dr. SVN Vishwanathan (Purdue University)
- Dr. KK Biswas (CSE - IITD)
Binary Classification

There are two classes:

1. Class 1: \( X = \{X_1, X_2, \ldots, X_n\} \)
2. Class -1: \( Y = \{Y_1, Y_2, \ldots, Y_m\} \)
3. Data points with known labels constitute the training set
4. A "Classifier" is trained on training set and then tested on unknown(test) set
   \( Z = \{Z_1, Z_2, Z_3, \ldots, Z_k\} \)

A classification problem can also be multiclass but here we are only concerned with "Binary Classification Problems"
Binary Classification

There are two classes:

1. Class 1: \( X = \{X_1, X_2, \ldots, X_n\} \)
2. Class -1: \( Y = \{Y_1, Y_2, \ldots, Y_m\} \)
3. Data points with known labels constitute the training set
4. A ”Classifier” is trained on training set and then tested on unknown(test) set \( Z = \{Z_1, Z_2, Z_3, \ldots, Z_k\} \)

A classification problem can also be multiclass but here we are only concerned with ”Binary Classification Problems”

Different algorithms for Classification

1. Multi-Layer Perceptron (Neural Networks)
2. Support Vector Machines
3. k-nearest neighbours
4. Decision Tree
5. Boosting
Decision Function

$$\mathcal{F}(x) = \langle w, x \rangle + b$$

- Task 1: Learn $w$ and $b$ over known examples
- Task 2: Calculate $\mathcal{F}(x)$ for unknown examples and classify them using $\text{sign}(\mathcal{F}(x))$
Decision Function

\[ F(x) = \langle w, x \rangle + b \]

- Task 1: Learn \( w \) and \( b \) over known examples
- Task 2: Calculate \( F(x) \) for unknown examples and classify them using \( \text{sign}(F(x)) \)

Optimization Problem

\[ \min_w CL(\phi w; y) + R(w) \]
Linear Support Vector Machines

Decision Function

\[ \mathcal{F}(x) = \langle w, x \rangle + b \]

- Task 1: Learn \( w \) and \( b \) over known examples
- Task 2: Calculate \( \mathcal{F}(x) \) for unknown examples and classify them using \( \text{sign}(\mathcal{F}(x)) \)

Optimization Problem

\[
\min_w \mathcal{L}(\phi w; y) + R(w)
\]

1. Formulated as an optimization problem
2. Consist of a loss and a regularizer term
Linear Support Vector Machines

Decision Function

\[ F(x) = \langle w, x \rangle + b \]

- Task 1: Learn \( w \) and \( b \) over known examples
- Task 2: Calculate \( F(x) \) for unknown examples and classify them using \( \text{sign}(F(x)) \)

Optimization Problem

\[ \min_w CL(\phi w; y) + R(w) \]

1. Formulated as an optimization problem
2. Consist of a loss and a regularizer term
3. "C" is the tradeoff between loss and regularizer (Overfitting Vs Underfitting)
4. Most commonly used loss function is hinge loss \( \max(0, 1 - y_i(\langle w, x_i \rangle + b)) \)
5. Most commonly used regularizer is "Euclidean Regularization"
Regularization

$L^2_2$ is a good choice because:

\[ R(w) = \frac{1}{2} \|w\|_2^2 \]

1. Convex
2. Smooth
3. Simple (e.g. $\nabla w = w$)
Regularization

$L^2_2$ is a good choice because:

$R(w) = \frac{1}{2} \|w\|^2_2$ is

1. Convex
2. Smooth
3. Simple (e.g. $\nabla w = w$)

More Importantly:
Euclidean regularization leads to an amazing generalization beyond finite dimensional feature vectors giving rise to concept of Kernels. [Refer: "Representer Theorem"]
Optimization Problem

$$\min_{w, b} \frac{1}{2} \|w\|_2^2 + C \sum_i \max(0, 1 - y_i(\langle w, x_i \rangle + b))$$
Optimization Problem

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_i \max(0, 1 - y_i(\langle w, x_i \rangle + b))$$

1. Unconstrained
Optimization Problem

\[
\min_{w, b} \frac{1}{2} \|w\|^2 + C \sum_i \max(0, 1 - y_i(\langle w, x_i \rangle + b))
\]

1. Unconstrained
2. Convex
Optimization Problem

\[ \min_{w, b} \frac{1}{2} \|w\|^2 + C \sum_i \max(0, 1 - y_i(\langle w, x_i \rangle + b)) \]

1. Unconstrained
2. Convex
3. Non-smooth
Optimization Problem

\[
\min_{w, b} \frac{1}{2} \|w\|^2 + C \sum_i \max(0, 1 - y_i(\langle w, x_i \rangle + b))
\]

1. Unconstrained
2. Convex
3. Non-smooth
4. Can be optimized using Stochastic Gradient Descent (works in primal)
Optimization Problem

\[
\min_{w, b} \frac{1}{2} \|w\|_2^2 + C \sum_i \max(0, 1 - y_i (\langle w, x_i \rangle + b))
\]

1. Unconstrained
2. Convex
3. Non-smooth
4. Can be optimized using Stochastic Gradient Descent (works in primal)

- Our Goal: To make above problem smooth and then optimize its Dual
Smooth Optimization Problem

\[
\begin{align*}
\min_{w, \xi, b} & \quad \frac{1}{2} \|w\|^2_2 + C \sum_i \xi_i \\
\text{s.t} \quad & \quad y_i (\langle w, x_i \rangle + b) \geq 1 - \xi_i \\
& \quad \xi \geq 0
\end{align*}
\]

1. Constrained
2. Convex
3. Smooth
Deriving the Dual

Lagrangian

- Introducing non-negative dual variables $\alpha$ and $\beta$

$$
\mathcal{L}(w, \xi, \alpha, \beta, b) = \frac{1}{2} \|w\|^2 + C \sum_i \xi_i - \sum_i \alpha_i (y_i(\langle w, x_i \rangle + b) + \xi_i - 1) - \sum_i \beta_i \xi_i
$$
Lagrangian

- Introducing non-negative dual variables $\alpha$ and $\beta$

$$L(w, \xi, \alpha, \beta, b) = \frac{1}{2} \|w\|^2 + C \sum \xi_i - \sum \alpha_i (y_i(\langle w, x_i \rangle + b) + \xi_i - 1) - \sum \beta_i \xi_i$$

What does Lagrangian do?

- Converts constrained optimization problem to un-constrained problem
- Critical points occur at saddle points and not local minima


**Deriving the Dual**

**Lagrangian**

- Introducing non-negative dual variables $\alpha$ and $\beta$

$$
\mathcal{L}(w, \xi, \alpha, \beta, b) = \frac{1}{2} \|w\|^2 + C \sum_i \xi_i - \sum_i \alpha_i(y_i(\langle w, x_i \rangle + b) + \xi_i - 1) - \sum_i \beta_i \xi_i
$$

**What does Lagrangian do?**

- Converts constrained optimization problem to un-constrained problem
- Critical points occur at saddle points and not local minima

**Dual Problem**

$$
\mathcal{D}(\alpha, \beta) = \inf_{w,b,\xi} \mathcal{L}(w, b, \xi, \alpha, \beta)
$$

- $\mathcal{D}^* = \max_{\alpha, \beta} \mathcal{D}(\alpha, \beta)$ and $\mathcal{P}^* = \min_{w, \xi, b} \mathcal{P}(w, \xi, b)$
- **Lower Bound Property**: $\mathcal{P}^* \geq \mathcal{D}^*$
- **Strong Duality**: $\mathcal{P}^* = \mathcal{D}^*$, holds for SVM QP
Deriving the Dual

Writing the gradients

\[ \nabla_w \mathcal{L}(w, b, \xi) = w - \sum_i \alpha_i y_i x_i = 0 \]

\[ \nabla_b \mathcal{L}(w, b, \xi) = \sum_i \alpha_i y_i = 0 \]

\[ \nabla_{\xi_i} \mathcal{L}(w, b, \xi) = C - \alpha_i - \beta_i = 0 \]
Writing the gradients

\[ \nabla_w \mathcal{L}(w, b, \xi) = w - \sum_i \alpha_i y_i x_i = 0 \]

\[ \nabla_b \mathcal{L}(w, b, \xi) = \sum_i \alpha_i y_i = 0 \]

\[ \nabla_{\xi_i} \mathcal{L}(w, b, \xi) = C - \alpha_i - \beta_i = 0 \]

Plug back into Lagrangian

Plug \( w = \sum_i \alpha_i y_i x_i; \beta_i + \alpha_i = C \) and \( \sum_i \alpha_i y_i = 0 \) into \( \mathcal{L}(w, b, \xi, \alpha, \beta) \)

\[
\max_{\alpha \in \mathcal{A}} \mathcal{D}(\alpha) := \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle
\]

\[
\mathcal{A} = \{ \alpha \mid 0 \leq \alpha \leq C \ and \ \sum_i \alpha_i y_i = 0 \}\]
Writing the gradients

\[ \nabla_w L(w, b, \xi) = w - \sum_i \alpha_i y_i x_i = 0 \]

\[ \nabla_b L(w, b, \xi) = \sum_i \alpha_i y_i = 0 \]

\[ \nabla_{\xi_i} L(w, b, \xi) = C - \alpha_i - \beta_i = 0 \]

Plug back into Lagrangian

Plug \( w = \sum_i \alpha_i y_i x_i \); \( \beta_i + \alpha_i = C \) and \( \sum_i \alpha_i y_i = 0 \) into \( L(w, b, \xi, \alpha, \beta) \)

\[ \max_{\alpha \in A} D(\alpha) := \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \]

\[ A = \{ \alpha \mid 0 \leq \alpha \leq C \text{ and } \sum_i \alpha_i y_i = 0 \} \]

- This Dual can be very efficiently optimized by using SMO Algorithm i.e choosing \( \{\alpha_i, \alpha_j\} \) and optimizing over them
Introducing Kernels

Issue

Solution
Introducing Kernels

Issue

▶ What if the data is not linearly separable?

Solution
Introducing Kernels

Issue

- What if the data is not linearly separable?
- This will lead to very poor learning

Solution
Introducing Kernels

Issue
  ▶ What if the data is not linearly separable?
  ▶ This will lead to very poor learning

Solution
  ▶ Project data to a higher dimensional space where we can have a better linear separability
Introducing Kernels

Issue

- What if the data is not linearly separable?
- This will lead to very poor learning

Solution

- Project data to a higher dimensional space where we can have a better linear separability
- Assume: $\phi(x)$ maps $x$ from a lower to a higher dimensional space
Introducing Kernels

Issue

▶ What if the data is not linearly separable?
▶ This will lead to very poor learning

Solution

▶ Project data to a higher dimensional space where we can have a better linear separability
▶ Assume: $\phi(x)$ maps $x$ from a lower to a higher dimensional space
▶ How to store $\phi(x)$?
Dual Problem

\[
\max_{\alpha \in A} \mathcal{D}(\alpha) := \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j) \rangle
\]

\[A = \{ \alpha \mid 0 \leq \alpha \leq C \text{ and } \sum_i \alpha_i y_i = 0 \}\]
Dual Problem

\[
\max_{\alpha \in \mathcal{A}} \mathcal{D}(\alpha) := \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j) \rangle
\]

\[\mathcal{A} = \{ \alpha \mid 0 \leq \alpha \leq C \text{ and } \sum_{i} \alpha_i y_i = 0 \}\]

Decision Function

Recall: \[w = \sum_i \alpha_i y_i \phi(x_i)\]

\[
\mathcal{F}(x) = w^T \phi(x) + b = \sum_{i \in \text{training set}} \alpha_i y_i \langle \phi(x_i), \phi(x) \rangle + b
\]
Dual Problem

\[
\max_{\alpha \in A} \mathcal{D}(\alpha) := \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j) \rangle
\]

\[A = \{ \alpha \mid 0 \leq \alpha \leq C \text{ and } \sum_i \alpha_i y_i = 0 \}\]

Decision Function
Recall: \( w = \sum_i \alpha_i y_i \phi(x_i) \)

\[
\mathcal{F}(x) = w^T \phi(x) + b = \sum_{i \in \text{training set}} \alpha_i y_i \langle \phi(x_i), \phi(x) \rangle + b
\]

- **Observe:** Everywhere \( \langle \phi(x_i), \phi(x_j) \rangle \) (scalar quantity) is required never \( \phi(x) \)
Kernel Function

Definition

\[ k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \]

- With every \( k(\cdot, \cdot) \) there is an associated \( \phi(\cdot) \)
- \( k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \)
- \( k(x_i, x_j) = k(x_j, x_i) \)
- \( K \) is positive semi-definite where \( K_{ij} = k(x_i, x_j) \)
- Refer: Mercer’s Theorem for Proof
Definition

\[ k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \]

- With every \( k(\cdot, \cdot) \) there is an associated \( \phi(\cdot) \)
- \( k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \)
- \( k(x_i, x_j) = k(x_j, x_i) \)
- \( K \) is positive semi-definite where \( K_{ij} = k(x_i, x_j) \)
- Refer: Mercer’s Theorem for Proof

Popular Kernel Functions

RBF: \( k(x, y) = \exp \left( \frac{-\|x - y\|_2^2}{2\gamma^2} \right) \)

Polynomial: \( k(x, y) = \left( 1 + x^T y \right)^n \)
Dual Problem

\[
\max_{\alpha \in \mathcal{A}} D(\alpha) := \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j)
\]

\[\mathcal{A} = \{\alpha \mid 0 \leq \alpha \leq C \text{ and } \sum_i \alpha_i y_i = 0\}\]

Decision Function

Recall: \( w = \sum_i \alpha_i y_i \phi(x_i) \)

\[\mathcal{F}(x) = w^T \phi(x) + b = \sum_{i \in \text{training set}} \alpha_i y_i k(x_i, x_j) + b\]
Motivation for MKL

SVM employs a kernel function \( k(x_i, x_j) \) which intuitively computes similarity between samples \( x_i \) and \( x_j \).

Success of SVM is dependent on the choice of good kernel which are typically hand-crafted and fixed in advance.

Practical learning problems involve multiple, heterogeneous data sources.

Which kernel is better RBF or Polynomial?

If RBF then of what bandwidth?

If Polynomial then of which degree?

Above issues have emphasized the need to consider multiple kernels and not a single fixed kernel.
Motivation for MKL

- SVM employs a kernel function $k(x_i, x_j)$ which intuitively computes similarity between samples $x_i$ and $x_j$
- Success of SVM is dependent on the choice of good kernel which are typically hand-crafted and fixed in advance
Multiple Kernel Learning (MKL)

Motivation for MKL

- SVM employs a kernel function $k(x_i, x_j)$ which intuitively computes similarity between samples $x_i$ and $x_j$
- Success of SVM is dependent on the choice of good kernel which are typically hand-crafted and fixed in advance
- Practical learning problems involve multiple, heterogeneous data sources
Motivation for MKL

> SVM employs a kernel function \( k(x_i, x_j) \) which intuitively computes similarity between samples \( x_i \) and \( x_j \)

> Success of SVM is dependent on the choice of good kernel which are typically hand-crafted and fixed in advance

> Practical learning problems involve multiple, heterogeneous data sources

> Which kernel is better RBF or Polynomial?

> If RBF then of what bandwidth?

> If Polynomial then of which degree?
Motivation for MKL

- SVM employs a kernel function \( k(x_i, x_j) \) which intuitively computes similarity between samples \( x_i \) and \( x_j \)
- Success of SVM is dependent on the choice of good kernel which are typically hand-crafted and fixed in advance
- Practical learning problems involve multiple, heterogeneous data sources
- Which kernel is better RBF or Polynomial?
- If RBF then of what bandwidth?
- If Polynomial then of which degree?
- Above issues have emphasized the need to consider multiple kernels and not a single fixed kernel
MKL Overview

MKL Aim

- MKL learns kernel from training data
- In particular it focuses on how the kernel can be learnt as a linear combination of given base kernels

\[ K(x_i, x_j) = \sum_{k} d_k K_k(x_i, x_j) \]

where \( d_k \geq 0 \) and \( K_k \) are base kernels

1. Resultant \( K(\cdot, \cdot) \) is PSD
2. Our goal is to learn a classifier and kernel weights
MKL Overview

MKL Aim

- MKL learns kernel from training data
- In particular it focuses on how the kernel can be learnt as a linear combination of given base kernels

Linear MKL

\[ K(x_i, x_j) = \sum_k d_k K_k(x_i, x_j) \]

where \( d_k \geq 0 \) and \( K_k \) are base kernels
MKL Aim

- MKL learns kernel from training data
- In particular it focuses on how the kernel can be learnt as a linear combination of given base kernels

Linear MKL

\[ K(x_i, x_j) = \sum_k d_k K_k(x_i, x_j) \]

where \( d_k \geq 0 \) and \( K_k \) are base kernels

1. Resultant \( K(\cdot, \cdot) \) is PSD
MKL Overview

MKL Aim

- MKL learns kernel from training data
- In particular it focuses on how the kernel can be learnt as a linear combination of given base kernels

Linear MKL

\[ K(x_i, x_j) = \sum_{k} d_k K_k(x_i, x_j) \]

where \( d_k \geq 0 \) and \( K_k \) are base kernels

1. Resultant \( K(\cdot, \cdot) \) is PSD
2. Our goal is to learn a classifier and kernel weights \( d_k \)
Linear MKL Geometric interpretation

- Linear combination of base kernels corresponds to concatenation of individual kernel feature spaces

\[ \phi(x) = \begin{bmatrix} \sqrt{d_1}\phi_1(x) \\ \sqrt{d_2}\phi_2(x) \\ \vdots \\ \sqrt{d_M}\phi_M(x) \end{bmatrix} \]
Linear MKL Geometric interpretation

- Linear combination of base kernels corresponds to concatenation of individual kernel feature spaces

\[
\phi(x) = \begin{bmatrix}
\sqrt{d_1}\phi_1(x) \\
\sqrt{d_2}\phi_2(x) \\
\vdots \\
\sqrt{d_M}\phi_M(x)
\end{bmatrix}
\]

- We can see

\[
K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle = d_1 \langle \phi_1(x_i), \phi_1(x_j) \rangle + \cdots + d_M \langle \phi_M(x_i), \phi_M(x_j) \rangle
\]
Linear MKL Geometric interpretation

- Linear combination of base kernels corresponds to concatenation of individual kernel feature spaces

\[
\phi(x) = \begin{bmatrix}
\sqrt{d_1} \phi_1(x) \\
\sqrt{d_2} \phi_2(x) \\
\vdots \\
\sqrt{d_M} \phi_M(x)
\end{bmatrix}
\]

- We can see

\[
K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle = d_1 \langle \phi_1(x_i), \phi_1(x_j) \rangle + \cdots + d_M \langle \phi_M(x_i), \phi_M(x_j) \rangle
\]

\[
K(x_i, x_j) = d_1 K_1(x_i, x_j) + \cdots + d_M K_M(x_i, x_j) = \sum_k d_k K_k(x_i, x_j)
\]
MKL Formulations and Optimization

- Kernel Target Alignment [Cristianini et al. 2001]
- Semi-Definite Programming-MKL (SDP-MKL) [Lanckriet et al. 2002]
- Block $l_1$-MKL (M-Y regularization + SMO) [Bach et al. 2004]
- Semi-Infinite Linear Programming-MKL (SILP) [Sonnenberg et al. 2005]
- Simple MKL (gradient descent) [Rakotomamonjy et al. 2007]
- GMKL [Varma & Babu ICML 2009]
- SMO-MKL [Vishwanathan & Varma NIPS 2010]
Optimization Problem

\[
\begin{align*}
\min_{w, \xi, b, d} & \quad \sum_k \frac{1}{2} w_k^T w_k + C \sum_i \xi_i \\
\text{st.} & \quad y_i \left( \sum_k \sqrt{d_k} w_k^T \phi_k(x_i) + b \right) \geq 1 - \xi_i \\
& \quad \xi \geq 0 \\
& \quad d \geq 0
\end{align*}
\]
Optimization Problem

\[
\begin{align*}
\min_{\mathbf{w}, \xi, b, d} & \quad \sum_k \frac{1}{2} \mathbf{w}_k^T \mathbf{w}_k + C \sum_i \xi_i \\
\text{st.} & \quad y_i \left( \sum_k \sqrt{d_k} \mathbf{w}_k^T \phi_k(x_i) + b \right) \geq 1 - \xi_i \\
& \quad \xi \geq 0 \\
& \quad d \geq 0
\end{align*}
\]

- This problem is non-convex because of the presence of \( \sqrt{d_k} \)
Optimization Problem

\[
\begin{align*}
\min_{w, \xi, b, d} & \quad \sum_k \frac{1}{2} w_k^T w_k + C \sum_i \xi_i \\
\text{st.} & \quad y_i \left( \sum_k \sqrt{d_k} w_k^T \phi_k(x_i) + b \right) \geq 1 - \xi_i \\
& \quad \xi \geq 0 \\
& \quad d \geq 0
\end{align*}
\]

- This problem is non-convex because of the presence of \( \sqrt{d_k} \)
- Replace \( w_k \rightarrow \frac{w_k}{\sqrt{d_k}} \)
Optimization Problem

\[
\begin{align*}
\min_{w, \xi \geq 0, b, d \geq 0} & \quad \frac{1}{2} \sum_k w_k^T w_k + C \sum_i \xi_i \\
\text{st.} & \quad y_i \left( \sum_k w_k^T \phi_k(x_i) + b \right) \geq 1 - \xi_i
\end{align*}
\]
Optimization Problem

\[
\begin{align*}
\min_{w, \xi \geq 0, b, d \geq 0} & \quad \frac{1}{2} \sum_k w_k^T w_k \frac{1}{d_k} + C \sum_i \xi_i \\
\text{st.} & \quad y_i \left( \sum_k w_k^T \phi_k(x_i) + b \right) \geq 1 - \xi_i
\end{align*}
\]

- Introducing a regularizer \( r(d) \) in objective to prevent \( d \) from going to infinity

\[
\begin{align*}
\min_{w, \xi \geq 0, b, d \geq 0} & \quad \frac{1}{2} \sum_k w_k^T w_k \frac{1}{d_k} + C \sum_i \xi_i + \lambda r(d) \\
\text{st.} & \quad y_i \left( \sum_k w_k^T \phi_k(x_i) + b \right) \geq 1 - \xi_i
\end{align*}
\]
Some Popular Regularizers

**\(l_p\) norm**

- \(r(d) = \|d\|_p^2\) where \(p \geq 1\)
- Special cases: \(r(d) = \|d\|_1^2\) and \(r(d) = \|d\|_2^2\)

**Elastic Net**

- \(r(d) = \lambda_1 \|d\|_1 + \frac{\lambda_2}{2} \|d\|_2^2\)

**Entropy Regularizer**

- \(r(d) = \sum_{m=1}^{M} d_m \log \frac{d_m}{d_0^m}; \quad d_m^0 = \frac{1}{M}\)
Partial Lagrangian

Introducing non-negative lagrangian multipliers $\alpha$ and $\beta$ and writing the partial lagrangian

$$L(w, \xi, \alpha, \beta, b, d) = \sum_k \frac{1}{2} \frac{w_k^T w_k}{d_k} + C \sum_i \xi_i + \lambda r(d)$$

$$- \sum_i \beta_i \xi_i - \sum_i \alpha_i \left( y_i \left( \sum_k w_k^T \phi_k(x_i) + b \right) - 1 + \xi_i \right)$$
MKL Dual

Partial Lagrangian

Introducing non-negative lagrangian multipliers $\alpha$ and $\beta$ and writing the partial lagrangian

$$
\mathcal{L}(w, \xi, \alpha, \beta, b, d) = \sum_k \frac{1}{2} \frac{w_k^T w_k}{d_k} + C \sum_i \xi_i + \lambda r(d)
$$

$$
- \sum_i \beta_i \xi_i - \sum_i \alpha_i \left( y_i \left( \sum_k w_k^T \phi_k(x_i) + b \right) - 1 + \xi_i \right)
$$

Taking derivative wrt to $w, \xi, b$ and setting them to zero we obtain

$$
\frac{\partial \mathcal{L}}{\partial w_k} = \frac{w_k}{d_k} - \sum_i \alpha_i y_i \phi_k(x_i) = 0 \implies \frac{w_k}{d_k} = \sum_i \alpha_i y_i \phi_k(x_i)
$$

$$
\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \beta_i - \alpha_i = 0 \implies \beta_i + \alpha_i = C
$$

$$
\frac{\partial \mathcal{L}}{\partial b} = \sum_i \alpha_i y_i = 0
$$
Partial Dual

Substitute the derivatives back into the lagrangian we can the intermediate saddle point problem

\[
J(\alpha, d) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \sum_{m=1}^{M} d_m k_m(x_i, x_j) + \lambda r(d)
\]
Partial Dual
Substitute the derivatives back into the lagrangian we can the intermediate saddle point problem

\[ J(\alpha, d) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \sum_{m=1}^{M} d_m k_m(x_i, x_j) + \lambda r(d) \]

As a consequence of convex duality

\[
\max_{\alpha \in A} \min_{d \in D} J(\alpha, d) = \min_{d \in D} \max_{\alpha \in A} J(\alpha, d)
\]

where \( A = \{\alpha| 0 \leq \alpha \leq C; \sum_{i} \alpha_i y_i = 0\} \)

\( D = \{d | d \geq 0\} \)
Partial Dual
Substitute the derivatives back into the lagrangian we can the intermediate saddle point problem

\[
J(\alpha, d) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \sum_{m=1}^{M} d_m k_m(x_i, x_j) + \lambda r(d)
\]

As a consequence of convex duality

\[
\max_{\alpha \in \mathcal{A}} \min_{d \in \mathcal{D}} J(\alpha, d) = \min_{d \in \mathcal{D}} \max_{\alpha \in \mathcal{A}} J(\alpha, d)
\]

where \(\mathcal{A} = \{\alpha | 0 \leq \alpha \leq C; \sum_i \alpha_i y_i = 0\}\)

\(\mathcal{D} = \{d | d \geq 0\}\)

- All existing MKL solvers stops over here and solves the problem by iteratively minimizing over \(d\) and maximizing over \(\alpha\) [known as Wrapper Method]
For certain choices of regularizers \( r(d) \), \( d \)'s can be completely eliminated and the resulting dual can be optimized using state of the art SMO algorithm.
For certain choices of regularizers \( r(d) \), \( d \)'s can be completely eliminated and the resulting dual can be optimized using state of the art SMO algorithm.

\[ l_p \] norm

Taking \( r(d) = \|d\|^2 \)

\[
\min_{d \in D} \max_{\alpha \in A} 1^T \alpha - \frac{1}{2} \sum_k d_k \alpha^T H_k \alpha + \frac{\lambda}{2} \|d\|^2_p
\]
For certain choices of regularizers ($r(d)$), $d$'s can be completely eliminated and the resulting dual can be optimized using state of the art SMO algorithm.

$l_p$ norm

Taking $r(d) = \|d\|^2_p$

\[
\min_{d \in D} \max_{\alpha \in A} 1^T \alpha - \frac{1}{2} \sum_k d_k \alpha^T H_k \alpha + \frac{\lambda}{2} \|d\|^2_p
\]

Now to eliminate $d$ again write the lagrangian by introducing non-negative lagrangian variable $\gamma$

\[
\mathcal{L} = 1^T \alpha - \frac{1}{2} \sum_k d_k \alpha^T H_k \alpha + \frac{\lambda}{2} \|d\|^2_p - \sum_k \gamma_k d_k
\]
For certain choices of regularizers \( r(d) \), \( d \)'s can be completely eliminated and the resulting dual can be optimized using state of the art SMO algorithm.

\( l_p \) norm

Taking \( r(d) = \|d\|_p^2 \)

\[
\begin{align*}
\min_{d \in D} \max_{\alpha \in A} & \quad 1^T \alpha - \frac{1}{2} \sum_k d_k \alpha^T H_k \alpha + \frac{\lambda}{2} \|d\|_p^2 \\
\end{align*}
\]

Now to eliminate \( d \) again write the lagrangian by introducing non-negative lagrangian variable \( \gamma \)

\[
L = 1^T \alpha - \frac{1}{2} \sum_k d_k \alpha^T H_k \alpha + \frac{\lambda}{2} \|d\|_p^2 - \sum_k \gamma_k d_k
\]

Setting derivative wrt \( d \) equal to 0

\[
\frac{\partial L}{\partial d_k} = 0 \implies \lambda \left( \sum_k d_k^p \right)^{\frac{2}{p}-1} d_k^{p-1} = \gamma_k + \frac{1}{2} \alpha^T H_k \alpha
\]
Dual

\[ \implies \lambda \left( \sum_k d_k^p \right)^2 = \sum_k d_k \left( \gamma_k + \frac{1}{2} \alpha^T H_k \alpha \right) \]

Substituting this back into the Lagrangian
Continued...

Dual

\[ \implies \lambda \left( \sum_k d_k^p \right)^{\frac{2}{p}} = \sum_k d_k \left( \gamma_k + \frac{1}{2} \alpha^T H_k \alpha \right) \]

Substituting this back into the Lagrangian

\[ L = 1^T \alpha - \frac{\lambda}{2} \left( \sum_k d_k^p \right)^{\frac{2}{p}} \]

\[ = 1^T \alpha - \frac{1}{2\lambda} \left( \sum_k \left( \gamma_k + \frac{1}{2} \alpha^T H_k \alpha \right)^q \right)^{\frac{2}{q}} \]

where \( \frac{1}{p} + \frac{1}{q} = 1 \)
Dual

\[ \implies \lambda \left( \sum_k d_k^p \right)^2 = \sum_k d_k (\gamma_k + \frac{1}{2} \alpha^T H_k \alpha) \]

Substituting this back into the Lagrangian

\[ \mathcal{L} = 1^T \alpha - \frac{\lambda}{2} \left( \sum_k d_k^p \right)^2 \]

\[ = 1^T \alpha - \frac{1}{2\lambda} \left( \sum_k (\gamma_k + \frac{1}{2} \alpha^T H_k \alpha)^q \right)^2 \]

where \( \frac{1}{p} + \frac{1}{q} = 1 \)

Since \( H_k \) is PSD, \( \alpha^T H_k \alpha \geq 0 \), and \( \gamma_k \geq 0 \). Therefore optimal value of \( \gamma_k \) is zero

\[ D = \max_{\alpha \in A} 1^T \alpha - \frac{1}{8\lambda} \left( \sum_k (\alpha^T H_k \alpha)^q \right)^2 \]
Extension to other regularizers

Elastic Net

\[ r(d) = \lambda_1 \|d\|_1 + \frac{\lambda_2}{2} \|d\|_2^2 \]

\[ D = \max_{\alpha \in \mathcal{A}} 1^T \alpha - \frac{1}{2\lambda_2} \sum_k \left( \max \left(0, \frac{1}{2} \alpha^T H_k \alpha - \lambda_1 \right) \right)^2 \]

\[ d_k = \frac{1}{\lambda_2} \max(0, \frac{1}{2} \alpha^T H_k \alpha - \lambda_1) \]
Extension to other regularizers

Elastic Net

\[ r(d) = \lambda_1 \|d\|_1 + \frac{\lambda_2}{2} \|d\|_2^2 \]

\[ D = \max_{\alpha \in A} 1^T \alpha - \frac{1}{2\lambda_2} \sum_k \left( \max \left( 0, \frac{1}{2} \alpha^T H_k \alpha - \lambda_1 \right) \right)^2 \]

\[ d_k = \frac{1}{\lambda_2} \max(0, \frac{1}{2} \alpha^T H_k \alpha - \lambda_1) \]

Other supported regularizers

- Normalized Entropy: \( r(d) = \sum_{m=1}^{M} d_m \log \frac{d_m}{d_0^m} \) where \( d_0^m = \frac{1}{M} \)
- Un-normalized Entropy: \( r(d) = \sum_{m=1}^{M} \left( d_m \log \frac{d_m}{d_0^m} + d_0^m - d_m \right) \)
- Generalized Elastic Net: \( r(d) = \lambda_1 \|d\|_1 + \frac{\lambda_2}{2} \|d\|_2^p \)
Optimizing the dual

Three essential questions

1. Selection criteria for \( \{\alpha_i, \alpha_j\} \)?
2. How to optimize over selected \( \{\alpha_i, \alpha_j\} \)?
3. When to stop?
Optimizing the dual

Three essential questions

1. Selection criteria for \( \{\alpha_i, \alpha_j\} \)?
2. How to optimize over selected \( \{\alpha_i, \alpha_j\} \)?
3. When to stop?

Reduced Variable Optimization

- SMO algorithm works by repeatedly choosing two variables \( \alpha_1 \) and \( \alpha_2 \)
- \( \alpha_1 \) and \( \alpha_2 \) are optimized while keeps all other variables constant
- Optimizing over \( \alpha_1 \) and \( \alpha_2 \) is a one dimensional concave optimization problem, which is efficiently optimized by Newton-Raphson method
Optimizing the dual

Three essential questions

1. Selection criteria for \( \{\alpha_i, \alpha_j\} \)?
2. How to optimize over selected \( \{\alpha_i, \alpha_j\} \)?
3. When to stop?

Reduced Variable Optimization

- SMO algorithm works by repeatedly choosing two variables \( \alpha_1 \) and \( \alpha_2 \)
- \( \alpha_1 \) and \( \alpha_2 \) are optimized while keeps all other variables constant
- Optimizing over \( \alpha_1 \) and \( \alpha_2 \) is a one dimensional concave optimization problem, which is efficiently optimized by Newton-Raphson method

Working Set Selection

- Selection of \( \alpha_1 \) and \( \alpha_2 \) have a big impact on training time
- Second order working set selection technique is used
- Selection of first variable involves computing \( \nabla_{\alpha} D \)
- Selection of second variable involves computing \( \nabla^2_{\alpha} D \)
\[ D = \min_{\alpha \in A} 1^T \alpha - \frac{1}{8\lambda} \left( \sum_k (\alpha^T H \alpha)^2 \right) \]

- No need to do an inexact line search while optimizing over \( \{\alpha_i, \alpha_j\} \)
- Can get the exact step length by finding roots of the cubic
Motivation

- Present MKL approaches are limited to linear combination of kernels
- Far richer representations can be achieved by combining kernels in other fashions
Motivation

- Present MKL approaches are limited to linear combination of kernels
- Far richer representations can be achieved by combining kernels in other fashions
- For example: Taking product of kernels corresponds to taking tensor products of their feature spaces
Motivation

- Present MKL approaches are limited to linear combination of kernels
- Far richer representations can be achieved by combining kernels in other fashions
- For example: Taking product of kernels corresponds to taking tensor products of their feature spaces

GMKL Formulation

\[
\min_{w, b, d \geq 0} \frac{1}{2} w^T w + \sum_i l(y_i, f(x_i)) + r(d)
\]
Formulation

\[
\min_d \ T(d) \quad \text{subject to} \quad d \geq 0
\]

where \( T(d) = \min_{w, b} \frac{1}{2} w^T w + \sum_i l(y_i, f(x_i)) + r(d) \)

1. The primal is formulated as a two-step optimization problem.
2. In the outer loop, the kernel is learnt by optimizing over \( d \).
3. In the inner loop, the kernel is held fixed and SVM parameters are learned.
Formulation

$$\min_d \ T(d) \ \text{subject to} \ d \geq 0$$

where $T(d) = \min_{w,b} \ \frac{1}{2} w^T w + \sum_i l(y_i, f(x_i)) + r(d)$

1. The primal is formulated as two step optimization problem
2. In outer loop kernel is learnt by optimizing over $d$
3. In inner loop kernel is held fixed and SVM parameters are learned

Does $\nabla_d T(d)$ exists ?

In order to show this lets move to the dual
Dual for Classification Problem

\[ W(d) = \max_{\alpha} \ 1^T \alpha - \frac{1}{2} \alpha^T Y K_d Y \alpha + r(d) \]

Subject to \( 1^T Y \alpha = 0, \ 0 \leq \alpha \leq C \)

- For any given \( d \), \( T(d) = W(d) \) because of strong duality
GMKL Dual

Dual for Classification Problem

\[ W(d) = \max_\alpha \ 1^T \alpha - \frac{1}{2} \alpha^T Y K_d Y \alpha + r(d) \]

Subject to \( 1^T Y \alpha = 0, \ 0 \leq \alpha \leq C \)

- For any given \( d \), \( T(d) = W(d) \) because of strong duality
- So it is sufficient to show that \( \nabla_d W \) exists

Partial derivatives:

\[ \partial_{d_i} T = \partial_{d_i} W = \partial_{d_i} r - \frac{1}{2} \alpha^* \partial_{d_i} \alpha \]
GMKL Dual

Dual for Classification Problem

\[ W(d) = \max_{\alpha} \quad 1^T \alpha - \frac{1}{2} \alpha^T Y K_d Y \alpha + r(d) \]

Subject to \( 1^T Y \alpha = 0, \ 0 \leq \alpha \leq C \)

- For any given \( d \), \( T(d) = W(d) \) because of strong duality
- So it is sufficient to show that \( \nabla_d W \) exists
- Differentiability of \( W(d) \) directly comes from Danskin’s Theorem
Dual for Classification Problem

\[ W(d) = \max_\alpha \ 1^T \alpha - \frac{1}{2} \alpha^T Y K_d Y \alpha + r(d) \]
Subject to \( 1^T Y \alpha = 0, \ 0 \leq \alpha \leq C \)

- For any given \( d \), \( T(d) = W(d) \) because of strong duality
- So it is sufficient to show that \( \nabla_d W \) exists
- Differentiability of \( W(d) \) directly comes from Danskin’s Theorem

Derivative exists if

1. \( K_d \) is strictly positive definite
2. \( \nabla_d K_d \) and \( \nabla_d r(d) \) exists
3. \( \alpha^* \) which optimizes \( W(d) \) is unique

\[ \frac{\partial T}{\partial d_k} = \frac{\partial W}{\partial d_k} = \frac{\partial r}{\partial d_k} - \frac{1}{2} \alpha^{*t} \frac{\partial H}{\partial d_k} \alpha^* \]
Steps Involved

1. For a given $d$, a single kernel problem is solved to obtain $\alpha^*$

2. $d_k^{n+1} := d_k^n - s^n \left( \frac{\partial r}{\partial d_k} - \frac{1}{2} \alpha^* t \frac{\partial H}{\partial d_k} \alpha \right)$

3. Projecting $d_k^{n+1}$ onto the feasible set

Above steps are repeated till convergence
Steps Involved

1. For a given $d$, a single kernel problem is solved to obtain $\alpha^*$

2. $d_{k}^{n+1} := d_{k}^{n} - s^{n} \left( \frac{\partial r}{\partial d_k} - \frac{1}{2} \alpha^* t \frac{\partial H}{\partial d_k} \alpha \right)$

3. Projecting $d_{k}^{n+1}$ onto the feasible set

Above steps are repeated till convergence

- Step $s^n$ is chosen based on Armijo rule (monotone) to guarantee convergence
Steps Involved

1. For a given $d$, a single kernel problem is solved to obtain $\alpha^*$

2. $d_{k}^{n+1} := d_{k}^{n} - s^n \left( \frac{\partial r}{\partial d_k} - \frac{1}{2} \alpha^* t \frac{\partial H}{\partial d_k} \alpha \right)$

3. Projecting $d_{k}^{n+1}$ onto the feasible set

Above steps are repeated till convergence

- Step $s^n$ is chosen based on Armijo rule (monotone) to guarantee convergence
- Every single update of $d_k$ may involve many SVM solving
- Solving inner SVM is very expensive as high precision solution is required
Optimizing \( d \)

- Direction of descent is "Spectral Projected Gradient" \( (\nabla_{SPG} W(d)) \)
- Step size is chosen based on non-monotone line search condition
Optimizing $d$

- Direction of descent is "Spectral Projected Gradient" ($\nabla_{SPG} W(d)$)
- Step size is chosen based on non-monotone line search condition

Taking step in $d$

We take a step in $d$ ($\lambda$ is the step size)

$$d_+ = d_k + \lambda \nabla_{SPG} W(d_k)$$

If, non-monotone line search condition is satisfied

$$W(d_+) \leq \max_{0 \leq j \leq \min\{k, M-1\}} W(d_{k-j}) + \gamma \lambda \langle \nabla_{SPG} W(d_k), \nabla_d W(d_k) \rangle$$
Optimizing $d$

- Direction of descent is "Spectral Projected Gradient" ($\nabla_{\text{SPG}} W(d)$)
- Step size is chosen based on non-monotone line search condition

Taking step in $d$

We take a step in $d$ ($\lambda$ is the step size)

$$d_+ = d_k + \lambda \nabla_{\text{SPG}} W(d_k)$$

If, non-monotone line search condition is satisfied

$$W(d_+) \leq \max_{0 \leq j \leq \min\{k, M-1\}} W(d_{k-j}) + \gamma \lambda \langle \nabla_{\text{SPG}} W(d_k), \nabla_d W(d_k) \rangle$$

Where

Projected Gradient: $\nabla_{\text{SPG}} W(d_k) = P(d_k - s_k \nabla_d W(d_k)) - d_k$

Spectral Step Length: $s_k = \frac{\langle d_k - d_{k-1}, d_k - d_{k-1} \rangle}{\langle d_k - d_{k-1}, \nabla_d W(d_k) - \nabla_d W(d_{k-1}) \rangle}$
SMO-MKL Code:
References