5: Vector Data: Support Vector Machine

Instructor: Yizhou Sun
yzsun@cs.ucla.edu

January 23, 2019
# Methods to Learn: Last Lecture

<table>
<thead>
<tr>
<th></th>
<th>Vector Data</th>
<th>Set Data</th>
<th>Sequence Data</th>
<th>Text Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Classification</strong></td>
<td><strong>Logistic Regression; Decision Tree; KNN</strong>&lt;br&gt; SVM; NN</td>
<td></td>
<td></td>
<td><strong>Naïve Bayes for Text</strong></td>
</tr>
<tr>
<td><strong>Clustering</strong></td>
<td>K-means; hierarchical clustering; DBSCAN; DBSCAN; Mixture Models</td>
<td></td>
<td></td>
<td><strong>PLSA</strong></td>
</tr>
<tr>
<td><strong>Prediction</strong></td>
<td><strong>Linear Regression</strong>&lt;br&gt; GLM*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Frequent Pattern Mining</strong></td>
<td>Apriori; FP growth</td>
<td></td>
<td>GSP; PrefixSpan</td>
<td></td>
</tr>
<tr>
<td><strong>Similarity Search</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>DTW</strong></td>
</tr>
</tbody>
</table>
# Methods to Learn

<table>
<thead>
<tr>
<th></th>
<th>Vector Data</th>
<th>Set Data</th>
<th>Sequence Data</th>
<th>Text Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Classification</strong></td>
<td>Logistic Regression; Decision Tree; KNN <strong>SVM</strong>; NN</td>
<td></td>
<td></td>
<td>Nä ve Bayes for Text</td>
</tr>
<tr>
<td><strong>Clustering</strong></td>
<td>K-means; hierarchical clustering; DBSCAN; DBSCAN; Mixture Models</td>
<td></td>
<td></td>
<td>PLSA</td>
</tr>
<tr>
<td><strong>Prediction</strong></td>
<td>Linear Regression GLM*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Frequent Pattern Mining</strong></td>
<td>Apriori; FP growth</td>
<td></td>
<td>GSP; PrefixSpan</td>
<td></td>
</tr>
<tr>
<td><strong>Similarity Search</strong></td>
<td></td>
<td></td>
<td></td>
<td>DTW</td>
</tr>
</tbody>
</table>
Support Vector Machine

• Introduction
• Linear SVM
• Non-linear SVM
• Scalability Issues*
• Summary
Math Review

• **Vector**
  - \( x = (x_1, x_2, \ldots, x_n) \)
  - **Subtracting two vectors:** \( x = b - a \)

• **Dot product**
  - \( a \cdot b = \sum a_i b_i \)
  - **Geometric interpretation:** projection
  - If **\( a \) and \( b \) are orthogonal**, \( a \cdot b = 0 \)
• Plane/Hyperplane
  • $a_1 x_1 + a_2 x_2 + \cdots + a_n x_n = c$
  • Line (n=2), plane (n=3), hyperplane (higher dimensions)

• Normal of a plane
  • $n = (a_1, a_2, \ldots, a_n)$
  • a vector which is perpendicular to the surface
Math Review (Cont.)

• Define a plane using normal \( \mathbf{n} = (a, b, c) \) and a point \((x_0, y_0, z_0)\) in the plane:
  
  \[
  (a, b, c) \cdot (x_0 - x, y_0 - y, z_0 - z) = 0 \Rightarrow \\
  ax + by + cz = ax_0 + by_0 + cz_0 (= d)
  \]

• Distance from a point \((x_0, y_0, z_0)\) to a plane \(ax + by + cz = d\):
  
  \[
  \left| (x_0 - x, y_0 - y, z_0 - z) \cdot \frac{(a, b, c)}{||(a, b, c)||} \right| = \\
  \frac{|ax_0 + by_0 + cz_0 - d|}{\sqrt{a^2 + b^2 + c^2}}
  \]
Linear Classifier

• Given a training dataset \( \{x_i, y_i\}_{i=1}^{N} \)
  
  - A separating hyperplane can be written as a linear combination of attributes
    \[ W \cdot x + b = 0 \]
    
    where \( W = \{w_1, w_2, ..., w_n\} \) is a weight vector and \( b \) a scalar (bias)
  
  - For 2-D it can be written as
    \[ w_0 + w_1 x_1 + w_2 x_2 = 0 \]
  
  - Classification:
    \[ w_0 + w_1 x_1 + w_2 x_2 > 0 \implies y_i = +1 \]
    \[ w_0 + w_1 x_1 + w_2 x_2 \leq 0 \implies y_i = -1 \]
Recall

• Is the decision boundary for logistic regression linear?

• Is the decision boundary for decision tree linear?
Simple Linear Classifier: Perceptron

\( x = (1, x_1, x_2, \ldots, x_d)^T \quad w = (\omega_0, \omega_1, \omega_2, \ldots, \omega_d)^T \)

\( y = \{1, -1\} \quad \alpha \in (0, 1] \) (learning rate)

Initialize \( w = 0 \) (can be any vector)

Repeat:

- For each training example \((x_i, y_i)\):
  - Compute \( \hat{y}_i = \text{sign}(w^T x_i) \)
  - If \((y_i \neq \hat{y}_i)\)
    \[ w = w + \alpha (y_i x_i) \]

Until \((y_i = \hat{y}_i \quad \forall i = 1 \ldots N)\)

Return \( w \)

Loss function: \( \max\{0, -y_i \ast w^T x_i\} \)
### Example ($\alpha = 0.9$)

<table>
<thead>
<tr>
<th>x0</th>
<th>x1</th>
<th>x2</th>
<th>true label</th>
<th>$w$ before update</th>
<th>predicted label</th>
<th>$w$ after update</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Y</td>
<td>(0.0, 0.0, 0.0)</td>
<td>N</td>
<td>(0.9, 0.0, 0.9)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>N</td>
<td>(0.9, 0.0, 0.9)</td>
<td>Y</td>
<td>(0.0, -0.9, 0.0)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Y</td>
<td>(0.0, -0.9, 0.0)</td>
<td>N</td>
<td>(0.9, -0.9, 0.0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Y</td>
<td>(0.9, -0.9, 0.0)</td>
<td>N</td>
<td>(1.8, 0.0, 0.0)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Y</td>
<td>(1.8, 0.0, 0.0)</td>
<td>Y</td>
<td>(1.8, 0.0, 0.0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>N</td>
<td>(1.8, 0.0, 0.0)</td>
<td>Y</td>
<td>(0.9, -0.9, -0.9)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Y</td>
<td>(0.9, -0.9, -0.9)</td>
<td>Y</td>
<td>(0.9, -0.9, -0.9)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Y</td>
<td>(0.9, -0.9, -0.9)</td>
<td>N</td>
<td>(1.8, 0.0, -0.9)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Y</td>
<td>(1.8, 0.0, -0.9)</td>
<td>Y</td>
<td>(1.8, 0.0, -0.9)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>N</td>
<td>(1.8, 0.0, -0.9)</td>
<td>Y</td>
<td>(0.9, -0.9, -1.8)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Y</td>
<td>(0.9, -0.9, -1.8)</td>
<td>Y</td>
<td>(0.9, -0.9, -1.8)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Y</td>
<td>(0.9, -0.9, -1.8)</td>
<td>N</td>
<td>(1.8, 0.0, -1.8)</td>
</tr>
</tbody>
</table>
Support Vector Machine

• Introduction

• Linear SVM

• Non-linear SVM

• Scalability Issues*

• Summary
Can we do better?

• Which hyperplane to choose?
SVM—Margins and Support Vectors

Small Margin

Large Margin

Support Vectors
Let data $D$ be $(X_1, y_1), \ldots, (X_{|D|}, y_{|D|})$, where $X_i$ is the set of training tuples associated with the class labels $y_i$.

There are infinite lines (hyperplanes) separating the two classes but we want to find the best one (the one that minimizes classification error on unseen data).

* SVM searches for the hyperplane with the largest margin, i.e., maximum marginal hyperplane (MMH)
A separating hyperplane can be written as

\[ W \cdot X + b = 0 \]

The hyperplane defining the sides of the margin, e.g.,:

\[ H_1: w_1 x_1 + w_2 x_2 + b \geq 1 \quad \text{for } y_i = +1, \text{ and} \]

\[ H_2: w_1 x_1 + w_2 x_2 + b \leq -1 \quad \text{for } y_i = -1 \]

Any training tuples that fall on hyperplanes \( H_1 \) or \( H_2 \) (i.e., the sides defining the margin) are support vectors.

This becomes a constrained (convex) quadratic optimization problem: Quadratic objective function and linear constraints → Quadratic Programming (QP) → Lagrangian multipliers
Maximum Margin Calculation

- \( \mathbf{w} \): decision hyperplane normal vector
- \( \mathbf{x}_i \): data point \( i \)
- \( y_i \): class of data point \( i \) (+1 or -1)

\[
\mathbf{w}^T \mathbf{x} + b = 0
\]

\[
\mathbf{w}^T \mathbf{x}_a + b = 1
\]

\[
\mathbf{w}^T \mathbf{x}_b + b = -1
\]

\[
\rho = \frac{2}{||\mathbf{w}||}
\]

**Hint:** what is the distance between \( x_a \) and \( \mathbf{w}^T \mathbf{x} + b = -1 \)**
SVM as a Quadratic Programming

- QP
  
  Objective: Find \( w \) and \( b \) such that \( \rho = \frac{2}{||w||} \) is maximized;
  
  Constraints: For all \( \{(x_i, y_i)\} \)
  
  \[ w^T x_i + b \geq 1 \text{ if } y_i = 1; \]
  
  \[ w^T x_i + b \leq -1 \text{ if } y_i = -1 \]

- A better form
  
  Objective: Find \( w \) and \( b \) such that \( \Phi(w) = \frac{1}{2} w^T w \) is minimized;
  
  Constraints: for all \( \{(x_i, y_i)\} : y_i (w^T x_i + b) \geq 1 \)
Solve QP

• This is now optimizing a *quadratic* function subject to *linear* constraints

• Quadratic optimization problems are a well-known class of mathematical programming problem, and many (intricate) algorithms exist for solving them (with many special ones built for SVMs)

• The solution involves constructing a *dual problem* where a *Lagrange multiplier* $\alpha_i$ is associated with every constraint in the primary problem:
Lagrange Formulation

• Introducing Lagrange multipliers $\alpha_i \geq 0$ for each constraint

Minimize

$$L(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i (y_i (w^T x_i + b) - 1)$$

Take the partial derivatives w.r.t $w, b$:

$$\nabla_w L = w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0 \implies w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = - \sum_{i=1}^{N} \alpha_i y_i = 0$$
Objective: Find $w$ and $b$ such that $\Phi(w) = \frac{1}{2} w^T w$ is minimized;

Constraints: for all $\{(x_i, y_i)\}$: $y_i (w^T x_i + b) \geq 1$

Equivalent under some conditions; also $w, b, \alpha$ satisfy KKT conditions

Objective: Find $\alpha_1...\alpha_n$ such that

$Q(\alpha) = \Sigma \alpha_i - \frac{1}{2} \Sigma \Sigma \alpha_i \alpha_j y_i y_j x_i^T x_j$ is maximized and

Constraints

(1) $\Sigma \alpha_i y_i = 0$

(2) $\alpha_i \geq 0$ for all $\alpha_i$

The Optimization Problem Solution

- The solution has the form:

\[ w = \sum \alpha_i y_i x_i \quad b = y_k - w^T x_k \text{ for any } x_k \text{ such that } \alpha_k \neq 0 \]

- Each non-zero \( \alpha_i \) indicates that corresponding \( x_i \) is a support vector.
- Then the classifying function will have the form:

\[ f(x) = \sum \alpha_i y_i x_i^T x + b \]

- Notice that it relies on an *inner product* between the test point \( x \) and the support vectors \( x_i \)
  - We will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products \( x_i^T x_j \) between all pairs of training points.
Soft Margin Classification

- If the training data is not linearly separable, *slack variables* $\xi_i$ can be added to allow misclassification of difficult or noisy examples.
- Allow some errors
- Let some points be moved to where they belong, at a cost
- Still, try to minimize training set errors, and to place hyperplane “far” from each class (large margin)
Soft Margin Classification
Mathematically

• The old formulation:

Find \( w \) and \( b \) such that
\[
\Phi(w) = \frac{1}{2} w^T w \text{ is minimized and for all } \{(x_i, y_i)\}
\]
\[
y_i (w^T x_i + b) \geq 1
\]

• The new formulation incorporating slack variables:

Find \( w \) and \( b \) such that
\[
\Phi(w) = \frac{1}{2} w^T w + C \sum \xi_i \text{ is minimized and for all } \{(x_i, y_i)\}
\]
\[
y_i (w^T x_i + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \text{ for all } i
\]

• Parameter \( C \) can be viewed as a way to control overfitting
  • A regularization term (L1 regularization)
Soft Margin Classification – Solution

- The dual problem for soft margin classification:

Find \( \alpha_1 \ldots \alpha_N \) such that
\[
Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j x_i^T x_j \]
is maximized and
\[
(1) \quad \sum \alpha_i y_i = 0 \\
(2) \quad 0 \leq \alpha_i \leq C \text{ for all } \alpha_i
\]

- Neither slack variables \( \xi_i \) nor their Lagrange multipliers appear in the dual problem!

- Again, \( x_i \) with non-zero \( \alpha_i \) will be support vectors.
  - If \( 0 < \alpha_i < C \), \( \xi_i = 0 \)
  - If \( \alpha_i = C \), \( \xi_i > 0 \)

- Solution to the problem is:

\[
w = \sum \alpha_i y_i x_i \\
b = y_k - w^T x_k \text{ for any } x_k \text{ such that } 0 < \alpha_k < C
\]

\( w \) is not needed explicitly for classification!

\[
f(x) = \sum \alpha_i y_i x_i^T x + b
\]
A Different View of Soft Margin SVM

- Hinge loss with regularization terms
  \[ \Phi(w) = \frac{1}{2} w^T w + C \sum \xi_i \]
  \[ = \frac{1}{2} w^T w + C \sum \max(0, 1 - y_i (w^T x_i + b)) \]

L2 regularization  Hinge loss

0-1 Loss:
\[ L(y, \hat{y}) = 1[\hat{y} \neq y] \]

Hinge loss:
\[ L(y, \hat{y}) = \max(0, 1 - \hat{y}y) \]
Classification with SVMs

- Given a new point $x$, we can score its projection onto the hyperplane normal:
  - I.e., compute score: $w^T x + b = \sum \alpha_i y_i x_i^T x + b$
  - Decide class based on whether $<$ or $> 0$

- Can set confidence threshold $t$.

Score $> t$: yes
Score $< -t$: no
Else: don’t know
Linear SVMs: Summary

• The classifier is a *separating hyperplane*.

• The most “important” training points are the support vectors; they define the hyperplane.

• Quadratic optimization algorithms can identify which training points $x_i$ are support vectors with non-zero Lagrangian multipliers $\alpha_i$.

• Both in the dual formulation of the problem and in the solution, training points appear only inside inner products:

  \[
  f(x) = \sum \alpha_i y_i x_i^T x + b
  \]

  Find $\alpha_1...\alpha_N$ such that

  $Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j x_i^T x_j$ is maximized and

  (1) $\sum \alpha_i y_i = 0$

  (2) $0 \leq \alpha_i \leq C$ for all $\alpha_i$
Support Vector Machine

• Introduction
• Linear SVM
• Non-linear SVM
• Scalability Issues*
• Summary
Non-linear SVMs

• Datasets that are linearly separable (with some noise) work out great:

• But what are we going to do if the dataset is just too hard?

• How about ... mapping data to a higher-dimensional space:
Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:

\[ \Phi: x \rightarrow \varphi(x) \]
The “Kernel Trick”

• The linear classifier relies on an inner product between vectors \( K(x_i, x_j) = x_i^T x_j \)

• If every data point is mapped into high-dimensional space via some transformation \( \Phi: x \rightarrow \phi(x) \), the inner product becomes:

\[
K(x_i, x_j) = \phi(x_i)^T \phi(x_j)
\]

• A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
Example

• 2-dimensional vectors $\mathbf{x} = [x_1 \ x_2]$, let

$K(x_i, x_j) = (1 + x_i^T x_j)^2$

• show that $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$:

$K(x_i, x_j) = (1 + x_i^T x_j)^2 = 1 + x_{i1}^2 x_{j1}^2 + 2 x_{i1} x_{i2} x_{j1} x_{j2} + x_{i2}^2 x_{j2}^2 + 2 x_{i1} x_{j1} + 2 x_{i2} x_{j2} =$

$= [1 \ x_{i1}^2 \ \sqrt{2} \ x_{i1} x_{i2} \ x_{i2}^2 \ \sqrt{2} x_{i1} \ \sqrt{2} x_{i2}]^T [1 \ x_{j1}^2 \ \sqrt{2} \ x_{j1} x_{j2} \ x_{j2}^2 \ \sqrt{2} x_{j1} \ \sqrt{2} x_{j2}]$

$= \phi(x_i)^T \phi(x_j)$

where $\phi(x) = [1 \ x_1^2 \ \sqrt{2} \ x_1 x_2 \ x_2^2 \ \sqrt{2} x_1 \ \sqrt{2} x_2]$
SVM: Different Kernel functions

- Instead of computing the dot product on the transformed data, it is math. equivalent to applying a kernel function $K(X_i, X_j)$ to the original data, i.e., $K(X_i, X_j) = \Phi(X_i)^T\Phi(X_j)$

- Typical Kernel Functions

  - Polynomial kernel of degree $h$:
    \[
    K(X_i, X_j) = (X_i \cdot X_j + 1)^h
    \]

  - Gaussian radial basis function kernel:
    \[
    K(X_i, X_j) = e^{-\|X_i - X_j\|^2 / 2\sigma^2}
    \]

  - Sigmoid kernel:
    \[
    K(X_i, X_j) = \tanh(\kappa X_i \cdot X_j - \delta)
    \]

- SVM can also be used for classifying multiple (> 2) classes and for regression analysis (with additional parameters)
Non-linear SVM

- Replace inner-product with kernel functions
  - Optimization problem

Find $\alpha_1...\alpha_N$ such that

$$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

is maximized and

1. $\sum \alpha_i y_i = 0$
2. $0 \leq \alpha_i \leq C$ for all $\alpha_i$

- Decision boundary

$$f(x) = \sum \alpha_i y_i K(x_i, x) + b$$
Support Vector Machine

• Introduction
• Linear SVM
• Non-linear SVM
• Scalability Issues*
• Summary
Scaling SVM by Hierarchical Micro-Clustering

- SVM is not scalable to the number of data objects in terms of training time and memory usage
- H. Yu, J. Yang, and J. Han, “Classifying Large Data Sets Using SVM with Hierarchical Clusters”, KDD'03

CB-SVM (Clustering-Based SVM)
- Given limited amount of system resources (e.g., memory), maximize the SVM performance in terms of accuracy and the training speed
- Use micro-clustering to effectively reduce the number of points to be considered
- At deriving support vectors, de-cluster micro-clusters near “candidate vector” to ensure high classification accuracy
**CF-Tree: Hierarchical Micro-cluster**

- Read the data set once, construct a statistical summary of the data (i.e., hierarchical clusters) given a limited amount of memory
- Micro-clustering: Hierarchical indexing structure
  - provide finer samples closer to the boundary and coarser samples farther from the boundary
*Selective Declustering: Ensure High Accuracy*

- CF tree is a suitable base structure for selective declustering
- De-cluster only the cluster $E_i$ such that
  - $D_i - R_i < D_s$, where $D_i$ is the distance from the boundary to the center point of $E_i$ and $R_i$ is the radius of $E_i$
  - Decluster only the cluster whose subclusters have possibilities to be the support cluster of the boundary
    - “Support cluster”: The cluster whose centroid is a support vector
**CB-SVM Algorithm: Outline**

- Construct two CF-trees from positive and negative data sets independently
  - Need one scan of the data set
- Train an SVM from the centroids of the root entries
- De-cluster the entries near the boundary into the next level
  - The children entries de-clustered from the parent entries are accumulated into the training set with the non-declustered parent entries
- Train an SVM again from the centroids of the entries in the training set
- Repeat until nothing is accumulated
*Accuracy and Scalability on Synthetic Dataset*

- Experiments on large synthetic data sets show better accuracy than random sampling approaches and far more scalable than the original SVM algorithm.
Support Vector Machine

• Introduction

• Linear SVM

• Non-linear SVM

• Scalability Issues*

• Summary
Summary

• Support Vector Machine
  • **Linear classifier; support vectors; kernel SVM**
SVM Related Links

- SVM Website: [http://www.kernel-machines.org/](http://www.kernel-machines.org/)
- Representative implementations
  - **LIBSVM**: an efficient implementation of SVM, multi-class classifications, nu-SVM, one-class SVM, including also various interfaces with java, python, etc.
  - **SVM-light**: simpler but performance is not better than LIBSVM, support only binary classification and only in C
  - **SVM-torch**: another recent implementation also written in C
- From classification to regression and ranking:
More about Lagrangian

• Objective with equality constraints

\[
\min_w f(w) \\
\text{s.t.} \\
h_i(w) = 0, \text{for } i = 1,2, \ldots, l
\]

• Lagrangian:

\[ L(w, \alpha) = f(w) + \sum_i \alpha_i h_i(w) \]

• \( \alpha_i \): Lagrangian multipliers

• Solution: setting the derivatives of Lagrangian to be 0

\[
\frac{\partial L}{\partial w} = 0 \text{ and } \frac{\partial L}{\partial \alpha_i} = 0 \text{ for every } i
\]
Generalized Lagrangian

• Objective with both equality and inequality constraints

\[
\min_{\mathbf{w}} f(\mathbf{w}) \\
\text{s.t.} \\
h_i(\mathbf{w}) = 0, \text{ for } i = 1,2, \ldots, l \\
g_j(\mathbf{w}) \leq 0, \text{ for } j = 1,2, \ldots, k
\]

• Lagrangian
  
• \[ L(\mathbf{w}, \alpha, \beta) = f(\mathbf{w}) + \sum_i \alpha_i h_i(\mathbf{w}) + \sum_j \beta_j g_j(\mathbf{w}) \]
  
• \( \alpha_i \): Lagrangian multipliers
  
• \( \beta_j \geq 0 \): Lagrangian multipliers
Why It Works

- Consider function
  \[ \theta_p (w) = \max_{\alpha, \beta: \beta_j \geq 0} L(w, \alpha, \beta) \]

- Therefore, minimize \( f(w) \) with constraints is equivalent to minimize \( \theta_p (w) \)
Lagrange Duality

• The primal problem

\[ p^* = \min_w \max_{\alpha, \beta: \beta_j \geq 0} L(w, \alpha, \beta) \]

• The dual problem

\[ d^* = \max_{\alpha, \beta: \beta_j \geq 0} \min_w L(w, \alpha, \beta) \]

• According to max-min inequality

\[ p^* \leq d^* \]

• When does equation hold?
Primal = Dual

• $p^* = d^*$, under some proper condition (Slater conditions)
  • $f, g_j$ convex, $h_i$ affine
  • Exists $w$, such that all $g_j(w) < 0$

• $(w^*, \alpha^*, \beta^*)$ need to satisfy KKT conditions
  • $\frac{\partial L}{\partial w} = 0$
  • $\beta_j g_j(w) = 0$
  • $h_i(w) = 0, g_j(w) \leq 0, \beta_j \geq 0$