Support Vector Machine and Convex Optimization

Ian En-Hsu Yen
Overview

• **Support Vector Machine**
  – The Art of Modeling --- Large Margin and Kernel Trick
  – Convex Analysis
  – Optimality Conditions
  – Duality

• **Optimization for Machine Learning**
  – Dual Coordinate Descent (fast convergence, moderate cost)
    • libLinear (Stochastic)
    • libSVM (Greedy)
  – Primal Methods
    • Non-smooth Loss ➔ Stochastic Gradient Descent (slow convergence, cheap iter.)
    • Differentiable Loss ➔ Quasi-Newton Method (very fast convergence, expensive iter.)
  – Demo
A Learning/Prediction Game

- Your team members suggest a Hypothesis Space: \{h1, h2 \ldots \}

- You can only request one sample.

- Finding a hypothesis with accuracy > 50%, you earn $100,000. wrong hypothesis (acc \leq 50%) get $100,000 punishment.
H={ h1 }, h1: (A+B) mod 13 = C
\( H = \{ h_1, h_2 \} \), \( h_1: (A+B) \mod 13 = C \), \( h_2: (A-B) \mod 13 = C \)
Large $|H|$ with Small $|Data|$ Guarantees Nothing

• First case: only one hypothesis $h_1$
  
  \[- \Pr \{ |\text{Train\_Error} - \text{Test\_Error}| \geq 50\% \} \leq 1/2 . \]

• Second case: two hypotheses $h_1$, $h_2$
  
  \[- \Pr\{ |\text{Train\_Error} - \text{Test\_Error}| \geq 50\% \text{ for } h_1 \text{ or } h_2 \} \leq 1/2 + 1/2 = 1. \]

\[ \Rightarrow \text{Guarantee Nothing.} \]
Why Support Vector Machine (SVM) ?

- Flexible Hypothesis Space. (Non-linear Kernel)
- Not to Overfit (Large-Margin)
- Sparsity (Support Vectors)
- Easy to find Global Optimum (Convex Problem)
Why Support Vector Machine (SVM) ?

• Flexible Hypothesis Space. (Non-linear Kernel)
• Not to Overfit (Large-Margin)
• Sparsity (Support Vectors)
• Easy to find Global Optimum (Convex Problem)
SVM: Large-Margin Perceptron

\[ w^* = \arg \max_w \min_n y_n \left( \frac{w^T x_n}{\|w\|} \right) \]

\[ \max_{\omega, \rho} \rho \]

\[ \text{s.t. } y_n \left( \frac{w^T x_n}{\|w\|} \right) \geq \rho, \ \forall n \]

\[ \min_n y_n \left( \frac{w^T x_n}{\|w\|} \right) = \rho \]

Choose \( \rho \) such that:

\[ \max_w \frac{1}{\|w\|} \]

\[ \text{s.t. } y_n (w^T x_n) \geq 1, \ \forall n \]

\[ \min_w \|w\| \]

\[ \text{s.t. } y_n (w^T x_n) \geq 1, \ \forall n \]

\[ \min_w \|w\|^2 \]

\[ \text{s.t. } y_n (w^T x_n) \geq 1, \ \forall n \]
SVM: Large-Margin Perceptron

Hard Margin

\[
\min_w \frac{1}{2} \|w\|^2 \\
\text{s.t. } y_n(w^T x_n) \geq 1, \forall n
\]

Non-Separable Problem

Soft Margin

\[
\min_{w, \xi \geq 0} \frac{1}{2} \|w\|^2 + C \sum_n \xi_n \\
\text{s.t. } y_n(w^T x_n) \geq 1 - \xi_n, \forall n
\]

A drawback of SVM: Solution sensitive to C

Small C

Large C
From Linear to Non-Linear

\[ \Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3 \]

\[(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)\]

Perceptron: \( a \, x_1 + b \, x_2 = 0 \)

Ellipse: \( a \, x_1^2 + b \, x_2^2 + c \, x_1x_2 = 0 \) (center at origin)
From Linear to Non-Linear

Linear SVM:

\[
\begin{align*}
\min_{w, \xi \geq 0} & \frac{1}{2} \|w\|^2 + C \sum_n \xi_n \\
\text{s.t.} & \quad y_n (w^T x_n) \geq 1 - \xi_n, \quad \forall n
\end{align*}
\]

Non-linear SVM:

\[
\begin{align*}
\min_{w, \xi \geq 0} & \frac{1}{2} \|w\|^2 + C \sum_n \xi_n \\
\text{s.t.} & \quad y_n (w^T \phi(x_n)) \geq 1 - \xi_n, \quad \forall n
\end{align*}
\]

\[
x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \rightarrow \quad \phi(x_n) = \begin{bmatrix} x_1^2 \\ x_1 x_2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \\ \sqrt{2} x_2 x_3 \\ x_3^2 \\ \sqrt{2} x_1 x_3 \end{bmatrix}
\]
SVM: Kernel Trick

Feature Expansion
\[ x \rightarrow \phi(x) \]
\[ x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \rightarrow \phi(x_n) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_3^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_2x_3 \\ \sqrt{2}x_1x_3 \end{bmatrix} \]

3 features \( \rightarrow \) \( 3 + C_3^2 = 6 \)
100 features \( \rightarrow \) \( 100 + C_{100}^0 = 5050 \)

Deg-2 Feature Expansion \( \rightarrow \) \( O(D^2) \)
Deg-K Feature Expansion \( \rightarrow \) \( O(D^K) \)

Dot Product can be computed efficiently:
\[
\phi(x)^T \phi(z) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_3^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_2x_3 \\ \sqrt{2}x_1x_3 \end{bmatrix}^T \begin{bmatrix} z_1^2 \\ z_2^2 \\ z_3^2 \\ \sqrt{2}z_1z_2 \\ \sqrt{2}z_2z_3 \\ \sqrt{2}z_1z_3 \end{bmatrix} = x_1^2z_1^2 + x_2^2z_2^2 + x_3^2z_3^2 + 2(x_1x_2z_1z_2 + x_2x_3z_2z_3 + x_1x_3z_1z_3) = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}^T \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}^2 = (x^T z)^2 \]

\( x^T z \)

Compute dot Product using \( K(x,z)=(x^Tz)^2 \)
\( \leftrightarrow \) deg-2 feature expansion

\( O(D^K) \)

O(D)
SVM: Kernel Trick

Feature Expansion

\[ x \rightarrow \phi(x) \]

\[
\begin{align*}
\min_{w, \xi \geq 0} & \quad \frac{1}{2} \|w\|^2 + C \sum_n \xi_n \\
\text{s.t.} & \quad y_n w^T \phi(x_n) \geq 1 - \xi_n, \quad \forall n
\end{align*}
\]

Can we formulate the problem only using dot product \( \phi(x_i)^T \phi(x_j) \)?

By **Representer Theorem**, solution \( w^* \) of the problem can be expressed as **linear combination of instances**:

\[
w^* = \sum_n \alpha_n y_n \phi(x_n) = [y_1 \phi(x_1) \ldots y_N \phi(x_N)]_{D \times N} \left[ \begin{array}{c} \alpha_1 \\ \vdots \\ \alpha_N \end{array} \right] = \Phi \alpha
\]

Prediction using only dot product \( \phi(x_i)^T \phi(x_j) \):

\[
w^T \phi(x_t) = (\sum_n \alpha_n y_n \phi(x_n))^T \phi(x_t)
\]

\[
= \sum_n \alpha_n y_n \phi(x_n)^T \phi(x_t) = \sum_n \alpha_n y_n K(x_n, x_t)
\]

\( O(N^2) \)

or \( O(|\text{Support Vector}| \times D) \)
SVM: Kernel Trick

Some popular Kernels:

- Polynomial Kernel: \( K(x, x') = (x^T x' + 1)^d \)
- RBF Kernel: \( K(x, x') = \exp(-\gamma \| x - x' \|^2) \)
- Linear Kernel: \( K(x, x') = x^T x' \)

Kernels may be easier to define than Features:

- String Kernel: Gene Classification / Rewriting or not
- Tree Kernel: Syntactic parse tree classification
- Graph Kernel: Graph Type Classification
Overview

• Support Vector Machine
  – The Art of Modeling --- Large Margin and Kernel Trick
  – Convex Analysis
  – Optimality Conditions
  – Duality

• Optimization for Machine Learning
  – Dual Coordinate Descent ( fast convergence, moderate cost )
    • libLinear (Stochastic)
    • libSVM (Greedy)
  – Primal Methods
    • Non-smooth Loss ➔ Stochastic Gradient Descent ( slow convergence, cheap iter. )
    • Differentiable Loss ➔ Quasi-Newton Method ( very fast convergence, expensive iter. )
Convex Analysis

General Optimization Problem:

\[
\begin{align*}
\text{minimize} & \quad f_0(x) \\
\text{subject to} & \quad f_i(x) \leq b_i, \quad i = 1, \ldots, m
\end{align*}
\]

is very difficult to solve. (very long time vs. approximate)

Optimization is much easier if the problem is convex, that is:

1. The **objective** function is convex:
   \[
   f_0(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y) \quad \text{for} \quad 0 \leq \alpha \leq 1
   \]

2. The **feasible domain** (constrained space) is convex:
   \[
   \text{if} \quad x \in C, \quad y \in C \quad \Rightarrow \quad \alpha x + (1-\alpha)y \in C, \quad 0 \leq \alpha \leq 1
   \]

⇒ All local minimum is global minimum !!
Convex Analysis

Simple Example:

\[ x \leq a \quad x \geq b \]

Non-Convex Set ;
Convex function

Convex Set ;
Non-Convex function

Convex Set ;
Convex function
If $x^*$ is a **local minimum**, there is a “ball”, in which any feasible $x'$ has $f(x') \geq f(x^*)$. 

Convex Analysis

$x^*$ is local minimum $\Rightarrow$ $x^*$ is global minimum (why?)
If $x^*$ is a **local minimum**, there is a “ball”, in which any feasible $x'$ has $f(x') \geq f(x^*)$.

Assume for contradiction that $x^*$ is **not a global minimum**. There should be a feasible $x'$ with $f(x') < f(x^*)$. 
Convex Analysis

If x* is a **local minimum**, there is a “ball”, in which any feasible x’ has f(x’) ≥ f(x*).

Assume for contradiction that x* is **not a global minimum**. There should be a feasible x’ with f(x’) < f(x*).

Then we can find a feasible αx’+(1-α)x* in the **ball** with:

\[
f(\alpha x’ +(1- \alpha)x* ) \leq \alpha f(x’)+(1-\alpha)f(x*)
\]
\[
< f(x*)
\]

⇒ contradiction.
Convex Analysis

Example of Convex Set:  
if $\mathbf{x} \in C$, $\mathbf{y} \in C$  \Rightarrow \alpha \mathbf{x} + (1 - \alpha) \mathbf{y} \in C, \quad 0 \leq \alpha \leq 1$

Linear equality constraint (Hyperplane)  
\{x \mid a^T x = b\} (a \neq 0)

Linear inequality constraint (Halfspace)  
\{x \mid a^T x \leq b\} (a \neq 0)
Convex Analysis

Example of Convex Set:  \( \text{if } x \in C, y \in C \implies \alpha x + (1-\alpha)y \in C, \quad 0 \leq \alpha \leq 1 \)

Intersection of Convex Set:

\[
\begin{align*}
  a_1 x & \leq b_1 \\
  a_2 x & \leq b_2 \\
  a_3 x & \leq b_3 \quad (Ax \leq b, \ Cx = d) \\
  c_4 x & = d_4 \\
  c_5 x & = d_5
\end{align*}
\]

\( x, y \in A \cap B, \quad A, B \text{ is convex} \)

\( \alpha x + (1-\alpha)y \in A \cap B \) ?
Convex Analysis

Example of Convex Function: \( f_0(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) \) for \( 0 \leq \alpha \leq 1 \)

Linear Function \[ f(x) = c^T x \]

Quadratic Function \[ f(x) = \frac{1}{2} x^T Q x + c^T x \]

Obviously, it depends ......

\[ ax^2 + bx + c, \quad a > 0 \]
\[ ax^2 + bx + c, \quad a < 0 \]
Convex Analysis

Example of Convex Function: \( f_0(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) \) for \( 0 \leq \alpha \leq 1 \)

Linear Function \( f(x) = c^T x \)

Quadratic Function \( f(x) = \frac{1}{2} x^T Q x + c^T x \)

A practical way to check convexity:
Check the second derivative \( \frac{\partial^2 f(x)}{\partial x^2} \geq 0 \) at \( \forall x \)

\[ \begin{align*}
\text{ax}^2 + bx + c, \quad a > 0 \\
\text{ax}^2 + bx + c, \quad a < 0
\end{align*} \]
Convex Analysis

Example of Convex Function: 
\[ f_0(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) \quad \text{for} \quad 0 \leq \alpha \leq 1 \]

Linear Function 
\[ f(x) = c^T x \]

Quadratic Function 
\[ f(x) = \frac{1}{2} x^T Q x + c^T x \]

In \( \mathbb{R}^D \), we have convexity if the **Hessian Matrix**:

\[
\frac{\partial^2 f(x)}{\partial x^2} = H(x) = \begin{bmatrix}
\frac{\partial^2 f(x)}{\partial x_1^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 x_D} \\
\vdots & \ddots & \vdots \\
\frac{\partial^2 f(x)}{\partial x_D x_1} & \cdots & \frac{\partial^2 f(x)}{\partial x_D^2}
\end{bmatrix}
\]

is positive semidefinite at \( \forall x \) 
\[
( z^T H(x) z \geq 0 \quad \text{for} \quad \forall z )
\]
\[
( \text{all eigenvalue} \geq 0 )
\]
In $\mathbb{R}^D$, we have convexity if the **Hessian Matrix**:

$$
\frac{\partial^2 f(x)}{\partial x^2} = H(x) = \begin{bmatrix}
\frac{\partial^2 f(x)}{\partial x_1^2} & \ldots & \frac{\partial^2 f(x)}{\partial x_1 x_D} \\
\vdots & \ddots & \vdots \\
\frac{\partial^2 f(x)}{\partial x_D x_1} & \ldots & \frac{\partial^2 f(x)}{\partial x_D^2}
\end{bmatrix}
$$

is positive (semi-) definite at $\forall x$

- $z^T H(x) z \geq 0$ for $\forall z$
- All eigenvalue $\geq 0$
Convex Analysis

In \( \mathbb{R}^D \), we have convexity if the **Hessian Matrix**:

\[
\frac{\partial^2 f(x)}{\partial x^2} = H(x) = \begin{bmatrix}
\frac{\partial^2 f(x)}{\partial x_1^2} & \ldots & \frac{\partial^2 f(x)}{\partial x_1 x_D} \\
\vdots & \ddots & \vdots \\
\frac{\partial^2 f(x)}{\partial x_D x_1} & \ldots & \frac{\partial^2 f(x)}{\partial x_D^2}
\end{bmatrix}
\]

is positive (semi-)definite at \( \forall x \)

\( z^T H(x) z \geq 0 \, \text{for} \, \forall z \)

( all eigenvalue \( \geq 0 \) )

**Other Cases?**

- \( H = 0 \) is positive (semi-)definite
- is negative (semi-)definite

\( H \) is positive definite

\( H \) is negative definite

\( H = O \)
Convex Analysis

Example of Convex Function: \( f_0(\alpha x + (1 - \alpha) y) \leq \alpha f(x) + (1 - \alpha) f(y) \) for \( 0 \leq \alpha \leq 1 \)

Linear Function \( f(x) = c^T x \quad \left( \frac{\partial^2 f(x)}{\partial x^2} = H(x) = 0 \right) \)

Quadratic Function \( f(x) = \frac{1}{2} x^T Q x + c^T x \quad \left( \frac{\partial^2 f(x)}{\partial x^2} = H(x) = Q \right) \)

Quadratic Function is convex if \( Q \) is **positive semi-definite**.

\[
\min_w \frac{1}{2} \|w\|^2 \\
\text{s.t.} \quad y_n w^T \phi(x_n) \geq 1, \quad \forall n
\]

\[
\frac{1}{2} \|w\|^2 = \frac{1}{2} w^T I w \quad \text{Is } I \text{ positive-semidefinite?}
\]
Convex Analysis

Example of Convex Function: \[ f_0(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) \text{ for } 0 \leq \alpha \leq 1 \]

Linear Function \[ f(x) = c^T x \quad \left( \frac{\partial^2 f(x)}{\partial x^2} = H(x) = 0 \right) \]

Quadratic Function \[ f(x) = \frac{1}{2} x^T Q x + c^T x \quad \left( \frac{\partial^2 f(x)}{\partial x^2} = H(x) = Q \right) \]

Quadratic Function is convex if \( Q \) is positive semi-definite.

\[
\begin{align*}
\min_{w, \xi \geq 0} & \quad \frac{1}{2} \|w\|^2 + C \sum_n \xi_n \\
\text{s.t.} & \quad y_n w^T \phi(x_n) \geq 1 - \xi_n, \forall n
\end{align*}
\]  

Is \( H \left( \begin{bmatrix} w \\ \xi \end{bmatrix} \right) = \begin{bmatrix} I_{D*D} & O_{D*N} \\ O_{D*N} & O_{N*N} \end{bmatrix} \) positive-semidefinite?  

Half-space constraint SVM problem is a convex problem. (Quadratic Program)
Convex Analysis

Example of Convex Problem:

Linear Programming:

\[
\begin{align*}
\text{minimize} & \quad c^T x + d \\ 
\text{subject to} & \quad Gx \leq h \\ 
& \quad Ax = b
\end{align*}
\]

Linear objective function
s.t. Linear Constraint.

Quadratic Programming:

\[
\begin{align*}
\text{minimize} & \quad (1/2)x^T Px + q^T x + r \\ 
\text{subject to} & \quad Gx \leq h \\ 
& \quad Ax = b
\end{align*}
\]

Quadratic objective function
s.t. Linear Constraint.

where \( P \) must be positive – semidefinite
Overview

• **Support Vector Machine**
  – The Art of Modeling --- Large Margin and Kernel Trick
  – Convex Analysis
  – Optimality Conditions
  – Duality

• **Optimization for Machine Learning**
  – Dual Coordinate Descent ( fast convergence, moderate cost )
    • libLinear (Stochastic)
    • libSVM (Greedy)
  – Primal Methods
    • Non-smooth Loss  ➔  Stochastic Gradient Descent ( slow convergence, cheap iter. )
    • Differentiable Loss  ➔  Quasi-Newton Method ( very fast convergence, expensive iter. )
Optimality Condition

There are many, many different solvers designed for different problem, but they share the same optimality condition.

First, we consider Unconstrained Problem:

\[
\min_x f(x) \quad \text{Example: Matrix Factorization (non-convex)}
\]

\[
\min_{P,Q} \sum (r_{ui} - p_u^T q_i)^2 + \lambda_p \|P\|^2 + \lambda_q \|Q\|^2
\]
Optimality Condition

There are many, many different solvers designed for different problem, but they share the same optimality condition.

First, we consider Unconstrained Problem:

\[ \min_x f(x) \]

\( x^* \) is Local minimizer \( \iff f(x^*) \leq f(x^* + p) \) for all \( p \) with \( \|p\| < \varepsilon \)

For twice-differenciable \( f(x) \), consider the Taylor Expansion:

\[ f(x^* + p) = f(x^*) + \nabla f(x^*)^T p + \frac{1}{2} p^T \nabla^2 f(x^*) p + \ldots \]

\( x^* \) is Local minimizer \( \implies \nabla f(x^*) = 0 \)

\( \nabla f(x^*) = 0 \implies x^* \) is Local minimizer ?
Optimality Condition

There are many, many different solvers designed for different problem, but they share the same optimality condition.

First, we consider Unconstrained Problem:

\[
\min_{x} f(x)
\]

\(x^*\) is Local minimizer \(\iff f(x^*) \leq f(x^* + p)\) for all \(p\) with \(\|p\| < \varepsilon\)

For twice-differenciable \(f(x)\), consider the Taylor Expansion:

\[
f(x^* + p) = f(x^*) + \nabla f(x^*)^T p + \frac{1}{2} p^T \nabla^2 f(x^*) p + \ldots
\]

\(x^*\) is Local minimizer \(\Rightarrow \nabla f(x^*) = 0\)

\(\nabla f(x^*) = 0\) \(\Rightarrow \) \(x^*\) is Local minimizer

\(\nabla^2 f(x^*)\) is positive-semidefinite

No need to check for Convex function (why?)
Optimality Condition

There are many, many different solvers designed for different problem, but they share the same optimality condition.

First, we consider Unconstrained Problem:

\[
\min_x f(x)
\]

\(x^*\) is Local minimizer \(\iff\) \(f(x^*) \leq f(x^* + p)\) for all \(p\) with \(\|p\| < \varepsilon\)

For twice-differenciable \(f(x)\), consider the Taylor Expansion:

\[
f(x^* + p) = f(x^*) + \nabla f(x^*)^T p + \frac{1}{2} p^T \nabla^2 f(x^*) p + \ldots
\]

For Convex \(f(x)\):

\(x^*\) is Global minimizer \(\iff\) \(\nabla f(x^*) = 0\)

Assume convexity for now on......
Optimality Condition

There are many, many different solvers designed for different problem, but they share the same optimality condition.

Now, consider **Equality Constrained Problem**: 

$$\min_x f(x)$$

s.t. $Ax = b$ (ex. $a_1^T x = b$) 

(nonslinear equality is, in general, not convex)

$x^*$ is Local minimizer $\iff f(x^*) \leq f(x^* + p)$ for all "feasible" $p$ with $\|p\| < \varepsilon$

$\iff f(x^*) \leq f(x^* + Zq)$ for all $q$ with $\|q\| < \varepsilon$

For twice-differenciable $f(x)$, consider the **Taylor Expansion**:

$$f(x^* + Z^T q) = f(x^*) + (Z^T \nabla f(x^*) )^T q + \frac{1}{2} q^T Z^T \nabla^2 f(x^*) Z q + ...$$

$f(x)$ is convex, $x^*$ is Glocal minimizer $\iff Z^T \nabla f(x^*) = 0$

$\iff -\nabla f(x^*) = A^T \lambda$ ( $\nabla f(x^*)$ in $\text{Row}(A)$ )

**Row(A)**: $A^T \lambda = \lambda_1 \bar{a}_1 + \lambda_2 \bar{a}_2 + ... + \lambda_n \bar{a}_n$

**Null(A)**: $Zq$, where $Z_{d^*\langle d-n \rangle} = [z_1, ..., z_{\langle d-n \rangle}]$
Optimality Condition

There are many, many different solvers designed for different problem, but they share the same optimality condition.

Now, consider Equality Constrained Problem:

\[
\begin{align*}
\min_x & \quad f(x) \\
\text{s.t.} & \quad Ax = b \quad \text{(ex. } a_1^T x = b) \\
& \quad \text{(nonlinear equality is, in general, not convex)}
\end{align*}
\]

\[x^*\] is Local (Glocal) minimizer \iff \[Z^T \nabla f(x^*) = 0\]

\[\lambda_n > 0 \quad \Rightarrow \quad ?\]
\[\lambda_n < 0 \quad \Rightarrow \quad ?\]
\[\lambda_n = 0 \quad \Rightarrow \quad ?\]
Optimality Condition

There are many, many different solvers designed for different problem, but they share the same optimality condition.

Now, consider Inequality Constrained Problem:

\[
\begin{align*}
\min_x f(x) \\
\text{s.t. } Ax \leq b \\
\end{align*}
\]

where \( Ax \leq b \) (ex. \( a_1^T x \leq b \))

(Assume linear inequality for simplicity.)

Let \( A^* \) (some rows of \( A \)) be the coefficients of binding constraints:

\( x^* \) is Local (Global) minimizer \( \iff -\nabla f(x^*) = A^T \lambda \) (ex. \( -\nabla f(x^*) = \lambda \bar{a}_1 \))

and \( \lambda \geq 0 \) (feasible direction not decrease \( f(x) \))

\[
\begin{align*}
\lambda_n < 0 & \quad \Rightarrow \quad \text{Detach from } a_n x < b \quad \text{can decrease } f(x)
\end{align*}
\]
Optimality Condition

There are many, many different solvers designed for different problems, but they share the same optimality condition.

Now, consider Inequality Constrained Problem:

\[ \min_x f(x) \]
\[ \text{s.t. } Ax \leq b \]

(Assume linear inequality for simplicity.)

Require \( \lambda_n = 0 \) for non-binding constraint:

\[ \lambda_n (a_n x - b_n) = 0, \quad \forall n \]

\( x^* \) is Local (Global) minimizer \( \iff \) \( -\nabla f(x^*) = A^T \lambda \)

(ex. \( -\nabla f(x^*) = \lambda \bar{a}_1 \))

KKT conditions.

\[ \lambda \geq 0 \] (feasible direction not decrease \( f(x) \))
Optimality Condition for SVM

What’s the KKT condition for:

\[
\begin{align*}
\min_{w, \xi} & \quad \frac{1}{2} \|w\|^2 + C \sum_n \xi_n \\
\text{s.t.} & \quad y_n w^T \phi(x_n) \geq 1 - \xi_n, \quad \forall n
\end{align*}
\]

\[
\begin{align*}
-\nabla f(x^*) &= A^T \lambda \\
&\Rightarrow \quad w = \sum_n \alpha_n y_n \phi(x_n)
\end{align*}
\]

\[
\begin{align*}
C &= \alpha_n + \beta_n
\end{align*}
\]

\[
\begin{align*}
\lambda \geq 0 &\Rightarrow \quad \alpha_n \geq 0 \\
\beta_n \geq 0 \\
\lambda_n (\alpha_n x - b_n) &= 0 \\
&\Rightarrow \quad \alpha_n (y_n w^T \phi(x_n) - 1 + \xi_n) \geq 0 \\
&\Rightarrow \quad \beta_n \xi_n \geq 0
\end{align*}
\]
Optimality Condition for SVM

What’s the KKT condition for:

\[
\begin{align*}
\beta_n \rightarrow & \min_{w, \xi \geq 0} \frac{1}{2} \|w\|^2 + C \sum_n \xi_n \\
\alpha_n \rightarrow & \text{s.t. } y_n w^T \phi(x_n) \geq 1 - \xi_n, \forall n
\end{align*}
\]

\[\lambda \geq 0 \Rightarrow \begin{cases} 
\alpha_n \geq 0 \\
(C - \alpha_n) \geq 0 
\end{cases} \]

\[\lambda_n (\alpha_n x - b_n) = 0 \Rightarrow \begin{cases} 
\alpha_n (y_n w^T \phi(x_n) - 1 + \xi_n) \geq 0 \\
(C - \alpha_n) \xi_n \geq 0 
\end{cases} \]

\[-\nabla f(x^*) = A^T \lambda \Rightarrow w = \sum_n \alpha_n y_n \phi(x_n) \]

\[\beta_n = C - \alpha_n \]
Optimality Condition for SVM

What’s the KKT condition for:

$$
\begin{align*}
\min_{\|w\|^2} & \left[ \frac{1}{2} \|w\|^2 + C \sum_n \xi_n \right] \\
\text{s.t.} & \quad y_n w^T \phi(x_n) \geq 1 - \xi_n, \quad \forall n
\end{align*}
$$

$$
\begin{align*}
\lambda \geq 0 & \quad \Rightarrow \quad 0 \leq \alpha_n \leq C \\
\lambda_n (a_n x - b_n) = 0 & \quad \Rightarrow \quad \alpha_n (y_n w^T \phi(x_n) - 1 + \xi_n) = 0 \\
(C - \alpha_n) \xi_n = 0 &
\end{align*}
$$

1. \( w = \sum_n \alpha_n y_n \phi(x_n) \) can be expressed as linear combination of instances.
2. If constraint \( y_n w^T \phi(x_n) \geq 1 - \xi_n \) not binding \( \Rightarrow \alpha_n = 0 \)
3. If \( \alpha_n > 0 \) \( \Rightarrow \) constraint is binding (Support Vectors !)
4. If loss of n-th instance \( \xi_n > 0 \) \( \Rightarrow \alpha_n = C \)
Overview

• **Support Vector Machine**
  – The Art of Modeling --- Large Margin and Kernel Trick
  – Convex Analysis
  – Optimality Conditions
  – Duality

• **Optimization for Machine Learning**
  – Dual Coordinate Descent (fast convergence, moderate cost)
    • libLinear (Stochastic)
    • libSVM (Greedy)
  – **Primal Methods**
    • Non-smooth Loss ➞ Stochastic Gradient Descent (slow convergence, cheap iter.)
    • Differentiable Loss ➞ Quasi-Newton Method (very fast convergence, expensive iter.)
Primal SVM Problem

\[
\min_{w, \xi \geq 0} \frac{1}{2} \|w\|^2 + C \sum_n \xi_n
\]

s.t. \( y_n w^T \phi(x_n) \geq 1 - \xi_n, \ \forall n \)  

(Let D:#feature, N:#samples)

Quadratic Program (QP) with:

- D + N variables
- N Linear constraints
- N nonnegative constraints

⇒ Intractable for median scale
(ex. N=1000, D=1000)
Primal SVM Problem

Constrained Problem $\rightarrow$ Non-smooth Unconstrained

\[
\min_{w, \xi \geq 0} \frac{1}{2}\|w\|^2 + C \sum_n \xi_n \\
\text{s.t. } y_n f(x_n) \geq 1 - \xi_n, \ \forall n
\]

Given $w$, minimize w.r.t. $\xi_n$

$$
\xi_n = \begin{cases} 
0 & \text{if } 1 - y_n f(x_n) \leq 0 \\
1 - y_n f(x_n), & \text{otherwise}
\end{cases}
$$

$$
\xi_n = \max\{ 1 - y_n f(x_n), \ 0 \}
$$

(Hinge-Loss $L(.)$)

(Nonsmooth, Unconstrained)
L2-Regularized Loss Minimization

\[
\min_w \frac{\lambda}{2} \|w\|^2 + \sum_n L(f(x_n), y_n)
\]

Hinge-Loss (L1-SVM, Structural SVM)

L2-SVM

SVR

0/1-Loss (Accuracy)

Logistic Regression (CRF)

(Least-Square) Regression
L2-Regularized Loss Minimization

$$\min_w \frac{\lambda}{2} \|w\|^2 + \sum_n L(f(x_n), y_n)$$

**Convex Loss**
- Solve with Global Minimum

**Hinge-Loss (L1-SVM, Structural SVM)**

**L2-SVM**

**SVR**

**0/1-Loss (Accuracy)**

**Logistic Regression (CRF)**

**(Least-Square) Regression**
L2-Regularized Loss Minimization

\[
\min_w \frac{\lambda}{2} \|w\|^2 + \sum_{n} L(f(x_n), y_n)
\]

**Convex Smooth Loss**
- Applicable for Second-Order Method
- Coordinate Descent (primal)
- Gradient Descent \( \Rightarrow O(\log(1/\varepsilon)) \) rate
- Non-smooth \( \Rightarrow O(1/ \varepsilon) \) rate

**Hinge-Loss** (L1-SVM, Structural SVM)

**L2-SVM**

**SVR**

**0/1-Loss** (Accuracy)

**Logistic Regression** (CRF)

**(Least-Square) Regression**
L2-Regularized Loss Minimization

\[ \min_w \frac{\lambda}{2} \|w\|^2 + \sum_n L(f(x_n), y_n) \]

Dual Sparsity

Hinge-Loss (L1-SVM, Structural SVM)

\[ -y_n f(x_n) \]

0

L2-SVM

\[ -y_n f(x_n) \]

0

SVR

\[ f(x_n) \]

\[ y_n \]

0/1-Loss (Accuracy)

\[ -y_n f(x_n) \]

Logistic Regression (CRF)

\[ -y_n f(x_n) \]

0

(Least-Square) Regression

\[ f(x_n) \]

\[ y_n \]
L2-Regularized Loss Minimization

$$\min_w \frac{\lambda}{2} \|w\|^2 + \sum_n L(f(x_n), y_n)$$

**Noise-Sensitive**

Hinge-Loss (L1-SVM, Structural SVM)

L2-SVM

SVR

0/1-Loss (Accuracy)

Logistic Regression (CRF)

(Least-Square) Regression

Most insensitive
Stochastic (sub-)Gradient Descent
(S. Shalev-Shwartz et al., ICML 2007)

\[ \min_w \frac{\lambda}{2} \|w\|^2 + \frac{1}{N} \sum_n L(f(x_n), y_n) \]

### Algorithm: Subgradient Descent

For \( t = 1 \ldots T \)

\[
    w^{(t+1)} = w^{(t)} - \eta_t \left( \lambda w^{(t)} + \frac{1}{N} \sum_n L'(n)\phi_n \right)
\]

End

A common choice: \( \eta_t = \frac{1}{t} \)

### Algorithm: Stochastic Subgradient Descent

For \( t = 1 \ldots T \)

Draw \( \tilde{n} \) from uniformly from \( \{1 \ldots N\} \)

\[
    w^{(t+1)} = w^{(t)} - \eta_t \left( \lambda w^{(t)} + L'(\tilde{n})\phi_{\tilde{n}} \right)
\]

End

\[
    \bar{w}^{(k)} := \frac{2}{k(k+1)} \sum_{t=1}^{k} t w^{(t)}
\]

(avg. over iterations \( \Rightarrow \) much faster)

Hinge-Loss (L1-SVM, Structural SVM)

sub-gradient at \( y_n f(x_n) = 1 \)

(Shamir, O. and Zhang, ICML, 2013)
Stochastic (sub-)Gradient Descent
(S. Shalev-Shwartz etal., ICML 2007)

\[
\min_w \frac{1}{2}\|w\|^2 + C \sum_n L(f(x_n), y_n)
\]

SGD
- Applicable to all.
- Non-Smooth $\Rightarrow$ GD:$O(1/\varepsilon)$, SGD:$O(1/\varepsilon)$
- Smooth $\Rightarrow$ GD:$O(\log 1/\varepsilon)$, SGD:$O(1/\varepsilon)$

Hinge-Loss (L1-SVM, Structural SVM)

L2-SVM

SVR

0/1-Loss (Accuracy)

Logistic Regression (CRF)

(Least-Square) Regression
SGD (Pegasos) vs. Batch Method (LibLinear)

• Cons: SGD converges very slowly. (sometimes seems not convergent....)

• Pros: SGD (online method) has same convergence rate for Testing and Training.

• Do you care a “Ratio Improvement” or “Absolute Improvement” in Testing?
• What’s your evaluation measure? (AUC, Prec/Recall, Accuracy....)
• ill-conditioned problems (pos/neg ratio, Large C)

(Heish, ICML 2008) (LibLinear)

Bottou, Léon. 2007. Learning with Large Scale Datasets. NIPS Tutorial.
Overview

• **Support Vector Machine**
  – The Art of Modeling --- Large Margin and Kernel Trick
  – Convex Analysis
  – Optimality Conditions
  – Duality

• **Optimization for Machine Learning**
  – Dual Coordinate Descent (fast convergence, moderate cost)
    • libLinear (Stochastic)
    • libSVM (Greedy)
  – Primal Methods
    • Non-smooth Loss ➔ Stochastic Gradient Descent (slow convergence, cheap iter.)
    • **Differentiable Loss ➔ Quasi-Newton Method (very fast convergence, expensive iter.)**
  – L1-regularized
    • Primal Coordinate Descent
Smooth Loss vs. Non-smooth Loss

$$\min_w \frac{\lambda}{2} \|w\|^2 + \sum_n L(f(x_n), y_n)$$

L1-SVM: \( \max(1-y_nf(x_n), 0) \)

L2-SVM: \( \max(1-y_nf(x_n), 0)^2 \)

➡️ Unconstrained Differentiable Problem.

Usage: train [options] training_set_file [model_file]

Options:

-s type : set type of solver (default 1)

for multi-class classification

0 -- L2-regularized logistic regression (primal)
1 -- L2-regularized L2-loss support vector classification (dual)
2 -- L2-regularized L2-loss support vector classification (primal)
3 -- L2-regularized L1-loss support vector classification (dual)
4 -- support vector classification by Crammer and Singer
5 -- L1-regularized L2-loss support vector classification
6 -- L1-regularized logistic regression
7 -- L2-regularized logistic regression (dual)

for regression

11 -- L2-regularized L2-loss support vector regression (primal)
12 -- L2-regularized L2-loss support vector regression (dual)
13 -- L2-regularized L1-loss support vector regression (dual)
Primal Quasi-Newton Method

\[ \min_w \frac{1}{2} \|w\|^2 + C \sum_n L(f(x_n), y_n) \]

- Gradient Descent (1st order) uses Linear Approximation by \( \nabla f(w) = g \)
- Newton Method (2nd order) uses Quadratic Approximation by \( \nabla f(w) = g \) and \( \nabla^2 f(w) = H \)
Primal Quasi-Newton Method

\[
\min_w f(w) = \frac{1}{2}\|w\|^2 + C \sum_n L(w^T x_n, y_n)
\]

\[
g = \nabla f(w) = w + C \sum_n L'(n) x_n
\]

\[
H = \nabla^2 f(w) = I + C \sum_n L''(n) x_n x_n^T
\]

Quadratic Approximation at \(w^{(t)}\):

\[
\min_{s = w - w^{(t)}} \frac{1}{2} s^T Hs + g^T s + f(w^{(t)})
\]

Minimum at \(s^*\):

\[
Hs^* = -g
\]

iteration cost: \(O(N*D^2 + D^3)\)

Algorithm: Newton Method

For \(t = 1...T\)

\[
\text{Solve } H^{(t)} s = -g^{(t)}
\]

\[
w^{(t+1)} = w^{(t)} + \eta_t s^*
\]

End
Primal Quasi-Newton Method

\[
\min_w f(w) = \frac{1}{2} \|w\|^2 + C \sum_n L(w^T x_n, y_n)
\]

\[
g = \nabla f(w) = w + C \sum_n L'(n) x_n
\]

\[
H = \nabla^2 f(w) = I + C \sum_n L''(n) x_n x_n^T
\]

Quadratic Approximation at \( w(t) \):
\[
\min_s \frac{1}{2} s^T H s + g^T s + f(w(t))
\]
Minimum at \( s^* \):
\[
H s^* + g = 0
\]

Iteration cost:
\[
O( N*D + |SV|*D^*_|T_{inner}|)
\]

Algorithm: Conjugate Gradient for \( Ax = b \).
For \( t = 1 \ldots T_{inner} \)
\[
r^{(t)} = b - Ax^{(t)}
\]
\[
d^{(t+1)} = d^{(t)} + \eta_t r^{(t)}
\]
\[
x^{(t+1)} = x^{(t)} - \eta'_t d^{(t)}
\]
End

Algorithm: Quasi-Newton Method
For \( t = 1 \ldots T \)
\[
\text{Solve } H^{(t)} s = -g^{(t)} \text{ approximately.}
\]
\[
w^{(t+1)} = w^{(t)} + \eta_t s^*
\]
End
Overview

• **Support Vector Machine**
  – The Art of Modeling --- Large Margin and Kernel Trick
  – Convex Analysis
  – Optimality Conditions
  – Duality

• **Optimization for Machine Learning**
  – Dual Coordinate Descent ( fast convergence, moderate cost )
    • libLinear (Stochastic)
    • libSVM (Greedy)
  – Primal Methods
    • Non-smooth Loss ➜ Stochastic Gradient Descent ( slow convergence, cheap iter. )
    • Differentiable Loss ➜ Quasi-Newton Method ( very fast convergence, expensive iter. )
Lagrangian Duality

First, we consider the **Equality Constrained Problem**: 

\[
\begin{align*}
& \min_x f(x) \\
& \text{s.t. } Ax = b
\end{align*}
\]

The optimal solution \( x^* \) is found iff:

\[
Ax^* = b \quad \text{and} \quad -\nabla f(x^*) = A^T \lambda^*
\]

If we define **Lagrangian Function** (Lagrangian) as:

\[
L(x, \lambda) = f(x) + \lambda^T (Ax - b)
\]

Then the **optimality condition** can be written as:

\[
\begin{align*}
\frac{\partial L(x, \lambda)}{\partial \lambda} &= 0 \quad \Rightarrow \quad Ax^* = b \quad (\lambda \text{ cannot increase } L(.) ) \\
\frac{\partial L(x, \lambda)}{\partial x} &= 0 \quad \Rightarrow \quad -\nabla f(x^*) = A^T \lambda^* \\
\end{align*}
\]

\( (x \text{ cannot decrease } L(.) ) \)

\[
\begin{align*}
\min_x \left\{ \max_{\lambda} L(x, \lambda) \right\} \\
\max_{\lambda} \left\{ \min_x L(x, \lambda) \right\}
\end{align*}
\]
Lagrangian Duality

If we define **Lagrangian Function** (Lagrangian) as:

\[ L(x, \lambda) = f(x) + \lambda^T (Ax - b) \]

Every point satisfies

\[ \frac{\partial L(x, \lambda)}{\partial \lambda} = 0 \Rightarrow Ax^* = b \]

\[ g(\lambda) = \min_x L(x, \lambda) \]

Every point satisfies

\[ \frac{\partial L(x, \lambda)}{\partial x} = 0 \Rightarrow -\nabla f(x^*) = A^T \lambda^* \]
Lagrangian Duality

Original (Primal) problem is:

$$\min_x L(x, \lambda) = f(x) + \lambda^T(Ax - b)$$

s.t. $$\frac{\partial L(x, \lambda)}{\partial \lambda} = Ax - b = 0$$

Primal problem is:

$$\min_x \max_{\lambda} L(x, \lambda) = f(x) + \lambda^T(Ax - b)$$

Dual Problem:

$$\max_{\lambda} L(x, \lambda) = f(x) + \lambda^T(Ax - b)$$

s.t. $$\frac{\partial L(x, \lambda)}{\partial x} = \nabla f(x) + A^T \lambda = 0$$

Dual problem is:

$$\max_{\lambda} \min_x L(x, \lambda) = f(x) + \lambda^T(Ax - b)$$
# Lagrangian Duality

For **Inequality Constrained Problem:**

\[
\min_{x} f(x) \\
\text{s.t. } Ax \leq b
\]

**Primal problem is:**

\[
\min \max_{x, \lambda \geq 0} L(x, \lambda) = f(x) + \lambda^T (Ax - b)
\]

\[
\max_{\lambda \geq 0} L(x, \lambda) = \begin{cases} f(x), & Ax - b \leq 0 \\ \infty, & Ax - b > 0 \end{cases}
\]

**Dual problem is:**

\[
\max_{\lambda \geq 0} \min_{x} L(x, \lambda) = f(x) + \lambda^T (Ax - b)
\]

\[
\min_{x} L(x, \lambda) \\
\Rightarrow \nabla f(x) + A^T \lambda = 0
\]
SVM Dual Problem

Primal Problem:

\[
\begin{align*}
\min_{w, \xi \geq 0} & \quad \frac{1}{2} \|w\|^2 + C \sum_n \xi_n \\
\text{s.t.} & \quad y_n w^T \phi(x_n) \geq 1 - \xi_n, \ \forall n
\end{align*}
\]

\[
\iff
\min_{w, \xi} \max_{\alpha \geq 0, \beta \geq 0} L(w, \xi, \alpha, \beta)
\]

Lagrangian:

\[
L(w, \xi, \alpha, \beta) = \left( \frac{1}{2} \|w\|^2 + C \sum_n \xi_n \right) - \sum_n \alpha_n (y_n w^T \phi(x_n) - 1 + \xi_n) - \sum_n \beta_n \xi_n
\]

Dual Problem:

\[
\max_{\alpha \geq 0, \beta \geq 0} \min_{w, \xi} L(w, \xi, \alpha, \beta)
\]
SVM Dual Problem

**Primal Problem:**

\[
\begin{align*}
\min_{w, \xi \geq 0} & \quad \frac{1}{2} \|w\|^2 + C \sum_n \xi_n \\
\text{s.t.} & \quad y_n w^T \phi(x_n) \geq 1 - \xi_n, \quad \forall n
\end{align*}
\]

**Lagrangian:**

\[
L(w, \xi, \alpha, \beta) = \left\{ \frac{1}{2} \|w\|^2 - \sum_n \alpha_n y_n w^T \phi(x_n) \right\} + \left\{ \sum_n (C - \alpha_n - \beta_n) \xi_n \right\} + \sum_n \alpha_n
\]

**Dual Problem:**

\[
\begin{align*}
\max_{\alpha \geq 0, \beta \geq 0} & \quad \min_{w, \xi} L(w, \xi, \alpha, \beta) \\
\text{s.t.} & \quad \frac{\partial L}{\partial w} = 0 \quad \Rightarrow \quad w = \sum_n \alpha_n y_n \phi(x_n) = \Phi \alpha \\
& \quad \frac{\partial L}{\partial \xi} = 0 \quad \Rightarrow \quad C = \alpha_n + \beta_n
\end{align*}
\]
SVM Dual Problem

Primal Problem:
\[
\begin{align*}
\min_{w, \xi \geq 0} & \quad \frac{1}{2} \|w\|^2 + C \sum_n \xi_n \\
\text{s.t.} & \quad y_n w^T \phi(x_n) \geq 1 - \xi_n, \quad \forall n
\end{align*}
\]

Lagrangian:
\[
L(\alpha, \beta) = \left\{ \frac{1}{2} \alpha^T \Phi^T \Phi \alpha - \alpha^T \Phi^T \Phi \alpha \right\} + \left\{ 0 \right\} + \sum_n \alpha_n
\]

Dual Problem:
\[
\begin{align*}
\max_{\alpha \geq 0, \beta \geq 0} & \quad \min_{w, \xi} L(w, \xi, \alpha, \beta) \quad \iff \quad \max_{\alpha \geq 0, \beta \geq 0} L(w, \xi, \alpha, \beta) \quad \left[ \begin{array}{c} y_1 \phi(x_1) \\ \vdots \\ y_N \phi(x_N) \end{array} \right] = \left[ \begin{array}{c} \alpha_1 \\ \vdots \\ \alpha_N \end{array} \right] \\
\text{s.t.} & \quad \frac{\partial L}{\partial w} = 0 \quad \Rightarrow \quad w = \sum_n \alpha_n y_n \phi(x_n) = \Phi \alpha \\
& \quad \frac{\partial L}{\partial \xi} = 0 \quad \Rightarrow \quad C = \alpha_n + \beta_n
\end{align*}
\]
SVM Dual Problem

Primal Problem:

\[
\min_{w, \xi \geq 0} \frac{1}{2} \|w\|^2 + C \sum_n \xi_n \quad \text{s.t.} \quad y_n w^T \phi(x_n) \geq 1 - \xi_n, \forall n
\]

\[\iff\]

\[
\min \max \left\{ \min \max \quad L(w, \xi, \alpha, \beta) \right\}
\]

Lagrangian:

\[
L(\alpha, \beta) = \left\{ \frac{1}{2} \alpha^T \Phi^T \Phi \alpha - \alpha^T \Phi^T \Phi \alpha \right\} + \{ 0 \} + \sum_n \alpha_n
\]

Dual Problem:

\[
\max_{\alpha \geq 0, \beta \geq 0} \min_{w, \xi} L(w, \xi, \alpha, \beta) \quad \iff \quad \max_{\alpha \geq 0, \beta \geq 0} L(\alpha, \beta) = \sum_n \alpha_n - \frac{1}{2} \alpha^T \Phi^T \Phi \alpha
\]

\[\text{s.t.} \quad C = \alpha_n + \beta_n\]
SVM Dual Problem

Primal Problem:

\[
\begin{align*}
\min_{w, \xi \geq 0} & \quad \frac{1}{2} \|w\|^2 + C \sum_n \xi_n \\
\text{s.t.} & \quad y_n w^T \phi(x_n) \geq 1 - \xi_n, \quad \forall n
\end{align*}
\]

\[\iff\]

\[
\begin{align*}
\min_{w, \xi} & \quad \max_{\alpha \geq 0, \beta \geq 0} L(w, \xi, \alpha, \beta)
\end{align*}
\]

Lagrangian:

\[
L(\alpha, \beta) = \left\{ \frac{1}{2} \alpha^T \Phi^T \Phi \alpha - \alpha^T \Phi^T \Phi \alpha \right\} + \{ 0 \} + \sum_n \alpha_n
\]

Dual Problem:

\[
\begin{align*}
\max_{\alpha \geq 0, \beta \geq 0} & \quad \min_{w, \xi} L(w, \xi, \alpha, \beta) \\
\iff\quad & \quad \max_{\alpha} L(\alpha, \beta) = \sum_n \alpha_n - \frac{1}{2} \alpha^T \Phi^T \Phi \alpha \\
\text{s.t.} & \quad 0 \leq \alpha \leq C
\end{align*}
\]
SVM Dual Problem

Dual Problem (only involve product $\phi(x_i)^T\phi(x_j)$):

$$\max_{\alpha} \sum_{n} \alpha_n - \frac{1}{2} \alpha^T \Phi^T \Phi \alpha$$

s.t. $0 \leq \alpha \leq C$

$$\max_{\alpha} \sum_{n} \alpha_n - \frac{1}{2} \alpha^T Q \alpha$$

s.t. $0 \leq \alpha \leq C$

$Q_{ij} = (y_i\phi(x_i))(y_j\phi(x_j)) = y_iy_jK(x_i, x_j)$

1. Only “Box Constraint” $\Rightarrow$ Easy to solve.

2. $\text{dim}(\alpha) = N = |\text{instance}|$, $\text{dim}(w) = D = |\text{features}|$

3. Weak Duality: $\text{Dual}(\alpha) \leq \text{Primal}(w)$

4. String Duality: $\text{Dual}(\alpha^*) = \text{Primal}(w^*)$
   (if primal is convex)
Overview

• **Support Vector Machine**
  – The Art of Modeling --- Large Margin and Kernel Trick
  – Convex Analysis
  – Optimality Conditions
  – Duality

• **Optimization for Machine Learning**
  – Dual Coordinate Descent (DCD) (fast convergence, moderate cost)
    • libLinear (Stochastic CD)
    • libSVM (Greedy CD)
  – Primal Methods
    • Non-smooth Loss ➔ Stochastic Gradient Descent (slow convergence, cheap iter.)
    • Differentiable Loss ➔ Quasi-Newton Method (very fast convergence, expensive iter.)
Dual Optimization of SVM

\[
\begin{align*}
\max_{\alpha} & \quad \sum_n \alpha_n - \frac{1}{2} \alpha^T Q \alpha \\
\text{s.t.} & \quad 0 \leq \alpha \leq C
\end{align*}
\]  \quad \Rightarrow \quad
\begin{align*}
\min_{\alpha} & \quad \frac{1}{2} \alpha^T Q \alpha - \sum_n \alpha_n \\
\text{s.t.} & \quad 0 \leq \alpha \leq C
\end{align*}

\[Q_{ij} = (y_i \phi(x_i))(y_j \phi(x_j)) = y_i y_j K(x_i, x_j)\]

Usage: train [options] training_set_file [model_file]
options:
-s type : set type of solver (default 1)
  for multi-class classification
    0 -- L2-regularized logistic regression (primal)
    1 -- L2-regularized L2-loss support vector classification (dual)
    2 -- L2-regularized L2-loss support vector classification (primal)
    3 -- L2-regularized L1-loss support vector classification (dual)
    4 -- support vector classification by Crammer and Singer
    5 -- L1-regularized L2-loss support vector classification
    6 -- L1-regularized logistic regression
    7 -- L2-regularized logistic regression (dual)
  for regression
    11 -- L2-regularized L2-loss support vector regression (primal)
    12 -- L2-regularized L2-loss support vector regression (dual)
    13 -- L2-regularized L1-loss support vector regression (dual)
Constrained Minimization

\[
\min_{w, \xi \geq 0} \frac{1}{2} \|w\|^2 + C \sum_n \xi_n
\]

s.t. \( y_n w^T \phi(x_n) \geq 1 - \xi_n, \ \forall n \)

General Constraint → Very Expensive:

1. Detecting binding constraint : \( O( |\text{constraint}| \times \text{dim}(\alpha) ) \)

2. Compute “Projected Gradient” : \( O( |\text{binding constraint}| \times \text{dim}(\alpha) ) \)

\[ -\nabla^P f(x) = -Z_1^T \nabla f(x) \]
Constrained Minimization for “Box Constraint”

\[
\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - \sum_n \alpha_n
\]
\[s.t. \quad 0 \leq \alpha \leq C\]

Cheap:

1. Detecting binding constraint: \(O(|\text{constraint}|)\)
2. Compute “Projected Gradient”: \(O(|\text{binding constraint}|)\)
Dual Coordinate Descent

\[
\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - \sum_n \alpha_n
\]

s.t. \( 0 \leq \alpha \leq C \)

Minimize w.r.t.

\[
\min_{\alpha_i} \frac{1}{2} [\nabla^2 f(\alpha)] \alpha_i^2 + [\nabla f(\alpha)] \alpha_i + \text{const.}
\]

s.t. \( 0 \leq \alpha_i \leq C \)

\[
\nabla^2 f(\alpha) = Q_{ii}
\]

\[
\nabla f(\alpha) = [Q \alpha - 1]_i
\]

\[
\alpha_i \leftarrow \min (\max (\alpha_i - \frac{\nabla f(\alpha)_i}{\nabla^2 f(\alpha)_{ii}}, 0), C)
\]
Dual Optimization of SVM

\[
\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - \sum_n \alpha_n \\
\text{s.t. } 0 \leq \alpha \leq C
\]

Even Computing Gradient is Expensive:
\[
\nabla f(\alpha) = Q_{N \times N} \alpha_{N \times 1} - 1 \quad (O(N^2))
\]

Coordinate Descent: (Optimize w.r.t. one variable at a time)
\[
\nabla f(\alpha)_{(i)} = [Q]_{i,:} \alpha_{N \times 1} - 1 = \sum_k \alpha_k y_i y_k K(x_i, x_k) - 1 \quad (O(N))
\]

How many variables? \(\Rightarrow\) As few as possible

- Sequential Minimal Optimization (LibSVM) (2 variable at a time)
- Coordinate Descent (LibLinear) (1 variable at a time)
NonLinear (LibSVM) vs. Linear (LibLinear)

**Linear:**

\[ \nabla f(\alpha)_{(i)} = [Q]_{i,:} \alpha_{N*1} - 1 = \sum_k \alpha_k y_i y_k K(x_i, x_k) - 1 \]

\[ = \sum_k \alpha_k y_i y_k (x_i^T x_k) - 1 = y_i x_i^T (\sum_k \alpha_k y_k x_k) - 1 = y_i x_i^T w - 1 \]

O( |non-zero Feature| )

**Non-Linear:**

\[ \nabla f(\alpha)_{(i)} = [Q]_{i,:} \alpha_{N*1} - 1 = \sum_k \alpha_k y_i y_k K(x_i, x_k) - 1 \]

O( |Instances| * |non-zero Features| )
NonLinear (LibSVM) vs. Linear (LibLinear)

**Linear:**

\[
\nabla f(\alpha)_{(i)} = y_i x_i^T w - 1
\]

\[w = \sum_{n} \alpha_n y_n x_n\]

\[
\nabla f(\alpha)_{(i)} = y_i x_i^T w - 1
\]

\[O(\mid \text{Feature} \mid )\]

( Cheap Update → Random Select Coordinate )

**Non-Linear:**

\[
\nabla f(\alpha)_{(i)} = \sum_{k} \alpha_k y_i y_k K(x_i, x_k) - 1
\]

\[O(\mid \text{Instances} \mid )\]

( Expensive Update → Select most Promising Coordinate )
LibLinear

Linear:

\[
\nabla f(\alpha)_{(i)} = y_i x_i^T w - 1
\]

\[O( |Feature| )\]
LibSVM

Non-Linear:

\[ \nabla f(\alpha)_{(i)} = \sum_k \alpha_k y_i y_k K(x_i, x_k) - 1 \]

- \( O(|\text{Instances}| \times |\text{Features}|) \)
  - (no cache)
- \( O(|\text{Instances}|) \)
  - (cache)

Choose 2 most Promising Coordinates

Update \( \nabla f(\alpha)_{(i)}, i=1 \sim N \)

\[ \alpha_i' = \text{proj}(\alpha_i + \eta^* \nabla f(\alpha)_{(i)}) \]
Demo: libSVM, libLinear

– Normalize Features:
  • `svm-scale -s [range_file] [train] > train.scale`
  • `svm-scale -r [range_file] [test] > test.scale`

– Training:
  • LibSVM: `svm-train [train.scale]` (produce `train.scale.model`)
  • LibSVM: `svm-predict [test.scale] [train.scale.model] [pred_output]`

  • LibLinear: `train [train.scale]` (produce `train.scale.model`)
  • LibLinear: `predict [test.scale] [train.scale.model] [pred_output]`