Support Vector Machines: Training with Stochastic Gradient Descent

Machine Learning
Fall 2023
Support vector machines

• Training by maximizing margin

• The SVM objective

• **Solving the SVM optimization problem**

• Support vectors, duals and kernels
SVM objective function

$$\min_{w, b} \frac{1}{2} w^T w + C \sum \max(0, 1 - y_i(w^T x_i + b))$$

Regularization term:
- Maximize the margin
- Imposes a preference over the hypothesis space and pushes for better generalization
- Can be replaced with other regularization terms which impose other preferences

Empirical Loss:
- Hinge loss
- Penalizes weight vectors that make mistakes
- Can be replaced with other loss functions which impose other preferences

A hyper-parameter that controls the tradeoff between a large margin and a small hinge-loss
Outline: Training SVM by optimization

1. Review of convex functions and gradient descent
2. Stochastic gradient descent
3. Gradient descent vs stochastic gradient descent
4. Sub-derivatives of the hinge loss
5. Stochastic sub-gradient descent for SVM
6. Comparison to perceptron
Outline: Training SVM by optimization

1. **Review of convex functions and gradient descent**
2. Stochastic gradient descent
3. Gradient descent vs stochastic gradient descent
4. Sub-derivatives of the hinge loss
5. Stochastic sub-gradient descent for SVM
6. Comparison to perceptron
Solving the SVM optimization problem

\[
\min_{w,b} \frac{1}{2} w^\top w + C \sum_i \max\left(0, 1 - y_i (w^\top x_i + b)\right)
\]

This function is **convex** in \(w, b\)

For convenience, use simplified notation:

\[
\begin{align*}
  w_0 &\leftarrow w \\
  w &\leftarrow [w_0, b] \\
  x_i &\leftarrow [x_i, 1]
\end{align*}
\]

\[
\min_w \frac{1}{2} w_0^\top w_0 + C \sum_i \max(0, 1 - y_i w^\top x_i)
\]
A function $f$ is convex if for every $u, v$ in the domain, and for every $\lambda \in [0,1]$ we have

$$f(\lambda u + (1 - \lambda)v) \leq \lambda f(u) + (1 - \lambda)f(v)$$
Recall: Convex functions

A function $f$ is **convex** if for every $u, v$ in the domain, and for every $\lambda \in [0,1]$ we have

$$f(\lambda u + (1 - \lambda)v) \leq \lambda f(u) + (1 - \lambda)f(v)$$

From geometric perspective

Every tangent plane lies below the function

$$f(x) \geq f(u) + \nabla f(u)^\top (x - u)$$
Convex functions

$f(x) = -x$
Linear functions

$\begin{align*}
  f(x_1, x_2) &= \frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} \\
  f(x) &= x^2
\end{align*}$

$f(x) = \max(0, x)$
$max$ is convex

Some ways to show that a function is convex:

1. Using the definition of convexity
2. Showing that the second derivative is nonnegative (for one dimensional functions)
3. Showing that the second derivative is positive semi-definite (for vector functions)
Not all functions are convex

\[ f(\lambda u + (1 - \lambda)v) \geq \lambda f(u) + (1 - \lambda)f(v) \]

These are concave

These are neither
Convex functions are convenient

A function $f$ is **convex** if for every $u, v$ in the domain, and for every $\lambda \in [0,1]$ we have

$$f(\lambda u + (1 - \lambda)v) \leq \lambda f(u) + (1 - \lambda)f(v)$$

In general: Necessary condition for $x$ to be a minimum for the function $f$ is $\nabla f (x) = 0$
Convex functions are convenient

A function \( f \) is **convex** if for every \( u, v \) in the domain, and for every \( \lambda \in [0,1] \) we have

\[
f(\lambda u + (1 - \lambda)v) \leq \lambda f(u) + (1 - \lambda)f(v)
\]

In general: Necessary condition for \( x \) to be a minimum for the function \( f \) is \( \nabla f(x) = 0 \)

For convex functions, this is both necessary *and* sufficient
Solving the SVM optimization problem

\[
\min_w \frac{1}{2} w_0^T w_0 + C \sum_i \max(0, 1 - y_i w^T x_i)
\]

This function is convex in \( w \)

- This is a quadratic optimization problem because the objective is quadratic

- Older methods: Used techniques from Quadratic Programming
  - Very slow

- No constraints, can use \textit{gradient descent}
  - Still very slow!
Gradient descent

General strategy for minimizing a function $J(w)$

- Start with an initial guess for $w$, say $w^0$
- Iterate till convergence:
  - Compute the gradient of $J$ at $w^t$
  - Update $w^t$ to get $w^{t+1}$ by taking a step in the opposite direction of the gradient

**Intuition**: The gradient is the direction of steepest increase in the function. To get to the minimum, go in the opposite direction.

We are trying to minimize

$$J(w) = \frac{1}{2} w_0^T w_0 + C \sum_i \max(0, 1 - y_i w^T x_i)$$
Gradient descent

General strategy for minimizing a function $J(w)$

- Start with an initial guess for $w$, say $w^0$
- Iterate till convergence:
  - Compute the gradient of $J$ at $w^t$
  - Update $w^t$ to get $w^{t+1}$ by taking a step in the opposite direction of the gradient

We are trying to minimize

$$J(w) = \frac{1}{2} w_0^T w_0 + C \sum_i \max(0, 1 - y_i w^T x_i)$$

**Intuition**: The gradient is the direction of steepest increase in the function. To get to the minimum, go in the opposite direction.
Gradient descent

General strategy for minimizing a function $J(\mathbf{w})$

- Start with an initial guess for $\mathbf{w}$, say $\mathbf{w}^0$

- Iterate till convergence:
  - Compute the gradient of $J$ at $\mathbf{w}^t$
  - Update $\mathbf{w}^t$ to get $\mathbf{w}^{t+1}$ by taking a step in the opposite direction of the gradient

Intuition: The gradient is the direction of steepest increase in the function. To get to the minimum, go in the opposite direction

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)$$
Gradient descent

General strategy for minimizing a function \( J(\mathbf{w}) \)

- Start with an initial guess for \( \mathbf{w} \), say \( \mathbf{w}^0 \)
- Iterate till convergence:
  - Compute the gradient of \( J \) at \( \mathbf{w}^t \)
  - Update \( \mathbf{w}^t \) to get \( \mathbf{w}^{t+1} \) by taking a step in the opposite direction of the gradient

\[
J(\mathbf{w}) = \frac{1}{2} \mathbf{w}_0^\top \mathbf{w}_0 + C \sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)
\]

**Intuition**: The gradient is the direction of steepest increase in the function. To get to the minimum, go in the opposite direction.
Gradient descent for SVM

1. Initialize $\mathbf{w}^0$

2. For $t = 0, 1, 2, \ldots$
   1. Compute gradient of $J(\mathbf{w})$ at $\mathbf{w}^t$. Call it $\nabla J(\mathbf{w}^t)$

   2. Update $\mathbf{w}$ as follows:
      
      $$
      \mathbf{w}^{t+1} = \mathbf{w}^t - r \nabla J(\mathbf{w}^t)
      $$

      $r$: Called the learning rate.

We are trying to minimize

$$
J(\mathbf{w}) = \frac{1}{2} \mathbf{w}_0^\top \mathbf{w}_0 + C \sum_i \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i)
$$
Outline: Training SVM by optimization

✓ Review of convex functions and gradient descent

2. **Stochastic gradient descent**

3. Gradient descent vs stochastic gradient descent

4. Sub-derivatives of the hinge loss

5. Stochastic sub-gradient descent for SVM

6. Comparison to perceptron
Gradient descent for SVM

1. Initialize $w^0$

2. For $t = 0, 1, 2, \ldots$
   1. Compute gradient of $J(w)$ at $w^t$. Call it $\nabla J(w^t)$

   Gradient of the SVM objective requires summing over the entire training set
   
   Slow, does not really scale

   $\beta$: Called the learning rate

We are trying to minimize

$$J(w) = \frac{1}{2} w_0^T w_0 + C \sum_i \max(0, 1 - y_i w^T x_i)$$
Stochastic gradient descent for SVM

Given a training set $S = \{(x_i, y_i)\}, x \in \mathbb{R}^n, y \in \{-1, 1\}$

1. Initialize $w^0 = 0 \in \mathbb{R}^n$

2. For epoch $= 1 \ldots T$:

3. Return final $w$
Stochastic gradient descent for SVM

Given a training set $S = \{(x_i, y_i)\}$, $x \in \mathbb{R}^n$, $y \in \{-1, 1\}$

1. Initialize $w^0 = 0 \in \mathbb{R}^n$

2. For epoch = 1 ... T:
   1. Pick a random example $(x_i, y_i)$ from the training set $S$

3. Return final $w$
Stochastic gradient descent for SVM

Given a training set \( S = \{ (x_i, y_i) \}, \ x \in \mathbb{R}^n, \ y \in \{-1, 1\} \)

1. Initialize \( w^0 = 0 \in \mathbb{R}^n \)

2. For epoch = 1 … T:
   1. Pick a random example \( (x_i, y_i) \) from the training set \( S \)
   2. Repeat \( (x_i, y_i) \) to make a full dataset and take the derivative of the SVM objective at the current \( w \) to be \( \nabla J^t(w) \)

3. Return final \( w \)
Stochastic gradient descent for SVM

Given a training set $S = \{(x_i, y_i)\}$, $x \in \mathbb{R}^n$, $y \in \{-1, 1\}$

1. Initialize $w^0 = 0 \in \mathbb{R}^n$

2. For epoch = 1 ... T:
   1. Pick a random example $(x_i, y_i)$ from the training set $S$
   2. Repeat $(x_i, y_i)$ to make a full dataset and take the derivative of the SVM objective at the current $w$ to be $\nabla J^t(w)$

   $$J^t(w) = \frac{1}{2} w_0^T w_0 + C \cdot N \max(0, 1 - y_i w^T x_i)$$

3. Return final $w$
Stochastic gradient descent for SVM

Given a training set \( S = \{(x_i, y_i)\}, x \in \mathbb{R}^n, y \in \{-1, 1\} \)

1. Initialize \( w^0 = 0 \in \mathbb{R}^n \)

2. For epoch = 1 ... T:
   1. Pick a random example \((x_i, y_i)\) from the training set \( S \)
   2. Repeat \((x_i, y_i)\) to make a full dataset and take the derivative of the SVM objective at the current \( w \) to be \( \nabla J^t(w) \)

\[
J^t(w) = \frac{1}{2} w_0^T w_0 + C \sum_i \max(0, 1 - y_i w^T x_i)
\]

3. Return final \( w \)
Stochastic gradient descent in general

\[ f(w) = R(w) + C \sum_{n=1}^{N} l(x_n, w) \]

Randomly sample, duplicate it to make a full data, then compute the gradient

\[ f(w) = R(w) + C \frac{1}{N} \sum_{n=1}^{N} N \cdot l(x_n, w) \]

\[ f(w) = R(w) + C E_{p(k)}[N \cdot l(x_k, w)] \]

\[ p(k) = \frac{1}{N} \quad (k = 1 \ldots N) \]

\[ f(w) = E_{p(k)}[R(w) + C \cdot N \cdot l(x_k, w)] \]

\[ \nabla f(w) = \nabla E_{p(k)}[R(w) + C \cdot N \cdot l(x_k, w)] \]

\[ = E_{p(k)}[\nabla (R(w) + C \cdot N \cdot l(x_k, w))] \]

\[ \nabla \tilde{f}(w) \]

Unbiased stochastic gradient: ensure convergence
Stochastic gradient descent for SVM

Given a training set $S = \{(x_i, y_i)\}, x \in \mathbb{R}^n, y \in \{-1, 1\}$

1. Initialize $w^0 = 0 \in \mathbb{R}^n$

2. For epoch $= 1 \ldots T$:
   1. Pick a random example $(x_i, y_i)$ from the training set $S$
   2. Repeat $(x_i, y_i)$ to make a full dataset and take the derivative of the SVM objective at the current $w$ to be $\nabla J^t(w)$

   $$J^t(w) = \frac{1}{2} w_0^T w_0 + C \cdot N \max (0, 1 - y_i w^T x_i)$$

3. Update: $w \leftarrow w - \gamma_t \nabla J^t(w)$

3. Return final $w$
Stochastic gradient descent for SVM

Given a training set $S = \{(x_i, y_i)\}$, $x \in \mathbb{R}^n$, $y \in \{-1, 1\}$

1. Initialize $w^0 = 0 \in \mathbb{R}^n$
2. For epoch = 1 ... T:
   1. Pick a random example $(x_i, y_i)$ from the training set S
   2. Repeat $(x_i, y_i)$ to make a full dataset and take the derivative of the SVM objective at the current $w$ to be $\nabla J^t(w)$
   3. Update: $w \leftarrow w - \gamma_t \nabla J^t(w)$

3. Return final $w$

This algorithm is guaranteed to converge to the minimum of $J$ if $\gamma_t$ is small enough.
Outline: Training SVM by optimization

 ✓ Review of convex functions and gradient descent
 ✓ Stochastic gradient descent

3. **Gradient descent vs stochastic gradient descent**
4. Sub-derivatives of the hinge loss
5. Stochastic sub-gradient descent for SVM
6. Comparison to perceptron
Gradient Descent vs SGD

Gradient descent
Gradient Descent vs SGD

Stochastic Gradient descent
Gradient Descent vs SGD

Stochastic Gradient descent
Gradient Descent vs SGD

Stochastic Gradient descent
Gradient Descent vs SGD

Stochastic Gradient descent
Gradient Descent vs SGD

Stochastic Gradient descent
Gradient Descent vs SGD

Stochastic Gradient descent
Gradient Descent vs SGD

Stochastic Gradient descent
Gradient Descent vs SGD

Stochastic Gradient descent
Gradient Descent vs SGD

Stochastic Gradient descent
Gradient Descent vs SGD

Stochastic Gradient descent
Gradient Descent vs SGD

Stochastic Gradient descent
Gradient Descent vs SGD

Stochastic Gradient descent
Gradient Descent vs SGD

Stochastic Gradient descent
Gradient Descent vs SGD

Stochastic Gradient descent
Gradient Descent vs SGD

Stochastic Gradient descent
Gradient Descent vs SGD

Stochastic Gradient descent
Gradient Descent vs SGD

Stochastic Gradient descent
Gradient Descent vs SGD

Many more updates than gradient descent, but each individual update is less computationally expensive

Stochastic Gradient descent
Outline: Training SVM by optimization

✓ Review of convex functions and gradient descent
✓ Stochastic gradient descent
✓ Gradient descent vs stochastic gradient descent

4. **Sub-derivatives of the hinge loss**
5. Stochastic sub-gradient descent for SVM
6. Comparison to perceptron
Stochastic gradient descent for SVM

Given a training set $S = \{(x_i, y_i)\}$, $x \in \mathbb{R}^n$, $y \in \{-1, 1\}$

1. Initialize $w^0 = 0 \in \mathbb{R}^n$
2. For epoch $= 1 \ldots T$:
   1. Pick a random example $(x_i, y_i)$ from the training set $S$
   2. Treat $(x_i, y_i)$ as a full dataset and take the derivative of the SVM objective at the current $w$ to be $\nabla J_t(w)$
   3. Update: $w \leftarrow w - \gamma_t \nabla J_t(w)$

3. Return final $w$

What is the derivative of the hinge loss with respect to $w$? (The hinge loss is not a differentiable function!)
Hinge loss is **not** differentiable!

What is the derivative of the hinge loss with respect to $w$?

$$J^t(w) = \frac{1}{2}w_0^\top w_0 + C \cdot N \max(0, 1 - y_i w^\top x_i)$$
Detour: Sub-gradients

Generalization of gradients to non-differentiable functions
(Remember that every tangent lies below the function for convex functions)

Informally, a sub-tangent line at a point is any line that crosses the point and lies below the entire function.
A sub-gradient is the slope of the sub-tangent line
Sub-gradients

Formally, g is a subgradient to f at x if

\[ f(y) \geq f(x) + g^T(y - x) \quad \text{for all } y \]

[Example from Boyd]
Sub-gradients

Formally, $g$ is a subgradient to $f$ at $x$ if

$$f(y) \geq f(x) + g^T(y - x) \quad \text{for all } y$$

$f$ is differentiable at $x_1$
Tangent at this point

$$f(x_1) + g_1^T(x - x_1)$$

$g_1$ is a gradient at $x_1$
Sub-gradients

Formally, $g$ is a subgradient to $f$ at $x$ if

\[ f(y) \geq f(x) + g^T(y - x) \quad \text{for all } y \]

$f$ is differentiable at $x_1$

Tangent at this point

\[ f(x_1) + g_1^T(x - x_1) \]

$g_1$ is a gradient at $x_1$

$g_2$ and $g_3$ are both subgradients at $x_2$

[Example from Boyd]
Sub-gradient of the SVM objective

\[ J^t(w) = \frac{1}{2} w_0^\top w_0 + C \cdot N \max(0, 1 - y_i w^\top x_i) \]

**General strategy**: First solve the max and compute the gradient for each case.
Sub-gradient of the SVM objective

\[
J^t(w) = \frac{1}{2} w_0^T w_0 + C \cdot N \max(0, 1 - y_i w^T x_i)
\]

**General strategy:** First solve the max and compute the gradient for each case

\[
\nabla J^t = \begin{cases} 
[w_0; 0] & \text{if } \max(0, 1 - y_i w^T x_i) = 0 \\
[w_0; 0] - C \cdot N y_i x_i & \text{otherwise}
\end{cases}
\]
Outline: Training SVM by optimization

- Review of convex functions and gradient descent
- Stochastic gradient descent
- Gradient descent vs stochastic gradient descent
- Sub-derivatives of the hinge loss

5. Stochastic sub-gradient descent for SVM

6. Comparison to perceptron
Stochastic sub-gradient descent for SVM

\[ \nabla J_t = \begin{cases} [w_0; 0] & \text{if } \max(0, 1 - y_i w^\top x_i) = 0 \\ [w_0; 0] - C \cdot N y_i x_i & \text{otherwise} \end{cases} \]

Given a training set \( S = \{(x_i, y_i)\}, \ x \in \mathbb{R}^n, \ y \in \{-1,1\} \)

1. Initialize \( w^0 = 0 \in \mathbb{R}^n \)

3. Return \( w \)
Stochastic sub-gradient descent for SVM

\[ \nabla J^t = \begin{cases} 
[w_0; 0] & \text{if } \max(0, 1 - y_i w^\top x_i) = 0 \\
[w_0; 0] - C \cdot N y_i x_i & \text{otherwise}
\end{cases} \]

Given a training set \( S = \{(x_i, y_i)\}, \ x \in \mathbb{R}^n, \ y \in \{-1, 1\} \)

1. Initialize \( w^0 = 0 \in \mathbb{R}^n \)

2. For epoch = 1 ... T:

3. Return \( w \)
Stochastic sub-gradient descent for SVM

\[ \nabla J^t = \begin{cases} 
[w_0; 0] & \text{if } \max(0, 1 - y_i w^\top x_i) = 0 \\
[w_0; 0] - C \cdot N y_i x_i & \text{otherwise}
\end{cases} \]

Given a training set \( S = \{(x_i, y_i)\}, x \in \mathbb{R}^n, y \in \{-1, 1\} \)

1. Initialize \( w^0 = 0 \in \mathbb{R}^n \)

2. For epoch = 1 ... T:
   1. For each training example \((x_i, y_i) \in S:\)

      Update \( w \leftarrow w - \gamma_t \nabla J^t \)

3. Return \( w \)
Stochastic sub-gradient descent for SVM

\[ \nabla J^t = \begin{cases} 
[w_0; 0] & \text{if } \max(0, 1 - y_i w^T x_i) = 0 \\
[w_0; 0] - C \cdot N y_i x_i & \text{otherwise}
\end{cases} \]

Given a training set \( S = \{(x_i, y_i)\}, x \in \mathbb{R}^n, y \in \{-1, 1\} \)

1. Initialize \( w^0 = 0 \in \mathbb{R}^n \)

2. For epoch = 1 ... T:
   1. For each training example \((x_i, y_i) \in S:\)
      1. If \( y_i w^T x_i \leq 1, \)
         \[
         w \leftarrow w - \gamma_t [w_0; 0] + \gamma_t C N y_i x_i
         \]
      else
         \[
         w_0 \leftarrow (1 - \gamma_t) w_0
         \]

3. Return \( w \)
Stochastic sub-gradient descent for SVM

Given a training set \( S = \{(x_i, y_i)\} \), \( x \in \mathbb{R}^n \), \( y \in \{-1, 1\} \)

1. Initialize \( w^0 = 0 \in \mathbb{R}^n \)

2. For epoch = 1 ... T:
   1. For each training example \((x_i, y_i)\) \( \in \) S:
      - If \( y_i w^T x_i \leq 1 \),
        \[ w \leftarrow w - \gamma_t [w_0; 0] + \gamma_t C N y_i x_i \]
      - else
        \[ w_0 \leftarrow (1 - \gamma_t) w_0 \]

3. Return \( w \)
Stochastic sub-gradient descent for SVM

Given a training set \( S = \{(x_i, y_i)\}, \mathbf{x} \in \mathbb{R}^n, y \in \{-1,1\} \)

1. Initialize \( w^0 = 0 \in \mathbb{R}^n \)

2. For epoch = 1 ... T:
   1. Shuffle the training set
   2. For each training example \((x_i, y_i) \in S:\)
      - If \( y_i \mathbf{w}^T \mathbf{x}_i \leq 1, \)
        \[ \mathbf{w} \leftarrow \mathbf{w} - \gamma_t \left[ w_0; 0 \right] + \gamma_t C N y_i \mathbf{x}_i \]
      - else
        \[ w_0 \leftarrow (1- \gamma_t) w_0 \]

3. Return \( w \)

\( \gamma_t \): learning rate, many tweaks possible
Convergence and learning rates

With enough iterations, it will converge in expectation

Provided the step sizes are “square summable, but not summable”

• Step sizes $\gamma_t$ are positive
• Sum of squares of step sizes over $t = 1$ to $\infty$ is not infinite
• Sum of step sizes over $t = 1$ to $\infty$ is infinity

• Some examples: $\gamma_t = \frac{\gamma_0}{1 + \gamma_0 t}$ or $\gamma_t = \frac{\gamma_0}{1 + t}$
Convergence and learning rates

• Number of iterations to get to accuracy within $\epsilon$

• For strongly convex functions, N examples, d dimensional:
  – Gradient descent: $O(Nd \ln(1/\epsilon))$
  – Stochastic gradient descent: $O(d/\epsilon)$

• More subtleties involved, but SGD is generally preferable when the data size is huge
Outline: Training SVM by optimization

✓ Review of convex functions and gradient descent
✓ Stochastic gradient descent
✓ Gradient descent vs stochastic gradient descent
✓ Sub-derivatives of the hinge loss
✓ Stochastic sub-gradient descent for SVM

6. **Comparison to perceptron**
Stochastic sub-gradient descent for SVM

Given a training set \( S = \{(x_i, y_i)\}, \ x \in \mathbb{R}^n, \ y \in \{-1,1\} \)

1. Initialize \( w^0 = 0 \in \mathbb{R}^n \)

2. For epoch = 1 ... T:
   1. Shuffle the training set
   2. For each training example \((x_i, y_i) \in S:\)
      - If \( y_i w^T x_i \leq 1, \)
        \[
        w \leftarrow w - \gamma_t [w_0; 0] + \gamma_t CN y_i x_i
        \]
      - else
        \[
        w_0 \leftarrow (1 - \gamma_t) w_0
        \]

3. Return \( w \)

Compare with the Perceptron update:
If \( y_i w^T x_i \leq 0, \) update \( w \leftarrow w + r y_i x_i \)
Perceptron vs. SVM

• Perceptron: Stochastic sub-gradient descent for a different loss
  – No regularization though

\[ L_{\text{Perceptron}}(y, x, w) = \max(0, -yw^T x) \]

• SVM optimizes the hinge loss
  – With regularization

\[ L_{\text{Hinge}}(y, x, w) = \max(0, 1 - yw^T x) \]
SVM summary from optimization perspective

- Minimize regularized hinge loss

- Solve using stochastic sub-gradient descent
  - Very fast, run time does not depend on number of examples
  - Compare with Perceptron algorithm: similar framework with different objectives!
  - Compare with Perceptron algorithm: Perceptron does not maximize margin width
    - Perceptron variants can force a margin

- Other successful optimization algorithms exist
  - Eg: Dual coordinate descent, implemented in liblinear

Questions?