CS 559 Machine Learning

Support Vector Machines

Yue Ning
Department of Computer Science
Stevens Institute of Technology

Plan for today

Vector algebra, Formulation, Margin

Non-separable Case, Penalties

Non-linearity, Kernels

Review of last lecture - Generative models

- We assumed $p(\mathbf{x}|(y=1)) \sim \mathcal{N}(\mu_+, \Sigma_+)$ and $p(\mathbf{x}|(y=-1)) \sim \mathcal{N}(\mu_-, \Sigma_-)$, and two class-probabilities $p(y=1)$ and $p(y=-1)$.
- This is called an generative model, as we have written down a full joint model over the data.
- We saw that violations of the model assumption can lead to “bad” decision boundaries.

Review of last lecture - Generative models - parameters

- How many parameters did we estimate to fit Gaussian class-conditional densities (the generative approach)?

$$\begin{aligned} p(C_1) &\Rightarrow 1 \\ \text{2 mean vectors } &\Rightarrow 2d \\ \Sigma &\Rightarrow d + \frac{d^2 - d}{2} = \frac{d^2 + d}{2} \\ \text{total } &= 1 + 2d + \frac{d^2 + d}{2} = O(d^2) \end{aligned}$$

Review of last lecture - Discriminative Models

- Two-class case:

$$P(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + \omega_0) = f(\mathbf{x})$$

$$P(C_2|\mathbf{x}) = 1 - P(C_1|\mathbf{x})$$

- This model is know as Logistic Regression.
- Assuming $\mathbf{x} \in \mathbb{R}^d$, how many parameters do we need to estimate?

$$d + 1$$

Review of last lecture - Logistic Regression

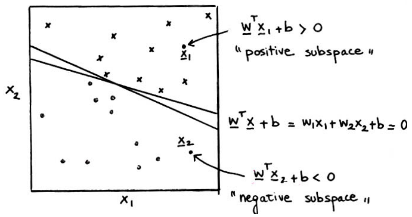
- From the previous introduction, we know that

$$P(y = 1|\mathbf{x}) = \sigma(a(\mathbf{x}))$$

  where $\sigma(a) = 1/(1 + \exp(-a))$ and $a(x) = \mathbf{w}^\top \mathbf{x} + \omega_o$.
- Notation is simpler if we use 0 and 1 as class labels, so we define $y_n = 1$ as the label for the positive class, and $y_n = 0$ as label for the negative class.
- In other words, $y|x \sim \text{Bernoulli}(\sigma(f(x)))$.
- The parameters of $a(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + \omega_o$ can be learned by maximizing the conditional log-likelihood $L(\mathbf{w}) = \sum_n \log p(y_n|x_n, \mathbf{w})$
- This is a discriminative approach to classification, as we only model the labels, and not the inputs.
- Decision rule and function shape of $p(y|\mathbf{x})$ will be the same for the generative and the discriminative model, but the parameters were obtained differently.

Linear Classifier

- Linear classifiers construct linear decision boundaries(hyperplanes) that try to separate the data into different classes as well as possible.



$\underline{w}^T\underline{x}_1 + b > 0$
"positive subspace"
$\underline{x}_1$
x_2
$\underline{w}^T\underline{x} + b = w_1x_1 + w_2x_2 + b = 0$
$\underline{x}_2$
$\underline{w}^T\underline{x}_2 + b < 0$
"negative subspace"
x_1

- Classification Rule (**the Perceptron model**):

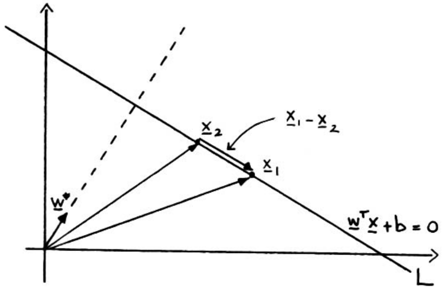
$$\text{Input: } \mathbf{x} \in \mathbb{R}^d$$

$$\text{Output: } \text{sign}(\mathbf{w}^T\mathbf{x} + b)$$

- The classifier computes a linear combination of the input features, and return the sign.

Some Vector Algebra

- Hyperplane L: $f(\mathbf{x}) = (\mathbf{w}^T\mathbf{x} + b) = 0$, when $\mathbf{x} \in \mathbb{R}^2$, $f(\mathbf{x})$ is a line.



$\underline{x}_1 - \underline{x}_2$
$\underline{x}_2$
$\underline{x}_1$
$\underline{w}^*$
$\underline{w}^T\underline{x} + b = 0$
L

Some Vector Algebra - property 1

- Consider any two points x_1, x_2, lying in L:
  $$\begin{matrix} \mathbf{w}\mathbf{x}_1 + b = 0 \\ \mathbf{w}\mathbf{x}_2 + b = 0 \end{matrix} \rightarrow \mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2) = 0$$
- Since $\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2) = \mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) = 0$, the two vectors $\mathbf{w}$ and $\mathbf{x}_1 - \mathbf{x}_2$ are orthogonal vectors.

  $\boxed{范式}$
  $$\mathbf{w}^* = \frac{\mathbf{w}}{||\mathbf{w}||}$$

  is the vector normal to the surface of L.
- Note 1: All vectors here are column vectors.
- Note 2: Dot product (inner product) of two vectors $\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T\mathbf{b} = ||\mathbf{a}||||\mathbf{b}|| \cos\alpha$ where α is the angle between a and b.

Some Vector Algebra - property 2

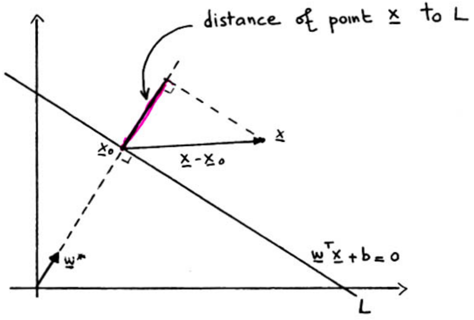
- For any point $\mathbf{x}_0$ in L:

$$\mathbf{w}^T\mathbf{x}_0 + b = 0$$

Thus:

$$\mathbf{w}^T\mathbf{x}_0 = -b$$

Some Vector Algebra - property 3


distance of point $\underline{x}$ to L
$\underline{x}$
$\underline{x}_0$
$\underline{x} - \underline{x}_0$
$\underline{w}^*$
$\underline{w}^T\underline{x} + b = 0$
L

The signed distance of any point $\mathbf{x}$ to L is:

$$(\mathbf{w}^*)^T(\mathbf{x} - \mathbf{x}_0) = \frac{\mathbf{w}^T}{||\mathbf{w}||}(\mathbf{x} - \mathbf{x}_0) = \tag{1}$$

$$\frac{1}{||\mathbf{w}||}(\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\mathbf{x}_0) = \frac{1}{||\mathbf{w}||}(\mathbf{w}^T\mathbf{x} + b) \tag{2}$$

Perceptron Learning Algorithm as Gradient Descent

Objective: Find a separating hyperplane that correctly classifier all input patterns.

- There are two types of error:

$$y_i = 1 \text{ and } \mathbf{w}^T \mathbf{x}_i + b < 0$$

$$y_i = -1 \text{ and } \mathbf{w}^T \mathbf{x}_i + b > 0$$

- Thus, for all misclassified points:

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) < 0$$

- To reduce the number of misclassified points, the goal is to minimize:

$$D(\mathbf{w}, b) = -\sum_{i \in M} y_i(\mathbf{w}^T \mathbf{x}_i + b)$$

  where M indexes the set of misclassified points.

Perceptron Learning Algorithm as Gradient Descent

Objective: Find a separating hyperplane that correctly classifier all input patterns.

- To minimize $D(\mathbf{w}, b)$ we can perform gradient descent over the surface represented by $D(\mathbf{w}, b)$ in parameter space. We iteratively move along the opposite direction of the gradient till a minimum is reached.
- The gradient is given by:

$$\frac{\partial D(\mathbf{w}, b)}{\partial \mathbf{w}} = -\sum_{i \in M} y_i \mathbf{x}_i$$

$$\frac{\partial D(\mathbf{w}, b)}{\partial b} = -\sum_{i \in M} y_i$$

- Update rule for parameters ("steepest descent"):

$$\mathbf{w}' = \mathbf{w} + \sum_{i \in M} y_i \mathbf{x}_i$$

$$b' = b + \sum_{i \in M} y_i$$

Perceptron Learning Algorithm as Gradient Descent

- In practice, “Stochastic Gradient Descent (SGD)” is used: rather than computing the sum of the gradient contributions of each x_i, followed by a step in the negative gradient direction, a step is taken after each observation is visited. The resulting update rule for parameters is:

$$\mathbf{w}' = \mathbf{w} + y_i \mathbf{x}_i$$

$$b' = b + y_i$$

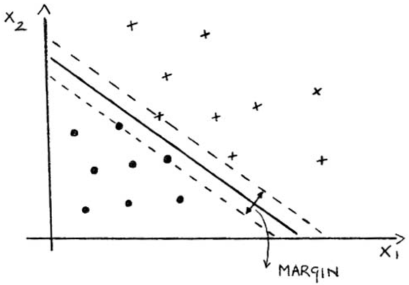
Problems with Perceptron Learning Algorithm

- When the data are linearly separable, there are many solutions, and which one is found depends on the starting values of the parameters.
- The finite number of steps to convergence can be very large: the smaller the gap between the two classes, the longer the time to find it.
- When the data are not separable, the algorithm will **not converge**, and cycles develop. The cycles can be long, and therefore hard to detect. We can provide a solution to the first problem by adding additional constraints to the separating hyperplane we want to find!

Largest Margin Hyperplanes

Goal: Find the hyperplane that separates the two classes and maximizes the distance to the closest points from either class.

- Such distance is called margin.



- The added constraint:
  - Provide a unique solution to the separating hyperplane problem;
  - Maximizing the margin between the two classes on the training data gives better classification performance on test data.

The Training Data

For two classes:

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)$$

$$\mathbf{x}_i \in \mathbf{R}^d$$
$$y_i = \{-1, +1\}$$

We need to formalize the largest margin criterion.

Formulation

Consider the following optimization problem:

$$\max_{\mathbf{w},b} 2C$$

$$\text{subject to } \frac{1}{||\mathbf{w}||} y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq C \quad i = 1, ..., N$$

- Remember Property 3: The signed distance of any point $\mathbf{x}$ to L is:

$$\frac{1}{||\mathbf{w}||}(\mathbf{w}^T\mathbf{x}_i + b)$$

- Thus, the set of conditions above ensure that all the training data are at least at distance C from the decision boundary.
- We seek the largest such C and associated parameters.
- We can rewrite the above conditions as:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq C||\mathbf{w}||$$

Formulation

- We can rewrite the above conditions as:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq C||\mathbf{w}||$$

- Since $\mathbf{w}^T\mathbf{x} + b = 0$ and $c(\mathbf{w}^T\mathbf{x} + b) = 0$ define the same plane. we can arbitrarily normalize $||\mathbf{w}|| = \frac{1}{C}$
- The original maximization problem is equivalent to:

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad i = 1, ..., N$$

- The constraints define an empty margin around the linear decision boundary of thickness $\frac{2}{||\mathbf{w}||}$. We choose $\mathbf{w}, b$ to maximize its thickness.
- This is a quadratic (convex) optimization problem subject to linear constraints and there is a unique minimum

Lagrange Multipliers

- We introduce Lagrange multipliers, which allow to take the constraints within the function to be minimized. Two reasons for doing this:
  1. The constraints will be replaced by constraints on the Lagrange multipliers themselves, which are easier to handle.
  2. In the new formulation of the problem, the training data will only appear (in the actual training and test algorithms) in the form of dot products between vectors. This is a crucial property which will allow us to generalize the procedure to the nonlinear case.
- We introduce the Lagrange multipliers $\alpha_i \geq 0, \ i = 1, \ldots, N$, one for each of the inequality constraints.
- Recall the rule: for constraints of the form $C_i \geq 0$, the constraint equations are multiplied by Lagrange multipliers and subtracted from the objective function, to form the Lagrangian.

Lagrange Multipliers

- We then obtain the Lagrangian: (also called "primal form"):

$$L_p = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{N} \alpha_i [y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1]$$

- We must now $\underline{\text{minimize } L_p}$ with respect to **w** and b:

$$\min_{\mathbf{w},b} = L_p$$

- Setting the derivatives to zero gives:

$$\frac{\partial L_p}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \tag{3}$$

$$\frac{\partial L_p}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{4}$$

Primal Form

- This indicates that this is the primal form of the optimization problem.
- We will actually solve the optimization problem by solving for the dual of the original problem
- The solution to the dual form provides a **lower bound** to the solution of the primal form
- What is the dual form?

Dual Form

Substituting eq. 3 and 4 in L_p gives:

$$\begin{aligned}
L_D &= \frac{1}{2}\Big(\sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i\Big)\Big(\sum_{k=1}^{N}\alpha_k y_k \mathbf{x}_k\Big) - \sum_{i=1}^{N}\alpha_i\Big[y_i\Big(\mathbf{x}_i^T\big(\sum_{k=1}^{N}\alpha_k y_k \mathbf{x}_k\big) + b\Big) - 1\Big] \\
&= \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N}\alpha_i\alpha_k y_i y_k \mathbf{x}_i^T\mathbf{x}_k - \sum_{i=1}^{N}\sum_{k=1}^{N}\alpha_i\alpha_k y_i y_k \mathbf{x}_i^T\mathbf{x}_k - b\underbrace{\sum_{i=1}^{N}\alpha_i y_i}_{=0} + \sum_{i=1}^{N}\alpha_i \\
&= \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N}\alpha_i\alpha_k y_i y_k \mathbf{x}_i^T\mathbf{x}_k
\end{aligned}$$

subject to $\alpha_i \geq 0$

The Lagrangian Dual Form

- The solution is obtained by maximizing L_D with respect to the α_i .
- The solution must satisfy the conditions:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$$

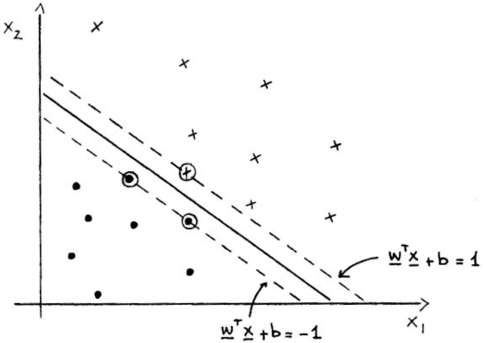
$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

$$\alpha_i \left[y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right] = 0 \quad \forall i = 1, ..., N$$

Dual Form

- If $\alpha_i > 0$, then $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$, that is $\mathbf{x}_i$ is on the boundary of the margin.
- If $y_i(\mathbf{w}^T\mathbf{x}_i + b) > 1$, $\mathbf{x}_i$ is not on the boundary of the margin, and $\alpha_i = 0$



x_2
x_1
$\underline{w}^T\underline{x} + b = 1$
$\underline{w}^T\underline{x} + b = -1$

Dual Form

- The solution vector $\mathbf{w}$ is: $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$. Thus: The solution is defined as a linear combination of those $\mathbf{x}_i$ for which $\alpha_i > 0$. Such $\mathbf{x}_i$ are the points on the boundary of the margin. They are called <u>SUPPORT VECTORS</u>. We have three support vectors in the above example.
- To obtain the value of b: solve $\alpha_i \left[y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right] = 0$ for any of the support vectors.
- The largest margin hyperplane gives a function: $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$ for classifying new observations $\hat{y} = \text{sign}(f(\mathbf{x}))$

Observations

- The support vectors are the critical elements of the training set. They lie closet to the decision boundary.
- Only the support vectors affect the solution - If all other training points were removed (or moved around, but so as not to cross the margin), and training was repeated, the same separating hyperplane would be found.
- However, the identification of the support vectors requires the use of all the training data.
- Although none of the training observations fall within the margin (by construction), this will not necessarily be the case of test data. (The intuition is that a large margin on the training data indicates a good separation of the two classes and therefore a good separation on the test data as well)

Summary so far

- Training data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)$, $\mathbf{x}_i \in \mathbb{R}^d, y_i \in \{-1, +1\}$
- When the two classes are linearly separable, we can find a function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ with $y_i f(\mathbf{x}_i) > 0\ \forall i$
- In particular, we can find the hyperplane that creates the largest margin between the training points.
- The optimization problem captures this concept

$$\max_{\mathbf{w},b} 2C$$

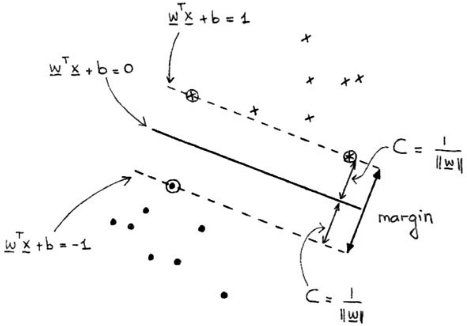
$$\text{subject to } \frac{1}{||\mathbf{w}||} y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq C \quad i = 1, ..., N$$

- It can be more conveniently rewritten as below where $C = \frac{1}{||\mathbf{w}||}$

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad i = 1, ..., N$$

Geometric perspective


$\underline{w}^T\underline{x}+b=1$
$\underline{w}^T\underline{x}+b=0$
$\underline{w}^T\underline{x}+b=-1$
$C=\frac{1}{\|\underline{w}\|}$
margin
$C=\frac{1}{\|\underline{w}\|}$

The Non-separable Case

- Suppose now the classes overlap. We can still maximize C, but allow for some points to be on the wrong side of the margin.
- We need to modify the constraints we had for the separable case: $\frac{1}{||\mathbf{w}||} y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq C \quad i = 1, ..., N$.
- To achieve this goal, we define N slack variables:

$$\xi_1, \xi_2, ..., \xi_N$$

- Then a natural way to modify the constraints above is:

$$\frac{1}{||\mathbf{w}||} y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq C(1 - \xi_i) \quad i = 1, ..., N$$

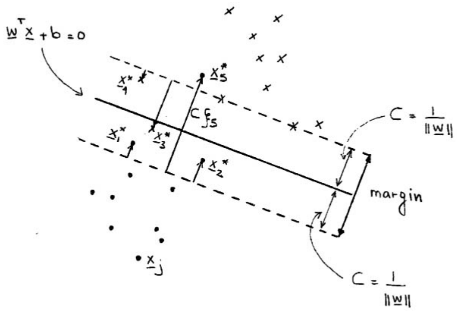
$$\text{with } \xi_i \geq 0 \quad \forall i, \quad \sum_{i=1}^{N} \xi_i \leq \text{constant}$$

The Non-separable Case

- Idea of the formulation: ξ_i is the proportional amount by which the prediction $f(\mathbf{x}_i)$ is on the wrong side of the margin.
- Note: $C(1 - \xi_i) = C - C\xi_i$

Slack Variables

- A geometric perspective:



- The points $(\mathbf{x}_1^*, \mathbf{x}_2^*, \mathbf{x}_3^*, \mathbf{x}_4^*, \mathbf{x}_5^*)$ are on the wrong side of their margin.
- Point $\mathbf{x}_i^*$ is on the wrong side of its margin by an amount $C\xi_i$
- Point $\mathbf{x}_j^*$ on the correct side have $\xi_j = 0$
- Misclassification occurs when $\xi_i > 1 \Rightarrow C(1-\xi_i) < 0$, e.g., points $\mathbf{x}_3^*$ and $\mathbf{x}_5^*$ are misclassified by the given boundary.

Slack Variables (2)

- The condition $\sum_{i=1}^{N} \xi_i \leq$ constant bounds the sum $\sum_i \xi_i$
- Thus, it bounds the total proportional amount by which predictions fall on the wrong side of their margin.
- Since misclassification occur when $\xi_i > 1$ (in this case $y_i f(\mathbf{x}_i) < 0$), bounding $\sum_{i=1}^{N} \xi_i \leq k$, bounds the total number of training misclassifications at k.
- So, for the non-separable case, we have the optimization problem:

$$\max_{\mathbf{w},b} 2C$$

$$\text{subject to: } \frac{1}{||\mathbf{w}||} y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq C(1 - \xi_i) \quad i = 1, ..., N$$

$$\text{with } \xi_i \geq 0 \ \ \forall i, \quad \sum_{i=1}^{N} \xi_i \leq \text{constant}$$

Slack Variables (3)

- As for the separable case, we define $C = \frac{1}{||\mathbf{w}||}$ and rewrite the above maximization problem in the equivalent form:

$$\min_{\mathbf{w},b} \frac{||\mathbf{w}||^2}{2}$$

$$\text{subject to: } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad i = 1, ..., N$$

$$\text{with } \xi_i \geq 0 \ \ \forall i, \quad \sum_{i=1}^{N} \xi_i \leq \text{constant}$$

- We have obtained a quadratic optimization problem with linear constraints. We will solve it using Lagrange multipliers.

Lagrange Multipliers for Slack Variables

- First, one more step: we have seen that the condition $\sum_i \xi_i \leq$ constant, bounds the number of training misclassifications.
- We can incorporate this condition into the objective function by adding an extra cost for errors:

$$\min_{\mathbf{w},b} \frac{||\mathbf{w}||^2}{2} + \gamma \sum_{i=1}^{N} \xi_i$$

$$\text{subject to: } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad i = 1, ..., N$$

$$\text{with } \xi_i \geq 0 \;\; \forall i, \quad \sum_{i=1}^{N} \xi_i \leq \text{constant}$$

here, γ is a parameter to be chosen by the user. A larger γ corresponds to assigning a higher penalty to errors.

Lagrange Multipliers for Slack Variables

$$\min_{\mathbf{w},b} \frac{||\mathbf{w}||^2}{2} + \gamma \sum_{i=1}^{N} \xi_i$$

$$\text{subject to: } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad i = 1, ..., N$$

$$\text{with } \xi_i \geq 0 \ \ \forall i, \quad \sum_{i=1}^{N} \xi_i \leq \text{constant}$$

- Introducing the Lagrange multipliers α_i and μ_i (one for each constraint), gives the following Lagrange (primal) function:

$$L_p = \frac{1}{2}||\mathbf{w}||^2 + \gamma \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i \left[y_i(\mathbf{w}^T\mathbf{x}_i + b) - (1 - \xi_i) \right] - \sum_{i=1}^{N} \mu_i \xi_i$$

- Our objective is:

$$\min_{\mathbf{w},b,\xi_i} L_p$$

Lagrange Multipliers Solution

- Setting the respective derivatives to zero gives:

$$\frac{\partial L_p}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \tag{5}$$

$$\frac{\partial L_p}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{6}$$

$$\frac{\partial L_p}{\partial \xi_i} = \gamma - \alpha_i - \mu_i \quad \forall i \Rightarrow \alpha_i = \gamma - \mu_i = 0 \quad \forall i \tag{7}$$

along with the positivity constraints $\alpha_i, \mu_i, \xi_i \geq 0 \quad \forall i$

- Substituting eq. 5, 6, 7 in L_p, we obtain the so called <u>dual objective function</u>:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

where L_D gives a lower bound on the objective function $\frac{1}{2}||\mathbf{w}||^2 + \gamma \sum_{i=1}^{N} \xi_i$

Deriving the Dual Form

$$\frac{1}{2}(\sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i)^T(\sum_{j=1}^{N}\alpha_j y_j \mathbf{x}_j) + \gamma\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i^T \mathbf{w}^T - \sum_{i=1}^{N}\alpha_i y_i b + \sum_i \alpha_i(1-\xi_i) - \sum_{i=1}^{N}\mu_i\xi_i$$

$$= \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j - \sum_i\sum_j \alpha_i\alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j + \gamma\sum_{i=1}^{N}\xi_i - b\underbrace{\sum_{i=1}^{N}\alpha_i y_i}_{=0} + \sum_i \alpha_i(1-\xi_i) - \sum_{i=1}^{N}\mu_i\xi_i$$

$$= -\frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j + \gamma\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\alpha_i - \sum_i^{N}\alpha_i\xi_i - \sum_i^{N}\mu_i\xi_i$$

$$= \sum_i^{N}\alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j + \underbrace{\sum_i \overbrace{(\gamma-\mu_i)}^{=\alpha_i}\xi_i - \sum_i \alpha_i\xi_i}_{=0}$$

$$= \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

Lagrange Multipliers Solution

- Thus: the solution is obtained by maximizing L_D w.r.t the α_i, subject to:

$$\sum_i^N \alpha_i y_i = 0, \quad 0 \le \alpha_i \le \gamma$$

- The solution must satisfy the conditions:

$$\mathbf{w} = \sum_i^N \alpha_i y_i \mathbf{x}_i \tag{8}$$

$$\sum_i^N \alpha_i y_i = 0 \tag{9}$$

$$\alpha_i = \gamma - \mu_i, \quad \forall i \tag{10}$$

$$\alpha_i \left[y_i(\mathbf{w}^T \mathbf{x}_i + b) - (1 - \xi_i) \right] = 0, \quad \forall i \tag{11}$$

$$\mu_i \xi_i = 0 \quad \forall i \tag{12}$$

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) - (1 - \xi_i) \ge 0 \quad \forall i \tag{13}$$

Lagrange Multipliers Solution

- From 8, the solution is $\mathbf{w} = \sum_i^N \alpha_i y_i \mathbf{x}_i$. From 11, $\alpha_i > 0$ when constraint 13 are exactly met, i.e., $y_i(\mathbf{w}^T \mathbf{x}_i + b) - (1 - \xi_i) = 0$
- The points $(\mathbf{x}_i)$ with $\alpha_i > 0$ are the <u>SUPPORT VECTORS</u>.
- We have two kinds of support vectors:
  - Those for which $\xi_i = 0$: they lie on the edge of the margin. From 12 and 10: $0 < \alpha_i < \gamma$
  - Those for which $\xi_i > 0$: they have $\alpha_i = \gamma$ and they lie on the wrong side of their margin.
- To estimate b, we can use 11 with any of the support vectors with $\xi_i = 0$.
- Once we have $\mathbf{w}$ and b, the decision function can be written:

$$\hat{y} = \text{sign}(f(\mathbf{x})) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$$

- The tuning parameter of this procedure is γ. Its optimal value can be estimated via cross validation.

Optimization

- A constrained optimization problem over **w** and ξ

$$\min_{\mathbf{w},b} \frac{||\mathbf{w}||^2}{2} + \gamma \sum_{i=1}^{N} \xi_i \tag{14}$$

$$\text{subject to: } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad i = 1, ..., N \tag{15}$$

- The constraint can be written more concisely as:

$$y_i f(\mathbf{x}_i) \geq 1 - \xi_i \tag{16}$$

together with $\xi_i > 0$ is equivalent to

$$\xi = max(0, 1 - y_i f(\mathbf{x}_i)) \tag{17}$$

- Hence the learning problem is equivalent to the unconstrained optimization problem over **w**:

$$\min_{\mathbf{w},b} \frac{||\mathbf{w}||^2}{2} + \gamma \sum_{i=1}^{N} max(0, 1 - y_i f(\mathbf{x}_i)) \tag{18}$$

Loss function

$$\min_{\mathbf{w},b} \frac{||\mathbf{w}||^2}{2} + \gamma \sum_{i=1}^{N} \underbrace{\max(0, 1 - y_i f(\mathbf{x}_i))}_{\text{Hinge loss}} \tag{19}$$

- $y_i f(\mathbf{x}_i) > 1$: points outside margin. No contribution to loss
- $y_i f(\mathbf{x}_i) = 1$: points on margin. No contribution to loss (hard margin case)
- $y_i f(\mathbf{x}_i) < 1$: points violates margin constraints. Contribute to loss.

Hinge Loss

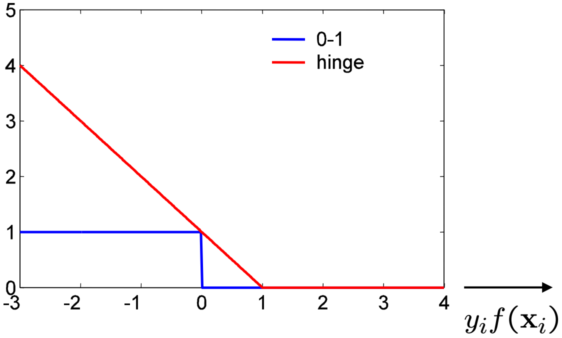


Figure: Hinge loss vs 0-1 loss

An approximation to the 0-1 loss. Convex?

Implementation

- Solving the Quadratic Programming Problems (slow)
- Use an interior point method that uses Newton-like iterations to find a solution of the KarushKuhnTucker conditions of the primal and dual problems
- Platt's sequential minimal optimization (SMO) algorithm
- Stochastic sub-gradient descent algorithms.

Non-linear SVM

- How can the above methods be generalized to the case where the decision function is not a linear function of the data?
- It turns out that the generalization to a nonlinear boundary can be accomplished in a straightforward way using a simple mathematical trick!
- One major observation (look at the dual objective function obtained earlier):

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j \tag{20}$$

$$= \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \underbrace{< \mathbf{x}_i, \mathbf{x}_j >}_{\text{dot product}} \tag{21}$$

- The only way in which the data appear in the training problem is in the form of dot products.

Non-linear SVM

- How about the solution function?
- From $\mathbf{w} = \sum_i^N \alpha_i y_i \mathbf{x}_i$, the solution function can be written as:

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{w}^T \mathbf{x} + b \\ &= \sum_i^{N_s} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b \\ &= \sum_i^{N_s} \alpha_i y_i < \mathbf{x}_i, \mathbf{x} > +b \end{aligned}$$

where N_s is the number of support vectors.
$\Rightarrow$Also in the solution function, the data appear in the form of dot products where the $(\mathbf{x}_i)$s are the support vectors.

Non-linear SVM

- Now, suppose we first map the data to some high dimension Euclidean space using a mapping Φ:

$$\Phi : \mathbb{R}^d \to \mathbb{R}^h$$

  usually $h > d$

- The idea is to enlarge the input space to achieve better training class separation.
- In general, linear boundaries in the enlarged space translate to nonliear boundaries in the original space (true for any nonlinear mapping Φ)

Mapping

- Then, we compute the largest margin hyperplane in the new space $\mathbb{R}^h$.
- Of course, the training algorithm would only depend on the data through dot products in $\mathbb{R}^h$, i.e., $< \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) >$ where $\Phi(\mathbf{x}_i) \in \mathbb{R}^h$.
- Suppose we have a function (called kernel function) K that computes such dot products in the transformed space:

$$K(\mathbf{x}_i, \mathbf{x}_j) =< \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) >$$

- Then: all we need in the training algorithm is K, and we would never need to explicitly even know what Φ is.
- Resulting procedure: replace $< \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) >$ with $K(\mathbf{x}_i, \mathbf{x}_j)$ everywhere in the training algorithm.
- Example of K: $K(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}}$ (Gaussian kernel)

Mapping

- The algorithm constructs a linear support vector machine in $\mathbb{R}^h$.
- It achieves this objective in roughly the same amount of time it would take to train on the un-mapped data.
- How can we use such a machine? In test phase, given the test points $\mathbf{x}$:

$$\begin{aligned} f(x) &= \sum_{i}^{N_s} \alpha_i y_i < \mathbf{x}_i, \mathbf{x} > + b \\ &= \sum_{i}^{N_s} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \end{aligned}$$

  where $\mathbf{x}_i$ are the support vectors and N_s is the number of support vectors.
- The fact that, through the kernel function K, we can work with vectors in input space, without even knowing the mapping function Φ is known as the "kernel trick".

Example - kernel functions

Example: an allowed kernel for which we can construct the mapping Φ

- Training data are vectors in $\mathbb{R}^2$.
- Suppose we choose $K(\mathbf{x}_i, \mathbf{x}_j) = (<\mathbf{x}_i, \mathbf{x}_j>)^2$.
- We can find a mapping

$$\Phi : \mathbb{R}^2 \to \mathbb{R}^h$$

such that $(<\mathbf{x}_i, \mathbf{x}_j>)^2 = <\Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)>$

- One such mapping is:

$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$

defined as

$$\Phi(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

where $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

Example - kernel functions

- We can verify that this is indeed the case:

$$\begin{aligned} K(\mathbf{x},\mathbf{y}) &= (\mathbf{x},\mathbf{y})^2 = (x_1y_1 + x_2y_2)^2 \\ &= x_1^2y_1^2 + x_2^2y_2^2 + 2x_1y_1x_2y_2 \end{aligned}$$

$$\begin{aligned} <\Phi(\mathbf{x}),\Phi(\mathbf{y})> &= \Phi(\mathbf{x})^T\Phi(\mathbf{y}) \\ &= (x_1^2, \sqrt{2}x_1x_2, x_2^2)\begin{pmatrix} y_1^2 \\ \sqrt{2}y_1y_2 \\ y_2^2 \end{pmatrix} \\ &= x_1^2y_1^2 + x_2^2y_2^2 + 2x_1y_1x_2y_2 \end{aligned}$$

- Note: in general neither the mapping Φ nor the space $\mathbb{R}^h$ are unique for a given kernel.

Exercise - kernel functions

- You can verify the following two mapping Φ also satisfy $K(\mathbf{x}, \mathbf{y}) =< \Phi(\mathbf{x}), \Phi(\mathbf{y}) >$ for kernel given above.
- Example 1

$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$\Phi(\mathbf{x}) = \frac{1}{\sqrt{2}} \begin{pmatrix} (x_1^2 - x_2^2) \\ 2x_1x_2 \\ (x_1^2 + x_2^2) \end{pmatrix}$$

- Example 2

$$\Phi : \mathbb{R}^2 \to \mathbb{R}^4$$
$$\Phi(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ x_1x_2 \\ x_1x_2 \\ x_2^2 \end{pmatrix}$$

Which functions are allowable as kernels?

- The mathematical properties that a function K must have so that a mapping Φ and an expansion $K(\mathbf{x},\mathbf{y}) = <\Phi(\mathbf{x}),\Phi(\mathbf{y})>$ exist, have been studied and are well known (Mercer theorem)
- Two popular choices for K are:
  - d^{th} degree polynomial: $K(\mathbf{x},\mathbf{y}) = (1+<\mathbf{x},\mathbf{y}>)^d$
  - Radial basis: $K(\mathbf{x},\mathbf{y}) = e^{\frac{-||\mathbf{x}_i-\mathbf{x}_j||^2}{2\sigma^2}}$

Importance of Parameter γ for Non-linear SVMs

- Perfect separation of training data can in general be achieved in the enlarged space $\mathbb{R}^h$.
- Such perfect separation leads to the danger of overfitting the data. As a consequence, the classifier will generalize poorly.
- A proper setting of γ allows to avoid overfitting.
- Lets look at the objective function minimized by an SVM: $\frac{1}{2}||\mathbf{w}||^2 + \gamma \sum_i^N \xi_i$ where γ is the penalty factor for errors.
- Large γ $\rightarrow$ discourage any positive ξ_i $\rightarrow$ tendency to overfit the data $\rightarrow$ highly complicated decision boundary in input space.
- Small γ $\rightarrow$ encourage a small value of $||\mathbf{w}||$ $\rightarrow$ larger margin $\rightarrow$ more data on the wrong side of their margin $\rightarrow$ smoother decision boundary in input space.
- In practice: we need to tune γ so to achieve best test error performance.

Limitations and Open Issues

- Choice of the kernel
  - performance may depend on the chosen kernel, and on the values of associated parameters;
  - the best choice of a kernel for a given problem is still a research issue; (e.g., Latent Semantic Kernel for document classification)
- Speed and size for training and testing
  - Training: the evaluation of the dual objective function requires the computation of all dot products $< \mathbf{x}_i, \mathbf{x}_j > \Rightarrow$ time complexity $O(N^2 d)$ where N is the number of training data and d is the dimension.
  - Testing: need to evaluate: $f(\mathbf{x}) = \sum_i^{N_s} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$.
    - $\Rightarrow$ time complexity $O(M \cdot N_s)$ where M is the number of operation required to evaluate the kernel.
    - $\Rightarrow$ time complexity $O(d \cdot N_s)$ when K is a RBF kernel, $M = O(d)$.
- Extension to multiple classes

Acknowledgements

Slides adapted from Dr. Carlotta Domeniconi's *Pattern Recognition* at George Mason University.