CS559 Machine Learning
Support Vector Machine

Tian Han

Department of Computer Science
Stevens Institute of Technology

Week 14

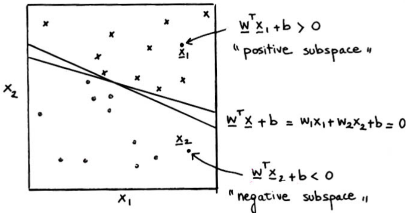
Outline

- Linear classifier, large margin, SVM
- Non-separable case, slack variable
- Non-linearity, kernels

Linear classifier, large margin, SVM

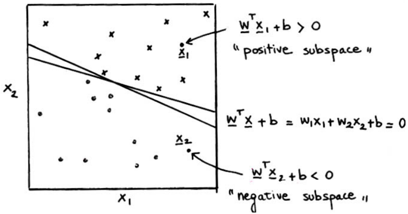
Linear Classifier

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



Linear Classifier

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



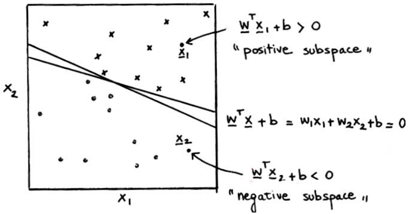
- Classification Rule (**the Perceptron model**):

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

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

Linear Classifier

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



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

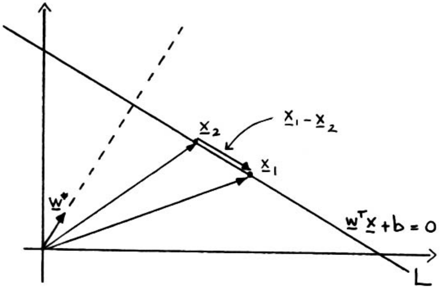
$$\text{Input: } \mathbf{x} \in \mathbb{R}^d$$

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

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

Some Vector Algebra

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



Property 1

- Consider any two points x_1, x_2, lying on hyperplane L:

$$\begin{matrix} \mathbf{w}^T\mathbf{x}_1 + b = 0 \\ \mathbf{w}^T\mathbf{x}_2 + b = 0 \end{matrix} \rightarrow \mathbf{w}^\top(\mathbf{x}_1 - \mathbf{x}_2) = 0$$

Property 1

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

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

is the vector normal to the surface of L.

Property 1

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

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

  is the vector normal to the surface of L.
- Note 1: All vectors here are column vectors.

Property 1

- Consider any two points x_1, x_2, lying on hyperplane L:
  $$\begin{matrix} \mathbf{w}^T\mathbf{x}_1 + b = 0 \\ \mathbf{w}^T\mathbf{x}_2 + b = 0 \end{matrix} \rightarrow \mathbf{w}^\top(\mathbf{x}_1 - \mathbf{x}_2) = 0$$
- Since $\mathbf{w}^\top(\mathbf{x}_1 - \mathbf{x}_2) = \mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) = 0$, the two vectors $\mathbf{w}$ and $\mathbf{x}_1 - \mathbf{x}_2$ are orthogonal vectors.
  $$\mathbf{w}^* = \frac{\mathbf{w}}{||\mathbf{w}||}$$
  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 $\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^\top\mathbf{b} = ||\mathbf{a}|| \times ||\mathbf{b}|| \times \cos\alpha$ where α is the angle between a and b.

Property 2

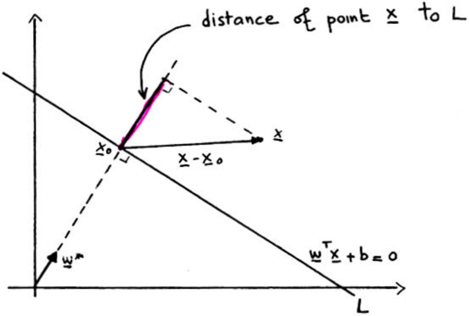
- For any point $\mathbf{x}_0$ on L:

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

Thus:

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

Property 3



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

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

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

Recall: perceptron classifier

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

- There are two types of error:

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

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

- Thus, for all misclassified points:

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

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

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

where M indexes the set of misclassified points.

Problems with perceptron classifier

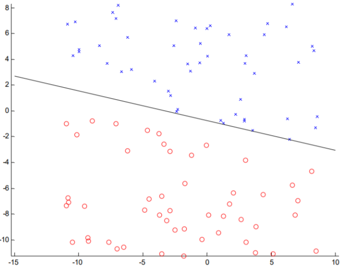


Figure: [A. Zisserman, C19, 2015]

- If the data is linearly separable, then the algorithm will converge.

Problems with perceptron classifier

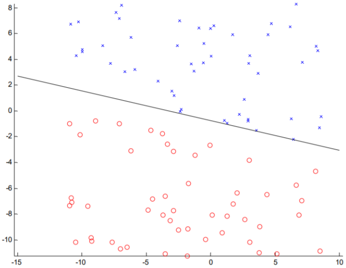


Figure: [A. Zisserman, C19, 2015]

- If the data is linearly separable, then the algorithm will converge.
- Convergence can be slow.

Problems with perceptron classifier

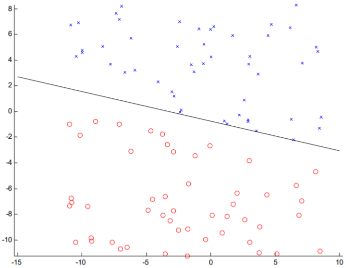


Figure: [A. Zisserman, C19, 2015]

- If the data is linearly separable, then the algorithm will converge.
- Convergence can be slow.
- Separating line close to training data.

Problems with perceptron classifier

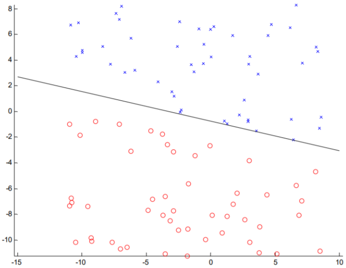


Figure: [A. Zisserman, C19, 2015]

- If the data is linearly separable, then the algorithm will converge.
- Convergence can be slow.
- Separating line close to training data.
- Prefer a larger margin for better generalization.

What's the best $\mathbf{w}$

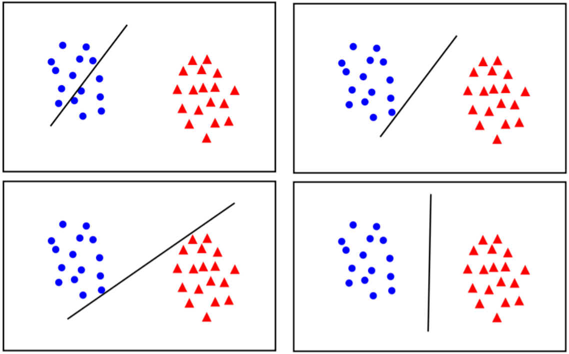


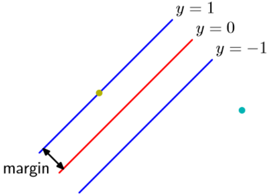
Figure: [A. Zisserman, C19, 2015]

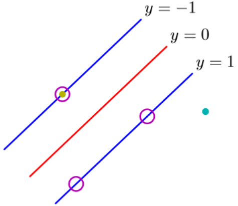
Maximum margin solution: most stable under perturbations of the inputs.

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.

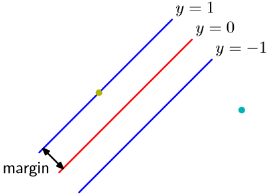


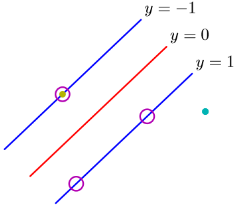


Largest Margin Hyperplanes

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.



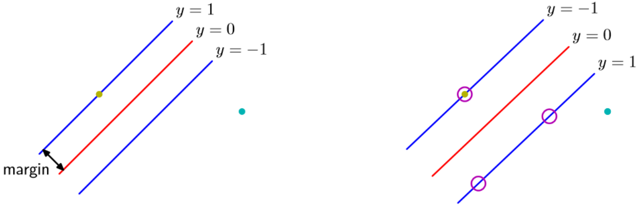


- The added constraint:

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.

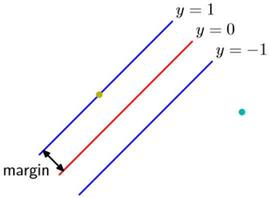


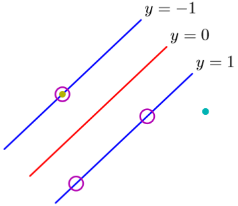
- The added constraint:
  - Provide a unique solution to the separating hyperplane problem;

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:

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

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

Formulation

Consider the following optimization problem:

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

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

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

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

Formulation

Consider the following optimization problem:

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

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

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

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

- Interested in solutions that all data point $\mathbf{x}_i$ are correctly classified.

Formulation

Consider the following optimization problem:

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

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

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

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

- Interested in solutions that all data point $\mathbf{x}_i$ are correctly classified.
- The set of conditions above ensure that all the training data are at least at distance C from the decision boundary.

Formulation

Consider the following optimization problem:

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

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

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

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

- Interested in solutions that all data point $\mathbf{x}_i$ are correctly classified.
- 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 C and associated parameters $\mathbf{w}$, b.

Formulation

Consider the following optimization problem:

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

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

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

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

- Interested in solutions that all data point $\mathbf{x}_i$ are correctly classified.
- 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 C and associated parameters $\mathbf{w}$, b.
- We can rewrite the above conditions as:

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

Formulation

- We can rewrite the above conditions as:

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

Formulation

- We can rewrite the above conditions as:

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

- Since $\mathbf{w}^\top \mathbf{x} + b = 0$ and $c(\mathbf{w}^\top \mathbf{x} + b) = 0$ define the same plane. we can arbitrarily normalize $||\mathbf{w}|| = \frac{1}{C}$

Formulation

- We can rewrite the above conditions as:

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

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

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

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

Formulation

- We can rewrite the above conditions as:

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

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

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

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

- The constraints define a margin around the linear decision boundary of thickness $\frac{2}{||\mathbf{w}||}$. We choose $\mathbf{w}, b$ to maximize its thickness.

Formulation

- We can rewrite the above conditions as:

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

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

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

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

- The constraints define a 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

- Lagrange multiplier allows to take the constraints within the function to be minimized (Recall we briefly introduced Lagrange multipliers in PCA and FLD). 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.

Lagrange Multipliers

- Lagrange multiplier allows to take the constraints within the function to be minimized (Recall we briefly introduced Lagrange multipliers in PCA and FLD). 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 $\alpha_i \geq 0, \ i = 1, ..., N$, one for each of the inequality constraints.

Lagrange Multipliers

- Lagrange multiplier allows to take the constraints within the function to be minimized (Recall we briefly introduced Lagrange multipliers in PCA and FLD). 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 $\alpha_i \geq 0,\ i = 1, ..., N$, 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"):

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

Lagrange Multipliers

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

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

- We must now <u>minimize L_p</u> with respect to $\mathbf{w}$ and b:

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

Lagrange Multipliers

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

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

- We must now <u>minimize L_p</u> with respect to $\mathbf{w}$ and b:

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

- Setting the derivatives to zero gives:

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

$$\frac{\partial L_p}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0 \qquad (2)$$

Primal Form

- This is the primal form of the optimization problem.
- We could also solve the optimization problem by solving for the dual of the original problem
- What is the dual form?

Dual Form

Substituting eq. 1 and 2 in L_p gives:

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

subject to $\alpha_i \geq 0, \sum_{i=1}^{N}\alpha_i y_i = 0$

The Lagrangian Dual Form

- The solution is obtained by maximizing L_D with respect to the α_i, i.e., $\max_\alpha \min_{\mathbf{w},b} L_p$.

The Lagrangian Dual Form

- The solution is obtained by maximizing L_D with respect to the α_i, i.e., $\max_\alpha \min_{\mathbf{w},b} L_p$.
- It can be shown that the solution must satisfy the conditions:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$$
$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{3}$$

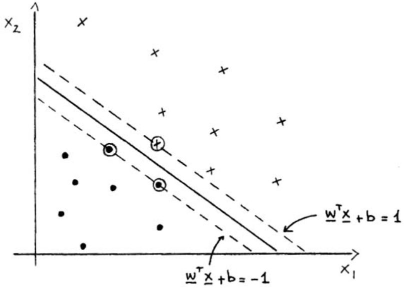
And the Karush-Kuhn-Tucker (KKT) conditions:

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

The complementary slackness

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

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



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 **SUPPORT VECTORS**. We have three support vectors in the above example.
- To obtain the value of b: solve $\alpha_i \left[y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 \right] = 0$ for any of the support vectors.
- The largest margin hyperplane gives a function:
  $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ 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 closest to the decision boundary.
- Only the support vectors affect the prediction.
- 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)

Non-separable case

Summary for linear separable case

- 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 $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ 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

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

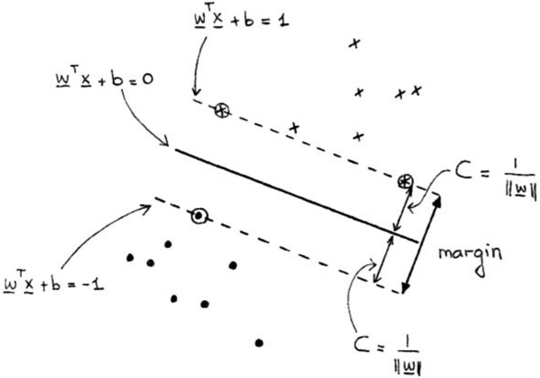
$$\text{subject to } \frac{1}{||\mathbf{w}||} y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq C \quad i = 1, ..., N$$

- It can be more conveniently rewritten as below

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

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

Geometric perspective



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

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

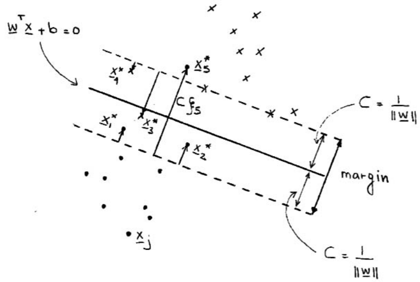
with $\xi_i \geq 0 \quad \forall i$

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 $C\xi_i$
- $0 < \xi_i \leq 1$: inside the margin, correct side of hyperplane. $C\xi_i \leq C$, Margin Violation
- $\xi_i > 1$: $C\xi_i > C$, misclassification

Slack Variables–soft margin

- We normalize $\mathbf{w}$ and consider soft-margin version:

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

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

with $\xi_i \geq 0 \quad \forall i$

- Goal: maximize the margin while softly penalizing points that lie on the wrong side of margin boundary.

Slack Variables–soft margin

- We normalize $\mathbf{w}$ and consider soft-margin version:

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

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

with $\xi_i \geq 0 \quad \forall i$

- Goal: maximize the margin while softly penalizing points that lie on the wrong side of margin boundary.
- Since misclassification occur when $\xi_i > 1$, $\sum_{i=1}^{N} \xi_i$ bounds the total number of training misclassifications.

Slack Variables–soft margin

- We normalize $\mathbf{w}$ and consider soft-margin version:

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

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

with $\xi_i \geq 0 \quad \forall i$

- Goal: maximize the margin while softly penalizing points that lie on the wrong side of margin boundary.
- Since misclassification occur when $\xi_i > 1$, $\sum_{i=1}^{N} \xi_i$ bounds the total number of training misclassifications.
- γ is a parameter to be chosen by the user. A larger γ corresponds to assigning a higher penalty to errors.

Slack Variables–soft margin

- We normalize $\mathbf{w}$ and consider soft-margin version:

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

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

with $\xi_i \geq 0 \quad \forall i$

- Goal: maximize the margin while softly penalizing points that lie on the wrong side of margin boundary.
- Since misclassification occur when $\xi_i > 1$, $\sum_{i=1}^{N} \xi_i$ bounds the total number of training misclassifications.
- γ is a parameter to be chosen by the user. A larger γ corresponds to assigning a higher penalty to errors.
- We have obtained a quadratic optimization problem with linear constraints. We will solve it using Lagrange multipliers.

Lagrange Multipliers for Slack Variables

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

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

with $\xi_i \geq 0 \quad \forall i$

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

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

- Our objective is:

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

Lagrange Multipliers Solution

- Setting the respective derivatives to zero gives:

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

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

$$\frac{\partial L_p}{\partial \xi_i} = \gamma - \alpha_i - \mu_i \quad \forall i \Rightarrow \alpha_i = \gamma - \mu_i \quad \forall i \tag{6}$$

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

Lagrange Multipliers Solution

- Setting the respective derivatives to zero gives:

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

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

$$\frac{\partial L_p}{\partial \xi_i} = \gamma - \alpha_i - \mu_i \quad \forall i \Rightarrow \alpha_i = \gamma - \mu_i \quad \forall i \tag{6}$$

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

- Substituting eq. 4, 5, 6 in L_p, we obtain the so called dual objective function:

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

Lagrange Multipliers Solution

The solution is obtained by maximizing L_D w.r.t the α_i, subject to:(similar to the separable case, but the constraints are different)

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

Lagrange Multiplier Solution

It can be shown that the solution must satisfy the conditions:

$$\mathbf{w} = \sum_{i}^{N} \alpha_i y_i \mathbf{x}_i \tag{7}$$

$$\sum_{i}^{N} \alpha_i y_i = 0 \tag{8}$$

$$\alpha_i = \gamma - \mu_i, \quad \forall i \tag{9}$$

$$\tag{10}$$

Also the KKT conditions: (complementary slackness)

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

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

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

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

Lagrange Multipliers Solution

- From 7, 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., $y_i(\mathbf{w}^\top \mathbf{x}_i + b) - (1 - \xi_i) = 0$

Lagrange Multipliers Solution

- From 7, 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., $y_i(\mathbf{w}^\top \mathbf{x}_i + b) - (1 - \xi_i) = 0$
- The points $(\mathbf{x}_i)$ with $\alpha_i > 0$ are the SUPPORT VECTORS.

Lagrange Multipliers Solution

- From 7, 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., $y_i(\mathbf{w}^\top \mathbf{x}_i + b) - (1 - \xi_i) = 0$
- The points $(\mathbf{x}_i)$ with $\alpha_i > 0$ are the SUPPORT VECTORS.
- We have two kinds of support vectors:
  - If $\alpha_i < \gamma$, then $\mu_i > 0$, implies $\xi_i = 0$, such point $\mathbf{x}_i$ lie on the margin.
  - If $\alpha_i = \gamma$, then $\mu_i = 0$, then $\mathbf{x}_i$ can be correctly classified if $\xi_i \leq 1$ or misclassified if $\xi_i > 1$.

Lagrange Multipliers Solution

- From 7, 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., $y_i(\mathbf{w}^\top \mathbf{x}_i + b) - (1 - \xi_i) = 0$
- The points $(\mathbf{x}_i)$ with $\alpha_i > 0$ are the SUPPORT VECTORS.
- We have two kinds of support vectors:
  - If $\alpha_i < \gamma$, then $\mu_i > 0$, implies $\xi_i = 0$, such point $\mathbf{x}_i$ lie on the margin.
  - If $\alpha_i = \gamma$, then $\mu_i = 0$, then $\mathbf{x}_i$ can be correctly classified if $\xi_i \leq 1$ or misclassified if $\xi_i > 1$.
- To estimate b, we can use 11 with any of the support vectors with $\xi_i = 0$.

Lagrange Multipliers Solution

- From 7, 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., $y_i(\mathbf{w}^\top \mathbf{x}_i + b) - (1 - \xi_i) = 0$
- The points $(\mathbf{x}_i)$ with $\alpha_i > 0$ are the <u>SUPPORT VECTORS</u>.
- We have two kinds of support vectors:
  - If $\alpha_i < \gamma$, then $\mu_i > 0$, implies $\xi_i = 0$, such point $\mathbf{x}_i$ lie on the margin.
  - If $\alpha_i = \gamma$, then $\mu_i = 0$, then $\mathbf{x}_i$ can be correctly classified if $\xi_i \leq 1$ or misclassified if $\xi_i > 1$.
- 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:

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

Lagrange Multipliers Solution

- From 7, 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., $y_i(\mathbf{w}^\top \mathbf{x}_i + b) - (1 - \xi_i) = 0$
- The points $(\mathbf{x}_i)$ with $\alpha_i > 0$ are the <u>SUPPORT VECTORS</u>.
- We have two kinds of support vectors:
  - If $\alpha_i < \gamma$, then $\mu_i > 0$, implies $\xi_i = 0$, such point $\mathbf{x}_i$ lie on the margin.
  - If $\alpha_i = \gamma$, then $\mu_i = 0$, then $\mathbf{x}_i$ can be correctly classified if $\xi_i \leq 1$ or misclassified if $\xi_i > 1$.
- 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:

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

- Need to tune γ.

Optimization

- A constrained optimization problem over $\mathbf{w}$ and ξ

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

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

- The constraint can be written more concisely as $y_i f(\mathbf{x}_i) \geq 1 - \xi_i$, together with $\xi_i > 0$ is equivalent to

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

- Hence the learning problem is equivalent to the unconstrained optimization problem over $\mathbf{w}$:

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

Loss function

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

- $y_i f(\mathbf{x}_i) > 1$: 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

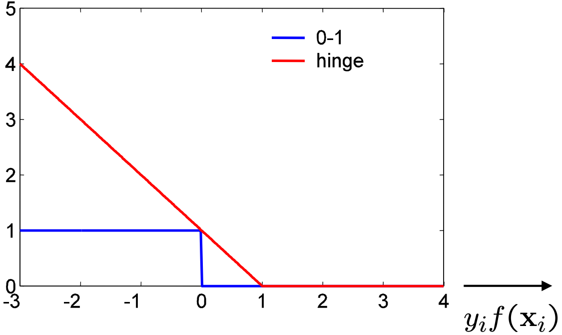


Figure: Hinge loss vs 0-1 loss

An approximation to the 0-1 loss.

Implementation

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

Primal vs dual formulation

- Primal problem: $\mathbf{w} \in R^{M-1}, b \in R$

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

Primal vs dual formulation

- Primal problem: $\mathbf{w} \in R^{M-1}, b \in R$

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

- Dual problem:

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \ s.b. \ \sum_{i}^{N} \alpha_i y_i = 0, \ 0 \leq \alpha_i \leq \gamma$$

Primal vs dual formulation

- Primal problem: $\mathbf{w} \in R^{M-1}, b \in R$

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

- Dual problem:

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \ \ s.b. \ \sum_{i}^{N} \alpha_i y_i = 0, \ 0 \leq \alpha_i \leq \gamma$$

- Need to learn M parameters for primal, and N for dual.

Primal vs dual formulation

- Primal problem: $\mathbf{w} \in R^{M-1}, b \in R$

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

- Dual problem:

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \;\; s.b. \;\; \sum_{i}^{N} \alpha_i y_i = 0, \;\; 0 \leq \alpha_i \leq \gamma$$

- Need to learn M parameters for primal, and N for dual.
- If $N << M$, more efficient to solve α than $\mathbf{w}$.

Primal vs dual formulation

- Primal problem: $\mathbf{w} \in R^{M-1}, b \in R$

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

- Dual problem:

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \;\; s.b. \;\; \sum_{i} \alpha_i y_i = 0, \;\; 0 \leq \alpha_i \leq \gamma$$

- Need to learn M parameters for primal, and N for dual.
- If $N << M$, more efficient to solve α than $\mathbf{w}$.
- Dual form only involve $\mathbf{x}_i^T \mathbf{x}_j$.

Primal vs dual formulation

- Primal version of classifier:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

- Dual version of classifier:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

Primal vs dual formulation

- Primal version of classifier:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

- Dual version of classifier:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

- At first sight, dual form requires all training data points $\mathbf{x}_i$, however, many of α_i are zero, only **support vectors** matters.

Non-linearity, kernels

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?

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 using a simple mathematical trick!

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 using a simple mathematical trick!
- One major observation (look at the dual objective function obtained earlier):

$$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \tag{14}$$

$$= \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \underbrace{< \mathbf{x}_i, \mathbf{x}_j >}_{\text{dot product}} \tag{15}$$

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 using a simple mathematical trick!
- One major observation (look at the dual objective function obtained earlier):

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \tag{14}$$

$$= \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \underbrace{< \mathbf{x}_i, \mathbf{x}_j >}_{\text{dot product}} \tag{15}$$

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

Non-linear SVM

Non-linear SVM

- How about the prediction function?

Non-linear SVM

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

$$
\begin{aligned}
f(\mathbf{x}) &= \mathbf{w}^\top \mathbf{x} + b \\
&= \sum_{i=1}^{N_s} \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b \\
&= \sum_{i=1}^{N_s} \alpha_i y_i < \mathbf{x}_i, \mathbf{x} > +b
\end{aligned}
$$

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

Non-linear SVM

- Suppose we map the data to some high dimension Euclidean space using a mapping Φ:

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

usually $h > d$

Non-linear SVM

- Suppose we map the data to some high dimension Euclidean space using a mapping Φ:

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

  usually $h > d$

- The idea is to enlarge the input space to achieve better training class separation.

Non-linear SVM

- Suppose we map the data to some high dimension Euclidean space using a mapping Φ:

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

  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

- We then compute the largest margin hyperplane in the new space $\mathbb{R}^h$.

Mapping

- We then compute the largest margin hyperplane in the new space $\mathbb{R}^h$.
- 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:

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

Mapping

- We then compute the largest margin hyperplane in the new space $\mathbb{R}^h$.
- 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:

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

- All we need in the training is K, and we do not need to explicitly define Φ.

Mapping

- We then compute the largest margin hyperplane in the new space $\mathbb{R}^h$.
- 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:

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

- All we need in the training is K, and we do not need to explicitly define Φ.
- Replace $< \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) >$ with $K(\mathbf{x}_i, \mathbf{x}_j)$ everywhere in the training algorithm.

Mapping

- We then compute the largest margin hyperplane in the new space $\mathbb{R}^h$.
- 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:

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

- All we need in the training is K, and we do not need to explicitly define Φ.
- 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 $\mathbb{R}^h$.

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.

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.
- In testing phase, given the test points $\mathbf{x}$:

$$
\begin{aligned}
f(x) &= \sum_{i=1}^{N_s} \alpha_i y_i < \mathbf{x}_i, \mathbf{x} > + b \\
&= \sum_{i=1}^{N_s} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b
\end{aligned}
$$

where $\mathbf{x}_i$ are the support vectors and N_s is the number of support vectors.

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.
- In testing phase, given the test points $\mathbf{x}$:

$$f(x) = \sum_{i=1}^{N_s} \alpha_i y_i < \mathbf{x}_i, \mathbf{x} > + b$$
$$= \sum_{i=1}^{N_s} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

  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 $\mathbb{R}^2$.

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 $K(\mathbf{x}_i, \mathbf{x}_j) = (< \mathbf{x}_i, \mathbf{x}_j >)^2$.

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

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

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

- One such mapping is:

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

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:

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

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

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

Which functions are allowable as kernels?

- As long as kernel K is positive definite, then there always exists the mapping Φ. (Mercer theorem)

Which functions are allowable as kernels?

- As long as kernel K is positive definite, then there always exists the mapping Φ. (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: $K(\mathbf{x}, \mathbf{y}) = e^{\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}}$

Acknowledgement and Further Reading

Slides are adapted from Dr. Y. Ning's Spring 19 offering of CS-559.

Part of the materials are taken from A. Zisserman's C19 Machine Learning course.

Further Reading:
Chapter 7.1.1 of *Pattern Recognition and Machine Learning* by C. Bishop.