Lecture 16: Modern Classification (I) - Separating Hyperplanes

Hao Helen Zhang

Outline

1. Soft and Hard Classifiers
2. Separating Hyperplane
   - Perceptron Algorithm
   - Binary SVM for Separable Case

Bayes Rule for Binary Problems

Consider the simplest case: two classes are **perfectly linearly** separable.

Linear Classifiers: Construct a linear decision boundary to separate data into different classes as much as possible.

- Under the equal-cost, the Bayes rule is given as:

$$\phi_B(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad P(Y=1|\mathbf{X}=\mathbf{x}) > 0.5 \\ 0 & \text{if} \quad P(Y=1|\mathbf{X}=\mathbf{x}) < P(Y=0|\mathbf{X}=\mathbf{x}). \end{cases}$$

We would like to mimic the Bayes rule.

Soft and Hard Classifiers

- *Soft* classifiers estimate class (posterior) probabilities first and then make a decision:
  (1) first estimate the conditional probability $P(Y = +1|X = x)$;
  (2) then construct the decision rule as $\hat{P}(Y = +1|x) > 0.5$.

  Examples: LDA, Logistics regression, knn methods.

- *Hard* classifiers do not estimate the class probabilities. Instead, they target on the Bayes rule directly:
  - estimate $I[\hat{P}(Y = +1|x) > 0.5]$ directly.

  Examples: SVM

Separating Hyperplanes

- Construct linear decision boundary that explicitly tries to separate the data into different classes as well as possible.
- Good separation is defined in a certain form mathematically.
- Even when the training data can be perfectly separately by hyperplanes, LDA and other linear methods developed under a statistical framework may not achieve perfect separation.

For convenience, we label two classes as $+1$ and -1 from now on.

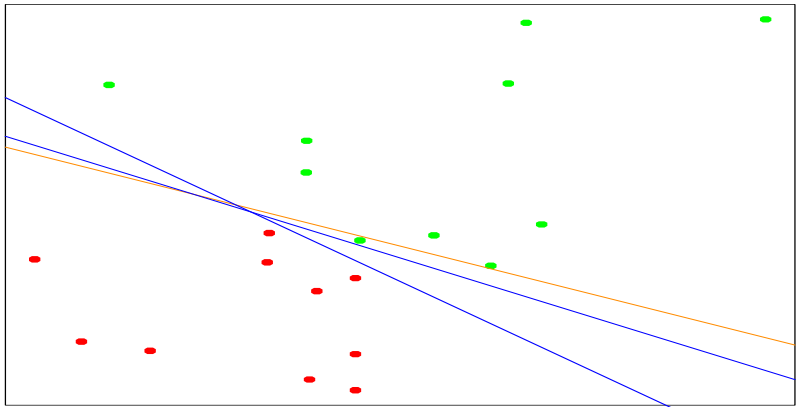

Figure 4.13: *A toy example with two classes separable by a hyperplane. The orange line is the least squares solution, which misclassifies one of the training points. Also shown are two blue separating hyperplanes found by the* perceptron learning algorithm *with different random starts.*

Review of Vector Algebra

Given two vectors **a** and **b** in R^d, the **vector projection** of **a** onto **b** is the orthogonal projection of **a** onto a straight line parallel to **b**,

$$\mathbf{a}_1 = a_1 \mathbf{b}_*,$$

where

$$\mathbf{b}_* = \frac{\mathbf{b}}{\|\mathbf{b}\|}, \quad a_1 = |\mathbf{a}| \cos\theta = \mathbf{a}^T \mathbf{b}_*,$$

$|\mathbf{a}|$ is the length of **a**, θ is the angle between **a** and **b**.

- $\mathbf{b}_*$ is the unit vector in the direction of **b**.
- $\mathbf{a}_1$ is a vector parallel to **b**.
- a_1 is a scalar, called the **scalar projection** of **a** onto **b**.
  - a_1 is a scalar, which can be positive or negative.
  - If $a_1 = 0$, we say **a** is perpendicular to **b**.

The scalar projection is equal to length of the vector projection, with a minus sign if the direction of the projection is opposite to the direction of **b**.

Hyperplane and Its Normal Vector

A hyperplane or *affine set* L is defined by the linear equation

$$L : \{\mathbf{x} : \beta_0 + \boldsymbol{\beta}^T \mathbf{x} = 0\}.$$

- A hyperplane of an d-dimensional space is a flat subset with dimension $d - 1$.
- If $d = 2$, the set L is a line.
- A hyperplane separates the space into two half spaces.

For any two points $\mathbf{x}_1$ and $\mathbf{x}_2$ in L, we have

$$\boldsymbol{\beta}^T(\mathbf{x}_1 - \mathbf{x}_2) = 0.$$

- $\boldsymbol{\beta}$ is perpendicular to L. Define the normal vector of L

$$\boldsymbol{\beta}_* = \boldsymbol{\beta}/\|\boldsymbol{\beta}\|.$$

Signed Distance

For any $\mathbf{x}_0 \in L$, we have

$$\beta^T \mathbf{x}_0 = -\beta_0.$$

For any $\mathbf{x}$ in R^d, its *signed distance* to L is the projection of $\mathbf{x} - \mathbf{x}_0$ onto the direction β_* (the normal vector of L), where $\mathbf{x}_0 \in L$,

$$\beta_*^T(\mathbf{x} - \mathbf{x}_0) = \frac{\beta^T(\mathbf{x} - \mathbf{x}_0)}{\|\beta\|}.$$

- The choice of $\mathbf{x}_0 \in L$ can be arbitrary.
- The signed distance can be positive or negative. It can be used for classification purpose.
  - Points with positive signed distances are assigned to $+1$ class.
  - Points with negative signed distances are assigned to -1 class.

Linear Classfication Boundary

For convenience, we define

$$f(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}^T \mathbf{x}.$$

From the previous slide, we can show that the signed distance of any point $\mathbf{x}$ to L has an expression

$$\frac{f(\mathbf{x})}{\|\boldsymbol{\beta}\|} = \frac{f(\mathbf{x})}{\|f'(\mathbf{x})\|}.$$

- $f(\mathbf{x})$ has the same sign (is proportional to) as the signed distance from $\mathbf{x}$ to L.
- If $f(\mathbf{x}) > 0$, we classify $\mathbf{x}$ to $+1$ Class.
- If $f(\mathbf{x}) < 0$, we classify $\mathbf{x}$ to -1 Class.

Functional Margin

Note that

- If a point is correctly classified, then its signed distance has the same sign as its label. It implies that

$$y_i f(\mathbf{x}_i) > 0.$$

- If a point is misclassified, then its signed distance has the opposite sign as its label. It implies that

$$y_i f(\mathbf{x}_i) < 0.$$

Call the product $y_i f(\mathbf{x}_i)$ the functional margin of the classifier f.

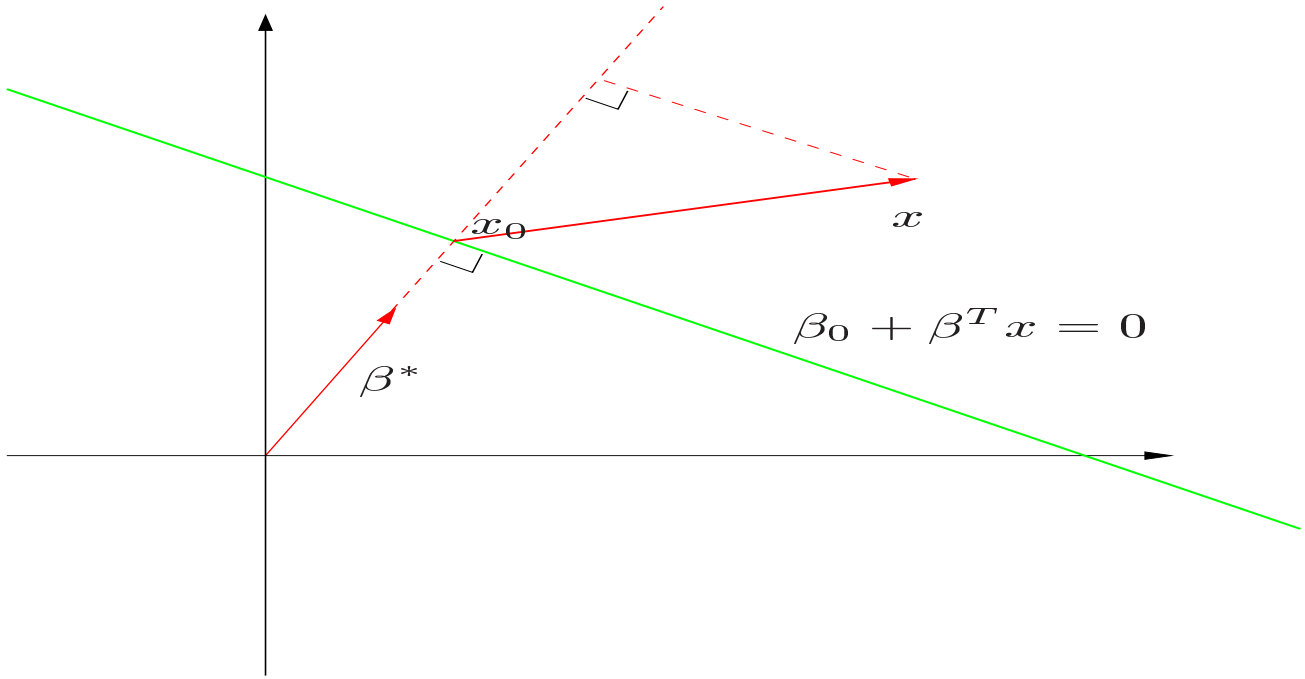

Figure 4.14: *The linear algebra of a hyperplane (affine set).*

Rosenblatt's Perceptron Learning Algorithm

Perceptron: the foundation for support vector machines (Rosenblatt 1958)

Key Idea: Find a separating hyperplane by minimizing the distance of misclassified points to the decision boundary.

- Code the points from two classes as $y_i = 1, -1$.
- If $y_i = 1$ is misclassified, then $\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0 < 0$. If $y_i = -1$ is misclassified, then $\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0 > 0$.
- Since the signed distance from $\mathbf{x}_i$ to L is $\frac{\boldsymbol{\beta}^T \mathbf{x}_i + \beta_0}{\|\boldsymbol{\beta}\|}$, the distance from a misclassified $\mathbf{x}_i$ to the decision boundary is

$$\frac{-y_i(\boldsymbol{\beta}^T \mathbf{x}_i + \beta_0)}{\|\boldsymbol{\beta}\|}.$$

Objective Function for Perceptron Algorithm

Define $\mathbf{M}$ to be the set of misclassified points

$$\mathbf{M} = \{(\mathbf{x}_i, y_i)'s : y_i \text{ and } f(\mathbf{x}_i) \text{ have opposite signs}\}.$$

The goal of the perceptron algorithm is to minimize

$$D(\beta_0, \boldsymbol{\beta}) = -\sum_{i \in \mathbf{M}} y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0)$$

with respect to $\beta_0, \boldsymbol{\beta}$.

Stochastic Gradient Algorithm

Assume $\mathbf{M}$ is fixed. To minimize $D(\beta_0, \boldsymbol{\beta})$, we compute the the gradient of D w.r.t. $(\beta_0, \boldsymbol{\beta})$

$$\nabla D(\boldsymbol{\beta}) = \frac{\partial D}{\partial(\beta_0, \boldsymbol{\beta})} = \begin{pmatrix} -\sum_{i \in \mathbf{M}} y_i \mathbf{x}_i \\ -\sum_{i \in \mathbf{M}} y_i \end{pmatrix}$$

- Stochastic gradient descent is used to minimize the piecewise linear criterion
- Adjustment on $\beta_0, \boldsymbol{\beta}$ is done after each misclassified point is visited.

Update Rule

The stochastic gradient algorithm updates

$$\begin{pmatrix} \beta \\ \beta_0 \end{pmatrix}_{new} = \begin{pmatrix} \beta \\ \beta_0 \end{pmatrix}_{old} + \rho \begin{pmatrix} y_i \mathbf{x}_i \\ y_i \end{pmatrix}$$

- ρ is the learning rate. For example, we may take $\rho = 1$.
- Instead of computing the sum of the gradient contributions of each observation, a step is taken after each observation is visited.
- Note that the true gradient algorithm:
  $\beta_{new} = \beta_{old} - \rho \nabla D(\beta_{old})$

Convergence of Stochastic Gradient Algorithm

Convergence Property: If the classes are linearly separable, the algorithm converges to a separating hyperplane in a finite number of steps.

- Assume the training data are linearly separable. We use $\rho = 1$ for the perceptron algorithm. Let β_{opt} be the coefficients satisfying that $y_i \beta_{opt}^T \mathbf{x} \geq 1$ for all $i = 1, \cdots, n$. Then

$$\|\beta_{new} - \beta_{opt}\|^2 \leq \|\beta_{old} - \beta_{opt}\|^2 - 1.$$

- The algorithm converges in no more than $\|\beta_{start} - \beta_{opt}\|^2$ steps. (Textbook Exercise 4.6)

About Perceptron Learning Algorithm

A number of problems about this algorithm:

- When the data are perfectly separable by hyperplane,
  - there are not unique solutions
  - the solution depends on the starting values.
- The finite number of steps can be "many" steps.
- When the data are not linearly separable
  - the algorithm does not converge.
  - the cycle can be long and therefore hard to detect.

So, we need to *add additional constraints* to the separating hyperplane, and find the optimal hyperplane in some sense.

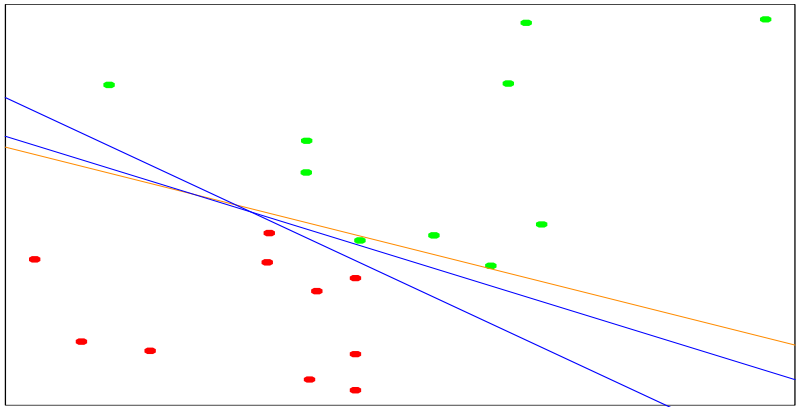

Figure 4.13: *A toy example with two classes separable by a hyperplane. The orange line is the least squares solution, which misclassifies one of the training points. Also shown are two blue separating hyperplanes found by the* perceptron learning algorithm *with different random starts.*

Optimal Separating Hyperplane

Main motivation:

Separate two classes and maximizes the distance to the closest points from either class (Vapnik 1996)

- Provides a unique solution
- Leads to better classification performance on test (future) data

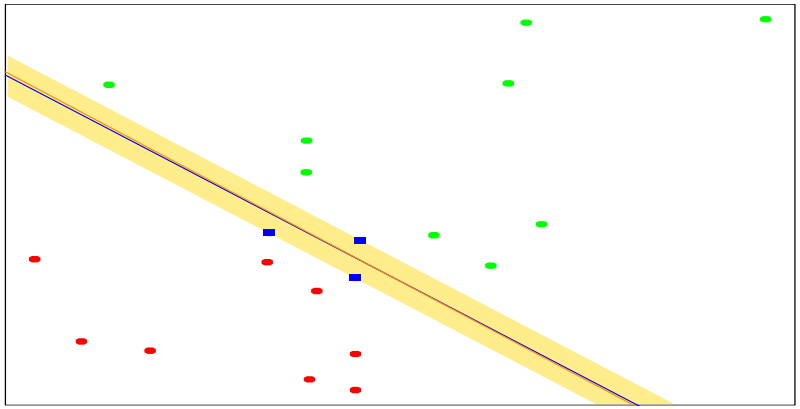

Figure 4.15: *The same data as in Figure 4.13. The shaded region delineates the maximum margin separating the two classes. There are three support points indicated, which lie on the boundary of the margin, and the optimal separating hyperplane (blue line) bisects the slab. Included in the figure is the boundary found using logistic regression (red line), which is very close to the optimal separating hyperplane (see Section 12.3.3).*

Objective Function

Let $f(x) = \beta_0 + \mathbf{x}^T\boldsymbol{\beta}$. Consider the problem:

$$\max_{\beta_0,\boldsymbol{\beta},C} C$$

$$\text{subject to} \quad \frac{y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0)}{\|\boldsymbol{\beta}\|} \geq C, \quad , i = 1, ..., n.$$

All the points are correctly satisfied, and they have at least a signed distance C from the separating hyperplane.

- The maximal C must be positive (for separable data).
- The quantity $yf(\mathbf{x})$ is called the functional "margin".
- A positive "margin" implies a correct classification on $\mathbf{x}$.

The goal is seek the largest such C and associated parameters.

Equivalent Formulation

$$\max_{\beta_0, \boldsymbol{\beta}, C} C$$

$$\text{subject to} \quad y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) \geq C\|\boldsymbol{\beta}\|, \quad , i = 1, ..., n.$$

- If $(\boldsymbol{\beta}, \beta_0)$ is the solution to the above problem, so is any positively scaled $(\alpha\boldsymbol{\beta}, \alpha\beta_0)$, with $\alpha > 0$.
- For identifiability, we can scale $\boldsymbol{\beta}$ by requiring $\|\boldsymbol{\beta}\| = \frac{1}{C}$
- Then the objective function becomes $C = 1/\|\boldsymbol{\beta}\|$.

This lead to

$$\max_{\boldsymbol{\beta}} \frac{1}{\|\boldsymbol{\beta}\|}$$

$$\text{subject to} \quad y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) \geq 1, \quad , i = 1, ..., n.$$

Linear SVM

The above problem is equivalent to This lead to

$$\min_{\beta_0,\boldsymbol{\beta}} \|\boldsymbol{\beta}\|$$
$$\text{subject to} \quad y_i(\mathbf{x}_i^T\boldsymbol{\beta}+\beta_0) \geq 1, \quad , i = 1, ..., n.$$

For computational convenience, we further replace $\|\boldsymbol{\beta}\|$ by $\frac{1}{2}\|\boldsymbol{\beta}\|^2$ (a monotonic transformation), which leads to

$$\min_{\beta_0,\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{\beta}\|^2$$
$$\text{subject to} \quad y_i(\mathbf{x}_i^T\boldsymbol{\beta}+\beta_0) \geq 1, \quad \text{for} \quad i = 1, ..., n$$

This is the linear SVM for perfectly separable cases.

How to Interpret $\frac{1}{\|\beta\|}$

Take one point $\mathbf{x}_1$ from $f(\mathbf{x}_1) = +1$, one point $\mathbf{x}_2$ from $f(\mathbf{x}_2) = -1$

$$\begin{aligned} & \beta_0 + \beta^T \mathbf{x}_1 = +1 \\ & \beta_0 + \beta^T \mathbf{x}_2 = -1 \\ \Longrightarrow \quad & \beta^T (\mathbf{x}_1 - \mathbf{x}_2) = 2 \\ \Longrightarrow \quad & \frac{\beta^T}{\|\beta\|} (\mathbf{x}_1 - \mathbf{x}_2) = \frac{2}{\|\beta\|} \end{aligned}$$

- We projected the vector $\mathbf{x}_1 - \mathbf{x}_2$ onto the normal vector of our hyperplane.
- $2/\|\beta\|$ is the width of margin between two hyperplanes $f(\mathbf{x}) = +1$ and $f(\mathbf{x}) = -1$.

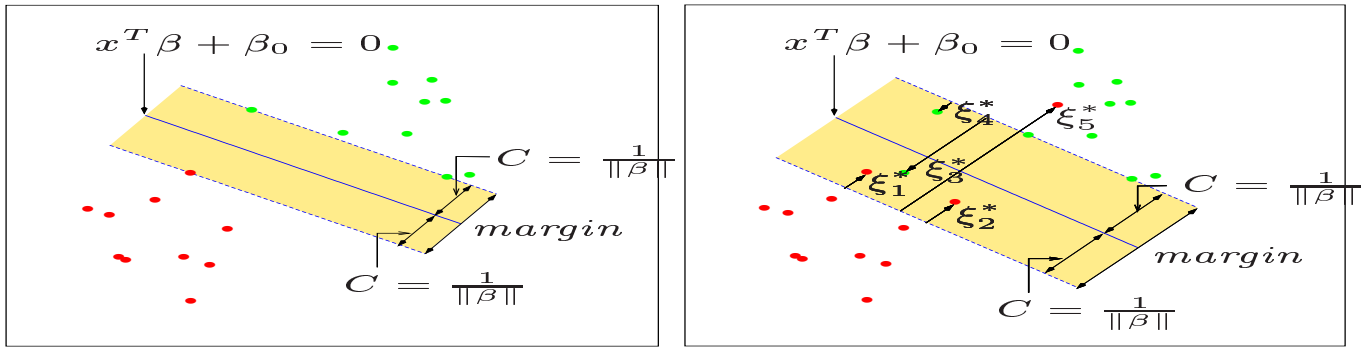

Figure 12.1: *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width* $2C = 2/\|\beta\|$. *The right panel shows the nonseparable (overlap) case. The points labeled* ξ_j^* *are on the wrong side of their margin by an amount* $\xi_j^* = C\xi_j$; *points on the correct side have* $\xi_j^* = 0$. *The margin is maximized subject to a total budget* $\sum \xi_i \leq$ *constant. Hence* $\sum \xi_j^*$ *is the total distance of points on the wrong side of their margin.*

Maximal Margin Classifier

Geometrical **Margin**:
defined as $d_+ + d_-$, where $d_+(d_-)$ is the shortest distance from the separating hyperplane to the closest positive (negative) training data point.

- Margin is bounded below by $\frac{2}{\|\beta\|}$
- Use squared margin for computation convenience
- A large margin on the training data will lead to good separation on the test data.

Defined validly only for separable cases

Solve SVM by Quadratic Programming

Need to solve a convex optimization problem:

- quadratic objective function
- linear inequality constraints

Lagrange (primal) function: introduce Lagrange multipliers $\alpha_i \geq 0$

$$L_P(\boldsymbol{\beta}, \beta_0, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) - 1]$$

L has to be minimized with respect to the *primal variables* $(\boldsymbol{\beta}, \beta_0)$ and maximized with respect to the *dual variables* α_i. (In other words, we need to find the saddle points for L_P).

Solve the Wolfe Dual Problem

At the saddle points, we have

$$\frac{\partial}{\partial \beta_0} L(\boldsymbol{\beta}, \beta_0, \boldsymbol{\alpha}) = 0, \quad \frac{\partial}{\partial \boldsymbol{\beta}} L(\boldsymbol{\beta}, \beta_0, \boldsymbol{\alpha}) = 0$$

i.e.

$$\begin{aligned} 0 &= \sum_i \alpha_i y_i, \\ \boldsymbol{\beta} &= \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i. \end{aligned}$$

By substituting both into L to get the *dual problem*:

$$L_D(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to

$$\alpha_i \geq 0, \quad i = 1, ..., n; \quad \sum_{i=1}^n \alpha_i y_i = 0.$$

SVM Solutions

- First, solve the dual problem for α.

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - \sum_{i=1}^{n} \alpha_i,$$

  subject to $\alpha_i \geq 0, i = 1, ..., n$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$. Denote the solution as $\hat{\alpha}_i, i = 1, \ldots, n$.
- The SVM slope is obtained by $\hat{\boldsymbol{\beta}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x}_i$.
- The SVM intercept $\hat{\beta}_0$ is solved from KKT condition

$$\hat{\alpha}_i [y_i (\mathbf{x}_i^T \hat{\boldsymbol{\beta}} + \beta_0) - 1] = 0,$$

  with any of the points with $\hat{\alpha} > 0$ (support vectors)
- The SVM boundary is given by $\hat{f}(\mathbf{x}) = \hat{\beta}_0 + \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x}_i^T \mathbf{x}$.

Karush-Kuhn-Tucker (KKT) Optimality Condition

The optimal solution should satisfy the following:

$$0 = \sum_i \alpha_i y_i \quad (1)$$

$$\boldsymbol{\beta} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \quad (2)$$

$$\alpha_i \geq 0, \quad \text{for} \quad i = 1, ..., n \quad (3)$$

$$0 = \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) - 1], \quad (4)$$

$$1 \leq y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0), \quad \text{for} \quad i = 1, ...n. \quad (5)$$

Interpretation of Solution

Optimality condition (4) is called the "complementary slackness"

$$0 = \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) - 1].$$

Since $\alpha_i \geq 0$ and $y_i f(\mathbf{x}_i) \geq 1$ for all i, this implies that the SVM solution must satisfy

$$\alpha_i [y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) - 1] = 0, \quad \forall i = 1, \ldots, n.$$

In other words,

- If $\alpha_i > 0$, then $y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) = 1$, so $\mathbf{x}_i$ must lie on the margin.
- If $y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) > 1$ (not on the margin), then $\alpha_i = 0$.

No training points fall in the margin.

Support Vectors

Support Vectors (SVs): all the points with $\alpha_i > 0$. Define the index set

$$SV = \{i | \hat{\alpha}_i > 0, \quad i = 1..., n\}$$

- The solution and decision boundary can be expressed only in terms of SVs

$$\hat{\boldsymbol{\beta}} = \sum_{i \in SV} \hat{\alpha}_i y_i \mathbf{x}_i$$

$$\hat{f}(\mathbf{x}) = \hat{\beta}_0 + \sum_{i \in SV} \hat{\alpha}_i y_i \mathbf{x}_i^T \mathbf{x}.$$

- The identification of the SV points requires the use of all the data points.

Various Classifiers

If the classes are really Gaussian, then

- the LDA is optimal.
- the separating hyperplane pays a price for focusing on the (noisier) data at the boundaries

Optimal separating hyperplane has less assumptions, thus more robust to model misspecification.

- In the separable case, logistic regression solution is similar to the separating hyperplane solution.
- For perfectly separable case, the log-likelihood can be driven to zero.