

Linear methods for classification

In this lecture, we will take a close look at some classification schemes that are inspired by linear models:

- logistic regression
- separating hyperplanes

Recap: least-squares linear regression

Given training data $(X_1, Y_1), \dots, (X_n, Y_n)$ with $X_i \in \mathbb{R}^p$ and $Y_i \in \mathbb{R}$, find $\hat{\beta}^* \in \mathbb{R}^p$ and $\hat{c}^* \in \mathbb{R}$ to minimize

$$\hat{\mathcal{E}}(\beta, c) = \frac{1}{n} \sum_{i=1}^n (Y_i - \beta^T X_i - c)^2$$

(note the presence of the **intercept** c !!!)

Recap: least-squares linear regression

Given training data $(X_1, Y_1), \dots, (X_n, Y_n)$ with $X_i \in \mathbb{R}^p$ and $Y_i \in \mathbb{R}$, find $\hat{\beta}^* \in \mathbb{R}^p$ and $\hat{c}^* \in \mathbb{R}$ to minimize

$$\hat{\mathcal{E}}(\beta, c) = \frac{1}{n} \sum_{i=1}^n (Y_i - \beta^T X_i - c)^2$$

(note the presence of the **intercept** c !!!)

To incorporate the intercept, simply add an extra coordinate to each X_i :

$$X_i = (X_{i1}, X_{i2}, \dots, X_{ip}) \longrightarrow X'_i = (X_{i1}, X_{i2}, \dots, X_{ip}, 1)$$

and define the vector $\beta' = (\beta_1, \dots, \beta_p, c)$. Then for every training point X_i we will have

$$\beta'^T X'_i = \sum_{j=1}^p \beta_j X_{ij} + c = \beta^T X_i + c$$

Recap: least-squares linear regression

Given training data $(X_1, Y_1), \dots, (X_n, Y_n)$ with $X_i \in \mathbb{R}^p$ and $Y_i \in \mathbb{R}$, find $\hat{\beta}^* \in \mathbb{R}^p$ and $\hat{c}^* \in \mathbb{R}$ to minimize

$$\hat{\mathcal{E}}(\beta, c) = \frac{1}{n} \sum_{i=1}^n (Y_i - \beta^T X_i - c)^2$$

(note the presence of the **intercept** c !!!)

To incorporate the intercept, simply add an extra coordinate to each X_i :

$$X_i = (X_{i1}, X_{i2}, \dots, X_{ip}) \longrightarrow X'_i = (X_{i1}, X_{i2}, \dots, X_{ip}, 1)$$

and define the vector $\beta' = (\beta_1, \dots, \beta_p, c)$. Then for every training point X_i we will have

$$\beta'^T X'_i = \sum_{j=1}^p \beta_j X_{ij} + c = \beta^T X_i + c$$

From now on we will always assume that the last coordinate of every training point is equal to 1.

Recap: least-squares linear regression and classification

Minimize $\hat{\mathcal{E}}(\beta) = \frac{1}{n} \sum_{i=1}^n (Y_i - \beta^T X_i)^2$ over all β

The least-squares solution:

Assume that $\mathbf{X}^T \mathbf{X}$ is invertible. Then

$$\hat{\beta}^* = (\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{Y}),$$

where:

- \mathbf{X} is the design matrix, the i th row is the i th training feature vector X_i
- \mathbf{Y} is the response vector, the i th coordinate is Y_i

Recap: least-squares linear regression and classification

Minimize
$$\widehat{\mathcal{E}}(\beta) = \frac{1}{n} \sum_{i=1}^n (Y_i - \beta^T X_i)^2 \quad \text{over all } \beta$$

The least-squares solution:

Assume that $\mathbf{X}^T \mathbf{X}$ is invertible. Then

$$\widehat{\beta}^* = (\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{Y}),$$

where:

- \mathbf{X} is the design matrix, the i th row is the i th training feature vector X_i
- \mathbf{Y} is the response vector, the i th coordinate is Y_i

Binary classification:

Suppose each $Y_i \in \{0, 1\}$. Then to classify a new test point x use the rule

$$\widehat{f}(x) = \begin{cases} 1, & \text{if } \widehat{\beta}_*^T x \geq 1/2 \\ 0, & \text{if } \widehat{\beta}_*^T x < 1/2 \end{cases}$$

Regression-based classifier: pros and cons

Pros:

- relatively easy to compute
- has certain intuitive appeal

Cons:

- $\hat{\beta}_*^T x$ may be negative or it may be greater than 1: does not make sense since the true regression function $\eta(x)$ is a **probability** that $Y = 1$ given $X = x$
- too simple, thus may result in severe underfitting

Regression-based classifier: pros and cons

Pros:

- relatively easy to compute
- has certain intuitive appeal

Cons:

- $\hat{\beta}_*^T x$ may be negative or it may be greater than 1: does not make sense since the true regression function $\eta(x)$ is a **probability** that $Y = 1$ given $X = x$
- too simple, thus may result in severe underfitting

Alternative approaches:

- **logistic regression** – combines the simplicity of linear models with proper probabilities
- **optimal separating hyperplane** – works well when the data are known to be linearly separable

Binary logistic regression

We would like to model the probabilities $\mathbb{P}[Y = 1|X = x]$ and $\mathbb{P}[Y = 0|X = x]$ by linear functions of x , yet still ensure that our model gives a valid probability distribution.

Binary logistic regression

We would like to model the probabilities $\mathbb{P}[Y = 1|X = x]$ and $\mathbb{P}[Y = 0|X = x]$ by linear functions of x , yet still ensure that our model gives a valid probability distribution.

Main insight: suppose we have two real numbers, a and b , which can be positive or negative. But now observe that the numbers

$$p = \frac{e^a}{e^b + e^a}, \quad q = \frac{e^b}{e^b + e^a}$$

are nonnegative ($p \geq 0, q \geq 0$) and sum to one ($p + q = 1$).

Binary logistic regression

We would like to model the probabilities $\mathbb{P}[Y = 1|X = x]$ and $\mathbb{P}[Y = 0|X = x]$ by linear functions of x , yet still ensure that our model gives a valid probability distribution.

Main insight: suppose we have two real numbers, a and b , which can be positive or negative. But now observe that the numbers

$$p = \frac{e^a}{e^b + e^a}, \quad q = \frac{e^b}{e^b + e^a}$$

are nonnegative ($p \geq 0, q \geq 0$) and sum to one ($p + q = 1$).

Moreover, we can always set $b = 0$ because

$$p = \frac{e^a}{e^a + e^b} = \frac{e^b e^{a-b}}{e^b(1 + e^{a-b})} = \frac{e^c}{1 + e^c}$$
$$q = \frac{e^b}{e^b + e^a} = \frac{e^b}{e^b(1 + e^{a-b})} = \frac{1}{1 + e^c},$$

where we have set $c = a - b$.

Binary logistic regression

We would like to model the probabilities $\mathbb{P}[Y = 1|X = x]$ and $\mathbb{P}[Y = 0|X = x]$ by linear functions of x , yet still ensure that our model gives a valid probability distribution.

Main insight: suppose we have two real numbers, a and b , which can be positive or negative. But now observe that the numbers

$$p = \frac{e^a}{e^b + e^a}, \quad q = \frac{e^b}{e^b + e^a}$$

are nonnegative ($p \geq 0, q \geq 0$) and sum to one ($p + q = 1$).

Moreover, we can always set $b = 0$ because

$$p = \frac{e^a}{e^a + e^b} = \frac{e^b e^{a-b}}{e^b(1 + e^{a-b})} = \frac{e^c}{1 + e^c}$$
$$q = \frac{e^b}{e^b + e^a} = \frac{e^b}{e^b(1 + e^{a-b})} = \frac{1}{1 + e^c},$$

where we have set $c = a - b$. Finally, note that

$$\log \frac{p}{q} = c$$

Binary logistic regression

Logistic regression is based on this insight. We assume as our model that the regression function $\eta(x) = \mathbb{P}[Y = 1|X = x]$ satisfies

$$\log \frac{\mathbb{P}[Y = 1|X = x]}{\mathbb{P}[Y = 0|X = x]} = c + \beta^T x$$

for some $c \in \mathbb{R}$ and $\beta \in \mathbb{R}^p$. In other words, we have:

Logistic regression

Under the logistic regression model, we assume that the regression function $\eta(x) = \mathbb{P}[Y = 1|X = x]$ takes the form

$$\eta(x) = \frac{\exp(c + \beta^T x)}{1 + \exp(c + \beta^T x)}$$

for some $c \in \mathbb{R}$ and some $\beta \in \mathbb{R}^p$. Note that $\eta(x)$ is always between 0 and 1.

Notation: we will use a single symbol θ for the pair β, c and write $p_1(x; \theta)$ for $\eta(x)$ and $p_0(x; \theta)$ for $1 - \eta(x)$.

Fitting logistic regression models

Given training data $(X_1, Y_1), \dots, (X_n, Y_n)$ with all $Y_i \in \{0, 1\}$, we can fit a logistic regression model to it as follows.

Given the parameter $\theta = (\beta, c)$, we can view

$$p_y(x; \theta) = \frac{\exp[y(c + \beta^T x)]}{1 + \exp(c + \beta^T x)}, \quad y \in \{0, 1\}$$

as the **likelihood** that $Y = y$ given the observation $X = x$ and the parameter θ .

Fitting logistic regression models

Given training data $(X_1, Y_1), \dots, (X_n, Y_n)$ with all $Y_i \in \{0, 1\}$, we can fit a logistic regression model to it as follows.

Given the parameter $\theta = (\beta, c)$, we can view

$$p_y(x; \theta) = \frac{\exp[y(c + \beta^T x)]}{1 + \exp(c + \beta^T x)}, \quad y \in \{0, 1\}$$

as the **likelihood** that $Y = y$ given the observation $X = x$ and the parameter θ .

Again, assuming that each training point X_i has 1 as its last coordinate, we can write

$$p_y(x; \theta) = \frac{\exp(y\theta^T x)}{1 + \exp(\theta^T x)}, \quad y \in \{0, 1\}$$

Fitting logistic regression models

Given training data $(X_1, Y_1), \dots, (X_n, Y_n)$ with all $Y_i \in \{0, 1\}$, we can fit a logistic regression model to it as follows.

Given the parameter $\theta = (\beta, c)$, we can view

$$p_y(x; \theta) = \frac{\exp[y(c + \beta^T x)]}{1 + \exp(c + \beta^T x)}, \quad y \in \{0, 1\}$$

as the **likelihood** that $Y = y$ given the observation $X = x$ and the parameter θ .

Again, assuming that each training point X_i has 1 as its last coordinate, we can write

$$p_y(x; \theta) = \frac{\exp(y\theta^T x)}{1 + \exp(\theta^T x)}, \quad y \in \{0, 1\}$$

We can also write down the **log-likelihood**

$$\log p_y(x; \theta) = \log \left[\frac{\exp(y\theta^T x)}{1 + \exp(\theta^T x)} \right] = y\theta^T x - \log \left[1 + \exp(\theta^T x) \right]$$

Fitting logistic regression models

Suppose we have n points (X_i, Y_i) , $i = 1, \dots, n$, all drawn independently under the logistic regression model with parameter θ .

for $i = 1, \dots, n$ **do**:

- draw X_i from \mathbb{P}_X independently of everything else
- draw $Y_i = y$ with probability equal to $p_y(X_i; \theta)$

end for

Fitting logistic regression models

Suppose we have n points (X_i, Y_i) , $i = 1, \dots, n$, all drawn independently under the logistic regression model with parameter θ .

for $i = 1, \dots, n$ **do**:

- draw X_i from \mathbb{P}_X independently of everything else
- draw $Y_i = y$ with probability equal to $p_y(X_i; \theta)$

end for

The probability of a particular sequence $(x_1, y_1), \dots, (x_n, y_n)$ is

$$\begin{aligned} & \mathbb{P}_\theta[(X_i = x_i, Y_i = y_i) : 1 \leq i \leq n] \\ &= \prod_{i=1}^n \mathbb{P}_\theta[(X_i = x_i, Y_i = y_i)] \quad (\text{because of independence}) \\ &= \prod_{i=1}^n \mathbb{P}_X[X_i = x_i] \cdot p_{y_i}(x_i; \theta) \quad (\text{because prob. of } X \text{ is indep. of } \theta) \end{aligned}$$

Fitting logistic regression models

Suppose we have n points (X_i, Y_i) , $i = 1, \dots, n$, all drawn independently under the logistic regression model with parameter θ .

for $i = 1, \dots, n$ **do**:

- draw X_i from \mathbb{P}_X independently of everything else
- draw $Y_i = y$ with probability equal to $p_y(X_i; \theta)$

end for

The probability of a particular sequence $(x_1, y_1), \dots, (x_n, y_n)$ is

$$\begin{aligned}\mathbb{P}_\theta[(X_i = x_i, Y_i = y_i) : 1 \leq i \leq n] \\ &= \prod_{i=1}^n \mathbb{P}_\theta[(X_i = x_i, Y_i = y_i)] \quad (\text{because of independence}) \\ &= \prod_{i=1}^n \mathbb{P}_X[X_i = x_i] \cdot p_{y_i}(x_i; \theta) \quad (\text{because prob. of } X \text{ is indep. of } \theta)\end{aligned}$$

It is convenient to look at the **log-likelihood**:

$$\log \mathbb{P}_\theta[(X_i = x_i, Y_i = y_i) : 1 \leq i \leq n] = \sum_{i=1}^n \{ \log \mathbb{P}_X[X_i = x_i] + \log p_{y_i}(x_i; \theta) \}$$

Fitting logistic regression models

Log-likelihood of the training data (assuming parameter θ):

$$\log \mathbb{P}_\theta[(X_i = x_i, Y_i = y_i) : 1 \leq i \leq n] = \sum_{i=1}^n \{\log \mathbb{P}_X[X_i = x_i] + \log p_{y_i}(x_i; \theta)\}$$

We will find $\hat{\theta}^*$ under which the observed training data are **most probable** — need to **maximize** the log-likelihood.

Fitting logistic regression models

Log-likelihood of the training data (assuming parameter θ):

$$\log \mathbb{P}_\theta[(X_i = x_i, Y_i = y_i) : 1 \leq i \leq n] = \sum_{i=1}^n \{\log \mathbb{P}_X[X_i = x_i] + \log p_{y_i}(x_i; \theta)\}$$

We will find $\hat{\theta}^*$ under which the observed training data are **most probable** — need to **maximize** the log-likelihood.

Observe that we can write

$$\log \mathbb{P}_\theta[(X_i = x_i, Y_i = y_i) : 1 \leq i \leq n] = \underbrace{\sum_{i=1}^n \log \mathbb{P}_X[X_i = x_i]}_{\text{indep. of } \theta} + \underbrace{\sum_{i=1}^n \log p_{y_i}(x_i; \theta)}_{=\ell(\theta)}$$

Fitting logistic regression models

Log-likelihood of the training data (assuming parameter θ):

$$\log \mathbb{P}_\theta[(X_i = x_i, Y_i = y_i) : 1 \leq i \leq n] = \sum_{i=1}^n \{\log \mathbb{P}_X[X_i = x_i] + \log p_{y_i}(x_i; \theta)\}$$

We will find $\hat{\theta}^*$ under which the observed training data are **most probable** — need to **maximize** the log-likelihood.

Observe that we can write

$$\log \mathbb{P}_\theta[(X_i = x_i, Y_i = y_i) : 1 \leq i \leq n] = \underbrace{\sum_{i=1}^n \log \mathbb{P}_X[X_i = x_i]}_{\text{indep. of } \theta} + \underbrace{\sum_{i=1}^n \log p_{y_i}(x_i; \theta)}_{=\ell(\theta)}$$

So we will fit a logistic regression model to the data $(x_1, y_1), \dots, (x_n, y_n)$ by maximizing over θ the log likelihood

$$\begin{aligned} \ell(\theta) &= \sum_{i=1}^n \log p_{y_i}(x_i; \theta) \\ &= \sum_{i=1}^n \left\{ y_i \theta^T x_i + \log[1 - \exp(\theta^T x_i)] \right\} \end{aligned}$$

Fitting logistic regression models

Maximize $\ell(\theta) = \sum_{i=1}^n \left\{ y_i \theta^T x_i - \log[1 + \exp(\theta^T x_i)] \right\}$ over θ

Fitting logistic regression models

Maximize $\ell(\theta) = \sum_{i=1}^n \left\{ y_i \theta^T x_i - \log[1 + \exp(\theta^T x_i)] \right\}$ over θ

Differentiate w.r.t. θ and set the derivatives to zero:

$$\nabla \ell(\theta) = \sum_{i=1}^n x_i (y_i - p_0(x_i; \theta)) = 0$$

This is a system of $p + 1$ equations (why?) that are **nonlinear** in θ .

Fitting logistic regression models

Maximize $\ell(\theta) = \sum_{i=1}^n \left\{ y_i \theta^T x_i - \log[1 + \exp(\theta^T x_i)] \right\}$ over θ

Differentiate w.r.t. θ and set the derivatives to zero:

$$\nabla \ell(\theta) = \sum_{i=1}^n x_i (y_i - p_0(x_i; \theta)) = 0$$

This is a system of $p + 1$ equations (why?) that are **nonlinear** in θ .

It can be solved iteratively using the [Newton–Raphson method](#):

```
let  $t = 0$ 
choose  $\hat{\theta}_*(0)$  arbitrarily
do
  •  $\hat{\theta}_*(t + 1) = \hat{\theta}_*(t) - [\nabla^2 \ell(\hat{\theta}_*(t))]^{-1} \nabla \ell(\hat{\theta}_*(t))$ 
  •  $t \leftarrow t + 1$ 
until convergence
```

Matrix form of the Newton–Raphson updates

$$\nabla \ell(\theta) = \sum_{i=1}^n x_i (y_i - p_0(x_i; \theta)), \quad \nabla^2 \ell(\theta) = - \sum_{i=1}^n x_i x_i^T p_1(x_i; \theta) [1 - p_1(x_i; \theta)]$$

Matrix form of the Newton–Raphson updates

$$\nabla \ell(\theta) = \sum_{i=1}^n x_i (y_i - p_0(x_i; \theta)), \quad \nabla^2 \ell(\theta) = - \sum_{i=1}^n x_i x_i^T p_1(x_i; \theta) [1 - p_1(x_i; \theta)]$$

Define at each iteration t

$$\mathbf{p} = \left(p_1(x_1; \hat{\theta}_*(t)), \dots, p_1(x_n; \hat{\theta}_*(t)) \right)^T$$

$$\mathbf{W} = \text{diag} \left(p_1(x_1; \hat{\theta}_*(t)) [1 - p_1(x_1; \hat{\theta}_*(t))], \dots, p_1(x_n; \hat{\theta}_*(t)) [1 - p_1(x_n; \hat{\theta}_*(t))] \right)$$

Matrix form of the Newton–Raphson updates

$$\nabla \ell(\theta) = \sum_{i=1}^n x_i (y_i - p_0(x_i; \theta)), \quad \nabla^2 \ell(\theta) = - \sum_{i=1}^n x_i x_i^T p_1(x_i; \theta) [1 - p_1(x_i; \theta)]$$

Define at each iteration t

$$\mathbf{p} = \left(p_1(x_1; \hat{\theta}_*(t)), \dots, p_1(x_n; \hat{\theta}_*(t)) \right)^T$$

$$\mathbf{W} = \text{diag} \left(p_1(x_1; \hat{\theta}_*(t)) [1 - p_1(x_1; \hat{\theta}_*(t))], \dots, p_1(x_n; \hat{\theta}_*(t)) [1 - p_1(x_n; \hat{\theta}_*(t))] \right)$$

Then $\nabla \ell(\theta) = \mathbf{X}^T (\mathbf{Y} - \mathbf{p})$ and $\nabla^2 \ell(\theta) = -\mathbf{X}^T \mathbf{W} \mathbf{X}$, and we can rewrite the Newton–Raphson update as

$$\begin{aligned} \hat{\theta}_*(t+1) &= \hat{\theta}_*(t) + (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{Y} - \mathbf{p}) \\ &= (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{X} \hat{\theta}_*(t) + (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{W}) \mathbf{W}^{-1} (\mathbf{Y} - \mathbf{p}) \\ &= (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} (\mathbf{X} \hat{\theta}_*(t) + \mathbf{W}^{-1} (\mathbf{Y} - \mathbf{p})) \\ &= (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{z}, \end{aligned}$$

where $\mathbf{z} = \mathbf{X} \hat{\theta}_*(t) + \mathbf{W}^{-1} (\mathbf{Y} - \mathbf{p})$ is called the **adjusted response**

Logistic regression: issues

- Combines the simplicity of linear models with proper probability assignments.
- Each Newton–Raphson iteration requires inverting an $n \times n$ matrix, so complexity can be high when n is large.

Separating hyperplanes

Separating hyperplane methods try to find the best linear decision boundary that separates the two classes. Here is an example of 20 data points that can be separated by a hyperplane:

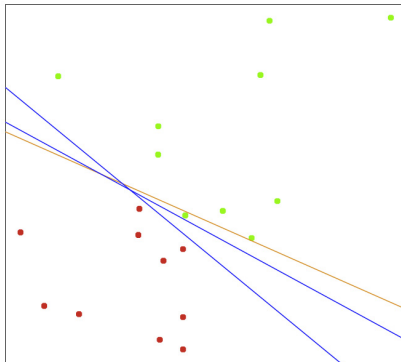


Figure taken from Hastie, Tibshirani and Friedman, 2nd ed.

Geometry of hyperplanes

A **hyperplane** L is defined by an equation

$$f(x) = \beta_0 + \beta^T x = 0$$

- For any two points $x_1, x_2 \in L$,

$$\beta^T (x_1 - x_2) = 0$$

Therefore, the unit vector $\beta^* = \beta / \|\beta\|$ is **normal** to L

- For any point $x \in L$, $\beta^T x = -\beta_0$
- The **signed distance** of any point x to L is given by

$$\begin{aligned} \beta^{*T} (x - x_0) &= \frac{1}{\|\beta\|} (\beta^T x + \beta_0) \\ &= \frac{1}{\|f'(x)\|} f(x) \end{aligned}$$

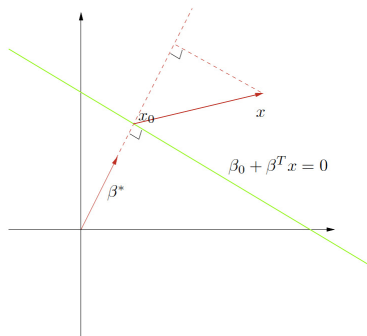


Figure taken from Hastie, Tibshirani and Friedman, 2nd ed.

The Perceptron Learning Algorithm

The earliest algorithm for fitting a separating hyperplane to a training set $(x_1, y_1), \dots, (x_n, y_n)$ with $y_i \in \{-1, +1\}$ is the [Perceptron Learning Algorithm](#) of F. Rosenblatt (1957).

The Perceptron Learning Algorithm

The earliest algorithm for fitting a separating hyperplane to a training set $(x_1, y_1), \dots, (x_n, y_n)$ with $y_i \in \{-1, +1\}$ is the **Perceptron Learning Algorithm** of F. Rosenblatt (1957).

The basic idea: for a candidate separating hyperplane $f(x) = \beta^T x + \beta_0$, let's classify a point x as

$$\hat{f}(x) = \begin{cases} +1, & \text{if } f(x) = \beta^T x + \beta_0 > 0 \\ -1, & \text{if } f(x) = \beta^T x + \beta_0 < 0 \end{cases}$$

Apply this classifier to each training point (x_i, y_i) . If

$$y_i f(x_i) = y_i [\beta^T x_i + \beta_0] < 0,$$

then the i th point is **misclassified**.

The Perceptron Learning Algorithm

The earliest algorithm for fitting a separating hyperplane to a training set $(x_1, y_1), \dots, (x_n, y_n)$ with $y_i \in \{-1, +1\}$ is the **Perceptron Learning Algorithm** of F. Rosenblatt (1957).

The basic idea: for a candidate separating hyperplane $f(x) = \beta^T x + \beta_0$, let's classify a point x as

$$\hat{f}(x) = \begin{cases} +1, & \text{if } f(x) = \beta^T x + \beta_0 > 0 \\ -1, & \text{if } f(x) = \beta^T x + \beta_0 < 0 \end{cases}$$

Apply this classifier to each training point (x_i, y_i) . If

$$y_i f(x_i) = y_i [\beta^T x_i + \beta_0] < 0,$$

then the i th point is **misclassified**. Hence we have:

The Perceptron Learning Algorithm (Rosenblatt)

Hence, find the pair β_0, β to minimize

$$N(\beta_0, \beta) = - \sum_{i \text{ misclassified}} y_i [\beta^T x_i + \beta_0]$$

Implementing Perceptron Learning

To minimize $N(\beta_0, \beta)$, compute the gradient (assuming the set M of misclassified examples is fixed):

$$\frac{\partial N(\beta, \beta_0)}{\partial \beta} = - \sum_{i \in M} y_i x_i$$

$$\frac{\partial N(\beta, \beta_0)}{\partial \beta_0} = - \sum_{i \in M} y_i$$

Implementing Perceptron Learning

To minimize $N(\beta_0, \beta)$, compute the gradient (assuming the set M of misclassified examples is fixed):

$$\frac{\partial N(\beta, \beta_0)}{\partial \beta} = - \sum_{i \in M} y_i x_i$$

$$\frac{\partial N(\beta, \beta_0)}{\partial \beta_0} = - \sum_{i \in M} y_i$$

Do an update of (β, β_0) by visiting each misclassified example: if the current example being visited is i , then do the update

$$\beta \leftarrow \beta + \rho y_i x_i, \quad \beta_0 \leftarrow \beta_0 + \rho y_i,$$

where ρ is the **learning rate**; can be taken to be 1

Perceptrons: Pros and Cons

- If the classes are linearly separable, the algorithm converges to a separating hyperplane in a finite number of steps.
- A number of problems with the algorithm:
 - When the data are separable, there are many solutions, and which one is found depends on the starting values.
 - The number of steps can be very large. The smaller the gap, the longer it takes 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.

Optimal separating hyperplane

Suppose the two classes are, in fact, linearly separable.

The [optimal separating hyperplane](#) algorithm (its roots go back to [Vapnik and Lerner, 1963](#)) separates the two classes by maximizing the [margin](#), i.e., distance from the hyperplane to the closest training point in each class.

It is formulated as follows:

Optimal Separating Hyperplane

$$\begin{aligned} & \max_{(\beta, \beta_0); \|\beta\|=1} C \\ & \text{subject to } y_i(\beta^T x_i + \beta_0) \geq C, \quad i = 1, \dots, n \end{aligned}$$

Rationale: Since $\|\beta\| = 1$, $\beta^T x_i + \beta_0$ is the [signed distance](#) between x_i and the hyperplane $f(x) = \beta^T x + \beta_0$. The constraints require that each training point is at least a signed distance C from the decision boundary. The goal, then, is to find the hyperplane with parameters β and β_0 that would have the largest such C .

A convex formulation

Equivalent problem — get rid of the $\|\beta\| = 1$ constraint:

$$\max_{(\beta, \beta_0)} C$$

$$\text{subject to } y_i(\beta^T x_i + \beta_0) \geq C\|\beta\|, \quad i = 1, \dots, n$$

A convex formulation

Equivalent problem — get rid of the $\|\beta\| = 1$ constraint:

$$\max_{(\beta, \beta_0)} C$$

$$\text{subject to } y_i(\beta^T x_i + \beta_0) \geq C\|\beta\|, \quad i = 1, \dots, n$$

If β and β_0 satisfy the constraints, we can rescale them by any positive constant and still satisfy the constraint, so we can choose $C = 1/\|\beta\|$ and instead solve

$$\min_{(\beta, \beta_0)} \frac{1}{2} \|\beta\|^2$$

$$\text{subject to } y_i(\beta^T x_i + \beta_0) \geq 1, \quad i = 1, \dots, n$$

A convex formulation

Equivalent problem — get rid of the $\|\beta\| = 1$ constraint:

$$\max_{(\beta, \beta_0)} C$$

$$\text{subject to } y_i(\beta^T x_i + \beta_0) \geq C\|\beta\|, \quad i = 1, \dots, n$$

If β and β_0 satisfy the constraints, we can rescale them by any positive constant and still satisfy the constraint, so we can choose $C = 1/\|\beta\|$ and instead solve

$$\min_{(\beta, \beta_0)} \frac{1}{2} \|\beta\|^2$$

$$\text{subject to } y_i(\beta^T x_i + \beta_0) \geq 1, \quad i = 1, \dots, n$$

Recall that

$$\frac{1}{\|\beta\|} (\beta^T x_i + \beta_0)$$

is the signed distance from x_i to the boundary. Thus, the constraints define a **margin** of thickness $1/\|\beta\|$ around the decision boundary that doesn't contain any training points.

A convex formulation

$$\min_{(\beta, \beta_0)} \frac{1}{2} \|\beta\|^2$$

subject to $y_i(\beta^T x_i + \beta_0) \geq 1, \quad i = 1, \dots, n$

A convex formulation

$$\min_{(\beta, \beta_0)} \frac{1}{2} \|\beta\|^2$$

subject to $y_i(\beta^T x_i + \beta_0) \geq 1, \quad i = 1, \dots, n$

This optimization problem is [convex](#); using Lagrange duality, it can be shown that the optimal solution (β_0^*, β^*) can be described in terms of the [Lagrange multipliers](#)

$\alpha_1, \dots, \alpha_n \geq 0$, such that:

$$\alpha_i [y_i(\beta^T x_i + \beta_0) - 1] = 0, \quad \forall i$$

A convex formulation

$$\min_{(\beta, \beta_0)} \frac{1}{2} \|\beta\|^2$$

$$\text{subject to } y_i(\beta^T x_i + \beta_0) \geq 1, \quad i = 1, \dots, n$$

This optimization problem is **convex**; using Lagrange duality, it can be shown that the optimal solution (β_0^*, β^*) can be described in terms of the **Lagrange multipliers**

$\alpha_1, \dots, \alpha_n \geq 0$, such that:

$$\alpha_i [y_i(\beta^T x_i + \beta_0) - 1] = 0, \quad \forall i$$

Thus,

- if $\alpha_i > 0$, then $y_i(\beta^T x_i + \beta_0) = 1$, and x_i is on the boundary of the slab
- if $y_i(\beta^T x_i + \beta_0) > 1$, then x_i is not on the boundary of the slab, and $\alpha_i = 0$

A convex formulation

$$\min_{(\beta, \beta_0)} \frac{1}{2} \|\beta\|^2$$

subject to $y_i(\beta^T x_i + \beta_0) \geq 1, \quad i = 1, \dots, n$

This optimization problem is **convex**; using Lagrange duality, it can be shown that the optimal solution (β_0^*, β^*) can be described in terms of the **Lagrange multipliers** $\alpha_1, \dots, \alpha_n \geq 0$, such that:

$$\alpha_i [y_i(\beta^T x_i + \beta_0) - 1] = 0, \quad \forall i$$

Thus,

- if $\alpha_i > 0$, then $y_i(\beta^T x_i + \beta_0) = 1$, and x_i is on the boundary of the slab
- if $y_i(\beta^T x_i + \beta_0) > 1$, then x_i is not on the boundary of the slab, and $\alpha_i = 0$

The points on the boundary of the slab are called **support vectors**.

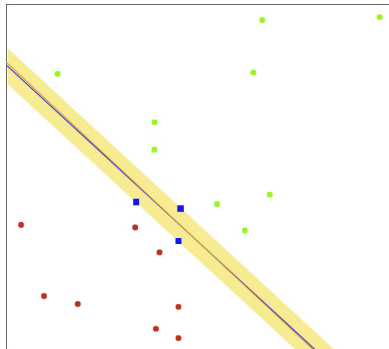


Figure taken from Hastie, Tibshirani and Friedman, 2nd ed.

Discussion of optimal separating hyperplanes

- If the two classes can be linearly separated, then the optimal separating hyperplane separates the two classes and maximizes the distance to the closest point from either class.
- There is a unique solution.
- Tend to have better classification performance on test data.
- Used as a building block in Support Vector Machines (SVMs) and their variants.
 - There exist many off-the-shelf software implementations of SVMs, such as `libsvm`.