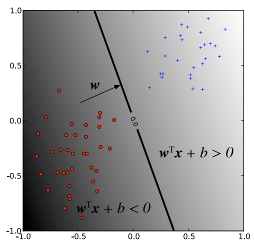
Linear models: the perceptron and closest centroid algorithms

Chapter 1, 7



Preliminaries

Definition: The Euclidean dot product between two vectors is the expression

$$\mathbf{w}^T\mathbf{x} = \sum_{i=1}^{d} w_i x_i$$

The dot product is also referred to as inner product or scalar product.

It is sometimes denoted as $\mathbf{w} \cdot \mathbf{x}$

(hence the name dot product).

Geometric interpretation. The dot product between two unit vectors is the cosine of the angle between them.

The dot product between a vector and a unit vector is the length of its projection in that direction.

Labeled data

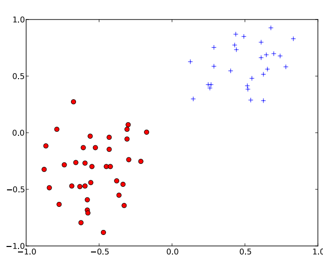
A labeled dataset:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

Where $\mathbf{x}_i \in \mathbb{R}^d$ are d-dimensional vectors

The labels:

are discrete for classification problems (e.g. +1, -1) for binary classification



Labeled data

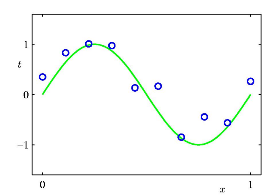
A labeled dataset:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

Where $\mathbf{x}_i \in \mathbb{R}^d$ are d-dimensional vectors

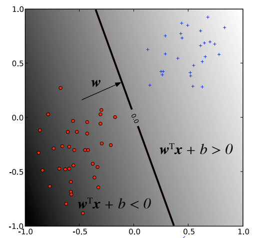
The labels:

are continuous values for a regression problem

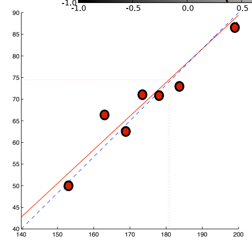


Linear models

Linear models for classification

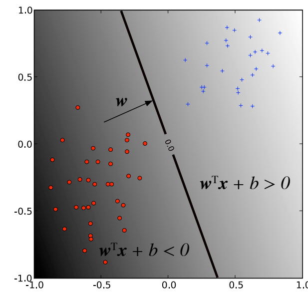


Linear models for regression



5

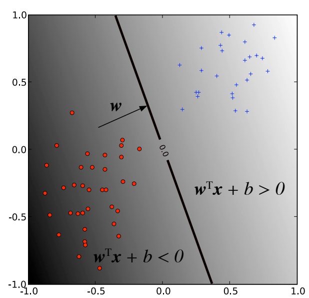
Linear models for classification



Discriminant/scoring function: $f(\mathbf{x}) = \underbrace{\mathbf{w}}_{\text{weight vector}} \cdot \mathbf{x} + \underbrace{b}_{\text{bias}}$

6

Linear models for classification



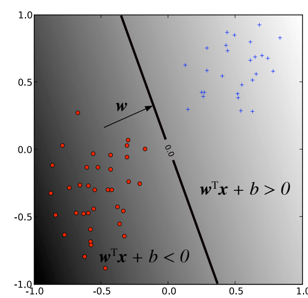
Decision boundary:

all x such that $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = 0$

For linear models the the decision boundary is a line in 2-d, a plane in 3-d and a hyperplane in higher dimensions

7

Linear models for classification

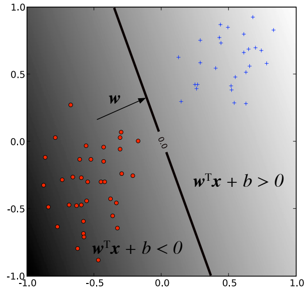


Using the discriminant to make a prediction: $\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$

the sign function equals 1 when its argument is positive and -1 otherwise

8

Linear models for classification



Decision boundary:
all x such that $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = 0$

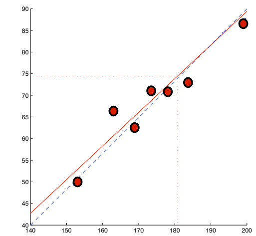
What can you say about the decision boundary when b = 0?

9

Linear models for regression

When using a linear model for regression the scoring function is the prediction:

$$\hat{y} = \mathbf{w} \cdot \mathbf{x} + b$$



10

Why linear?

- It's a good baseline: always start simple
- Linear models are stable
- Linear models are less likely to overfit the training data because they have relatively less parameters. Can sometimes underfit. Often all you need when the data is high dimensional.
- Lots of scalable algorithms

11

Closest centroid classifier

Define:

$$\boldsymbol{\mu}^{(+)} = \frac{1}{Pos} \sum_{\{i|y_i=1\}} \mathbf{x}_i \qquad \boldsymbol{\mu}^{(-)} = \frac{1}{Neg} \sum_{\{i|y_i=-1\}} \mathbf{x}_i$$

where Pos/Neg is the number of positive/negative examples.
This is the center of mass of the positive/negative examples.

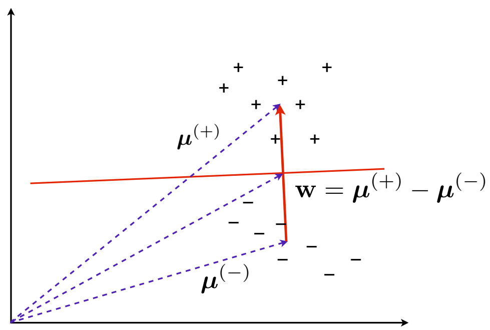
Classify an input x according to which center of mass it is closest to.

Let's express this as a linear classifier!

See page 21-22 in the textbook

12

Closest centroid classifier

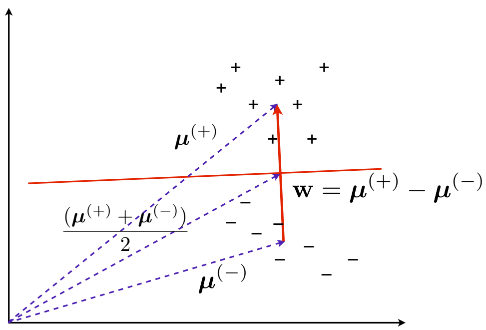


Our hyperplane is going be perpendicular to the vector that connects the two centers of mass. Therefore:

$$\mathbf{w} = \mu^{(+)} - \mu^{(-)}$$

13

Closest centroid classifier

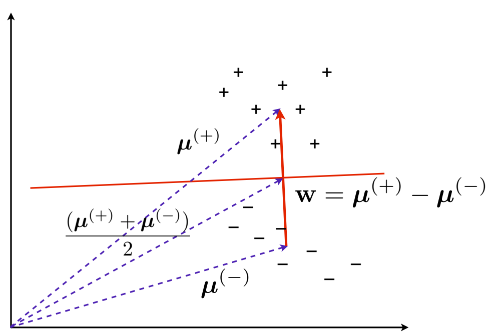


To find the bias term we use the fact that the midpoint between the means is on the hyperplane, i.e.

$$\mathbf{w} \cdot \frac{(\mu^{(+)} + \mu^{(-)})}{2} + b = 0$$

14

Closest centroid classifier



With a little algebra:
$$b = -\frac{1}{2}(\mu^{(+)} - \mu^{(-)})(\mu^{(+)} + \mu^{(-)})$$
$$= -\frac{1}{2}(||\mu^{(+)}||^2 - ||\mu^{(-)}||^2)$$

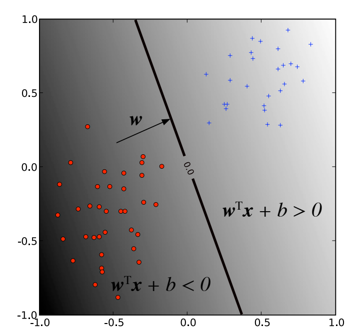
The norm of a vector:
$$||\mathbf{x}||^2 = \mathbf{x} \cdot \mathbf{x}$$

15

Linearly separable data

Linearly separable data: there exists a linear decision boundary separating the classes.

Example:



16

The bias and homogeneous coordinates

In some cases we will use algorithms that learn a discriminant function without a bias term. This does not reduce the expressivity of the model because we can obtain a bias using the following trick:

Add another dimension x_0 to each input and set it to 1.

Learn a weight vector of dimension d+1 in this extended space, and interpret w_0 as the bias term. With the notation

$$\mathbf{w} = (w_1, \ldots, w_d) \qquad \tilde{\mathbf{w}} = (w_0, w_1, \ldots, w_d)$$

$$\tilde{\mathbf{x}} = (1, x_1, \ldots, x_d)$$

We have that:

$$\tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}} = w_0 + \mathbf{w} \cdot \mathbf{x}$$

See page 4 in the book

The perceptron algorithm

Idea: iterate over the training examples, and update the weight vector w such that $\mathbf{x}_i$ is more likely to be correctly classified.

Let's assume that $\mathbf{x}_i$ is misclassified, and is a positive example i.e.

$$\mathbf{w} \cdot \mathbf{x}_i < 0$$

Note: we're learning a classifier without a bias term

We would like to update **w** to **w'** such that

$$\mathbf{w}' \cdot \mathbf{x}_i > \mathbf{w} \cdot \mathbf{x}_i$$

This can be achieved by choosing

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

Where $0 < \eta \leq 1$ is the learning rate

Section 7.2 in the book

The perceptron algorithm

If $\mathbf{x}_i$ is a negative example, the update needs to be opposite.

Overall, we can summarize the two cases as:

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

```
Input:  labeled data D in homogeneous coordinates
Output:  weight vector w

w = 0
converged = false
while not converged :
    converged = true
    for i in 1,…,|D| :
        if xi is misclassified update w and set
            converged=false
```


The perceptron algorithm

The algorithm makes sense, but let's try to derive in a more principled way.

The algorithm is trying to find a vector **w** that separates positive from negative examples.

We can express that as:

$$y_i \mathbf{w}^{\mathsf{T}} \mathbf{x}_i > 0, \quad i = 1, \ldots, n$$

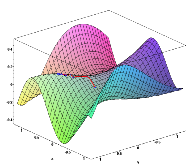
For a given weight vector **w** the degree to which this does not hold can be expressed as:

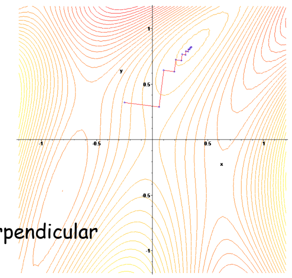
$$E(\mathbf{w}) = - \sum_{i:\ \mathbf{x}_i \text{ is misclassified}} y_i \mathbf{w}^{\mathsf{T}} \mathbf{x}_i$$

We want to find **w** that minimizes or maximizes this criterion?

Digression: gradient descent

Given a function E(**w**), the gradient is the direction of steepest ascent
Therefore to minimize E(**w**), take a step in the direction of the negative of the gradient





Notice that the gradient is perpendicular to contours of equal E(**w**)

Images from http://en.wikipedia.org/wiki/Gradient_descent

21

Gradient descent

We can now express gradient descent as:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \nabla E(\mathbf{w})$$
$$\mathbf{w}(t) - \eta \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}$$

where
$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \left(\frac{\partial E(\mathbf{w})}{\partial w_1}, \ldots, \frac{\partial E(\mathbf{w})}{\partial w_d} \right)^{\mathsf{T}}$$

And **w**(t) is the weight vector at iteration t

The constant η is called the step size (learning rate when used in the context of machine learning).

22

The perceptron algorithm

Let's apply gradient descent to the perceptron criterion:

$$E(\mathbf{w}) = - \sum_{i:\ \mathbf{x}_i \text{ is misclassified}} y_i \mathbf{w}^{\mathsf{T}} \mathbf{x}_i$$

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = - \sum_{i:\ \mathbf{x}_i \text{ is misclassified}} y_i \mathbf{x}_i$$

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}$$
$$= \mathbf{w}(t) + \eta \sum_{i:\ \mathbf{x}_i \text{ is misclassified}} y_i \mathbf{x}_i$$

Which is exactly the perceptron algorithm!

23

The perceptron algorithm

The algorithm is guaranteed to converge if the data is linearly separable, and does not converge otherwise.

Issues with the algorithm:

- The algorithm chooses an arbitrary hyperplane that separates the two classes. It may not be the best one from the learning perspective.
- Does not converge if the data is not separable (can halt after a fixed number of iterations).

There are variants of the algorithm that address these issues (to some extent).

24

Perceptron for regression

Replace the update equation with:

$$\mathbf{w}' = \mathbf{w} + \eta(y_i - \hat{y}_i)^2\mathbf{x}_i$$

This is not likely to converge so the algorithm is run for a fixed number of training epochs

Training epoch – one complete run through the training data