CS 6316 Machine Learning

Linear Predictors

Yangfeng Ji

Department of Computer Science
University of Virginia

Overview

1. Review: Linear Functions
2. Perceptron
3. Logistic Regression
4. Linear Regression

Review: Linear Functions

Linear Predictors

Linear predictors discussed in this course

- halfspace predictors
- logistic regression classifiers
- linear SVMs (lecture on support vector machines)
- naive Bayes classifiers (lecture on generative models)
- linear regression predictors

Linear Predictors

Linear predictors discussed in this course

- halfspace predictors
- logistic regression classifiers
- linear SVMs (lecture on support vector machines)
- naive Bayes classifiers (lecture on generative models)
- linear regression predictors

A common core form of these linear predictors

$$h_{\boldsymbol{w},b} = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = \left(\sum_{i=1}^{d} w_i x_i \right) + b \tag{1}$$

where $\boldsymbol{w}$ is the weights and b is the bias

Alternative Form

Given the original definition of a linear function

$$h_{\boldsymbol{w},b} = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = \Big(\sum_{i=1}^{d} w_i x_i \Big) + b, \tag{2}$$

we could redefine it in a more compact form

$$\begin{aligned} \boldsymbol{w} &\leftarrow (w_1, w_2, \ldots, w_d, b)^{\mathsf{T}} \\ \boldsymbol{x} &\leftarrow (x_1, x_2, \ldots, x_d, 1)^{\mathsf{T}} \end{aligned}$$

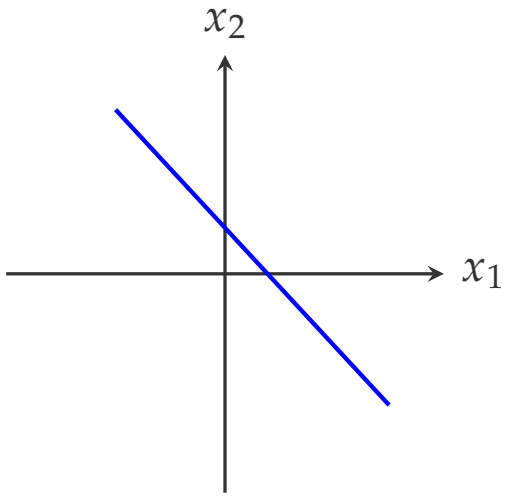
and then

$$h_{\boldsymbol{w},b}(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle \tag{3}$$

Linear Functions

Consider a two-dimensional case with $\boldsymbol{w} = (1, 1, -0.5)$

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} = x_1 + x_2 - 0.5 \tag{4}$$



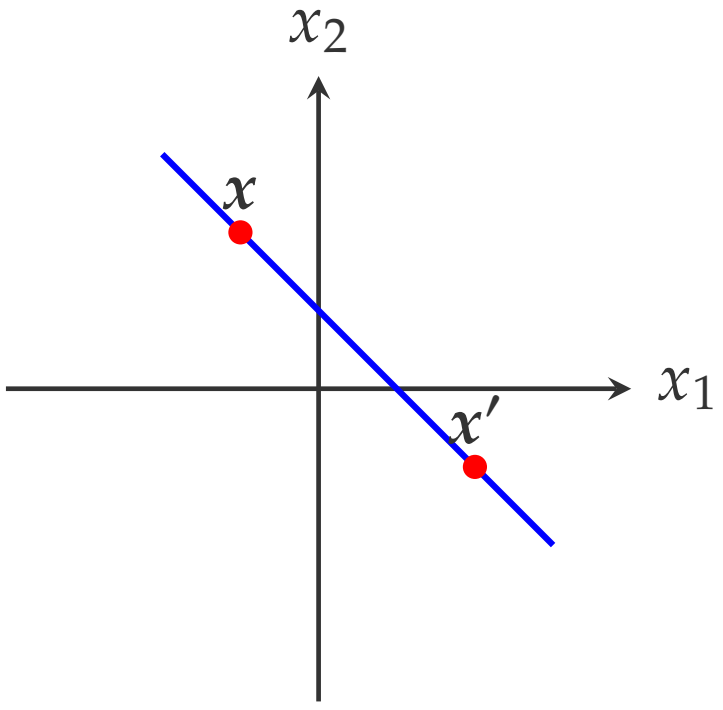
Different values of $f(\boldsymbol{x})$ map to different areas on this 2-D space. For example, the following equation defines the blue line L.

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} = 0 \tag{5}$$

Properties of Linear Functions (II)

For any two points x and x' lying in the line

$$f(x) - f(x') = w^\top x - w^\top x' = 0 \tag{6}$$



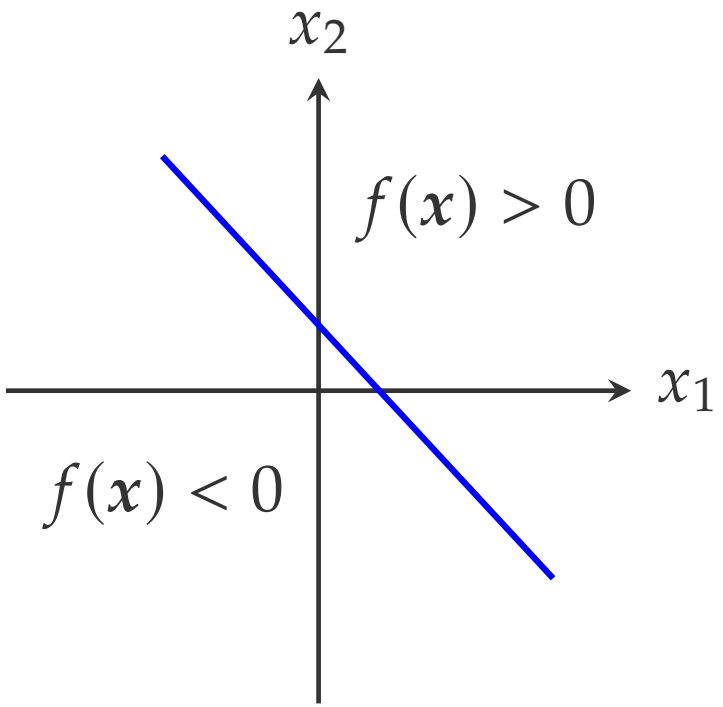
[Friedman et al., 2001, Section 4.5]

Properties of Linear Functions (III)

Furthermore,

$$f(\boldsymbol{x}) = x_1 + x_2 - 0.5 = 0 \tag{7}$$

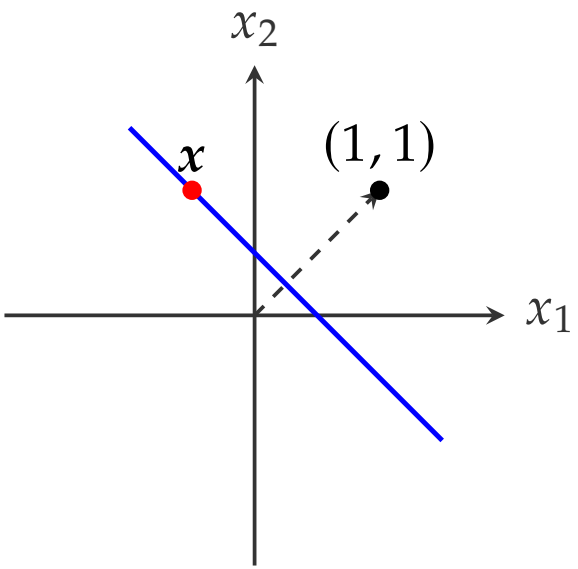
separates the 2-D space $\mathbb{R}^2$ into two half spaces



Properties of Linear Functions (IV)

From the perspective of linear projection, $f(\boldsymbol{x}) = 0$ defines the vectors on this 2-D space, whose projections onto the direction $(1, 1)$ have the same magnitude 0.5

$$x_1 + x_2 - 0.5 = 0 \Rightarrow (x_1, x_2) \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 0.5 \tag{8}$$

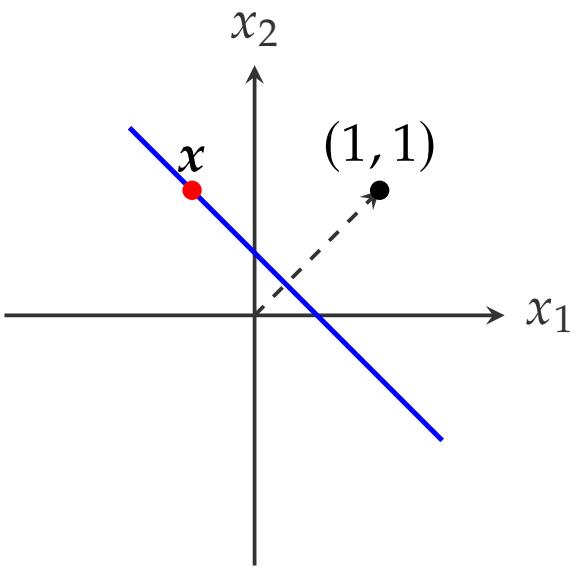


[Friedman et al., 2001, Section 4.5]

Properties of Linear Functions (IV)

From the perspective of linear projection, $f(\boldsymbol{x}) = 0$ defines the vectors on this 2-D space, whose projections onto the direction $(1, 1)$ have the same magnitude 0.5

$$x_1 + x_2 - 0.5 = 0 \Rightarrow (x_1, x_2) \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 0.5 \tag{8}$$



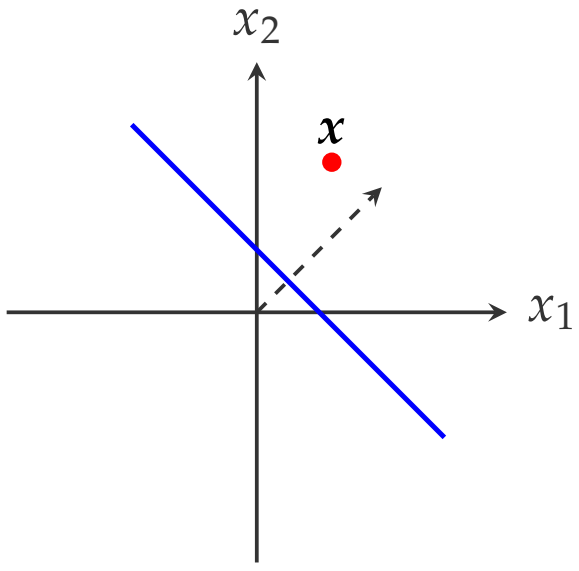
This idea can be generalized to compute the distance between a point and a line.

[Friedman et al., 2001, Section 4.5]

Properties of Linear Functions (IV)

The distance of point x to line $L : f(x) = \langle w, x \rangle = 0$ is given by

$$\frac{f(x)}{\|w\|_2} = \frac{\langle w, x \rangle}{\|x\|_2} = \langle \frac{w}{\|w\|_2}, x \rangle \tag{9}$$



[Friedman et al., 2001, Section 4.5]

Perceptron

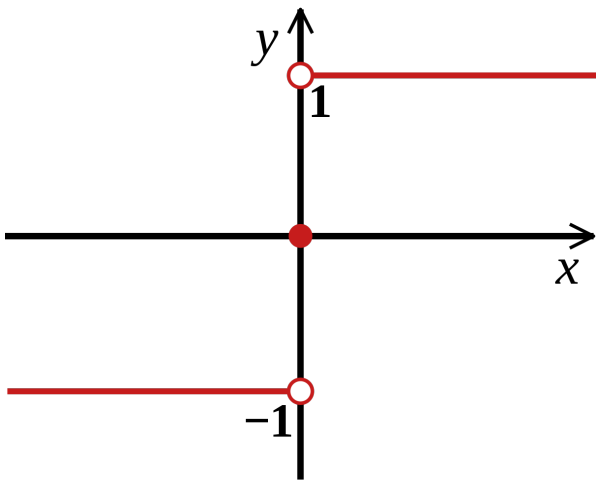
Halfspace Hypothesis Class

- $\mathcal{X} = \mathbb{R}^d$
- $\mathcal{Y} = \{-1, +1\}$
- Halfspace hypothesis class

$$\mathcal{H}_{\text{half}} = \{\text{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle) : \boldsymbol{w} \in \mathbb{R}^d\} \tag{10}$$

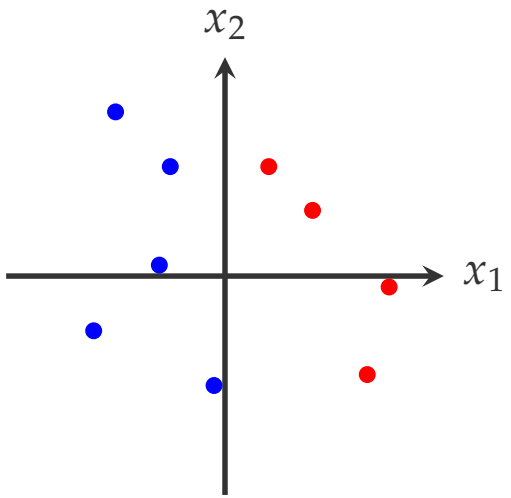
  which is an infinite hypothesis space.

The sign function $y = \text{sign}(x)$ is defined as



Linearly Separable Cases

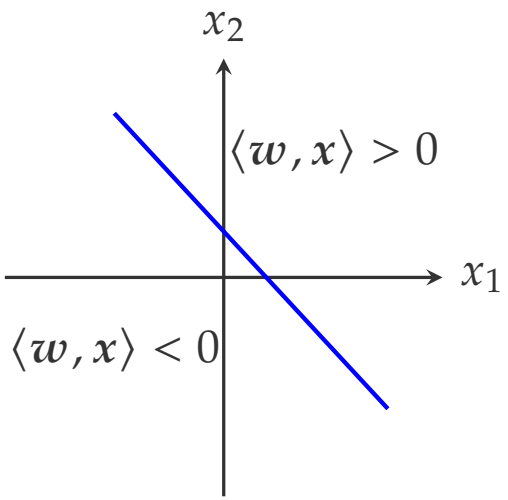
- The algorithm can find a hyperplane to separate all positive examples from negative examples
- It is a special realizable cases with linear classifiers. In other words, the realization assumption holds with $\mathscr{H}_{\text{half}}$

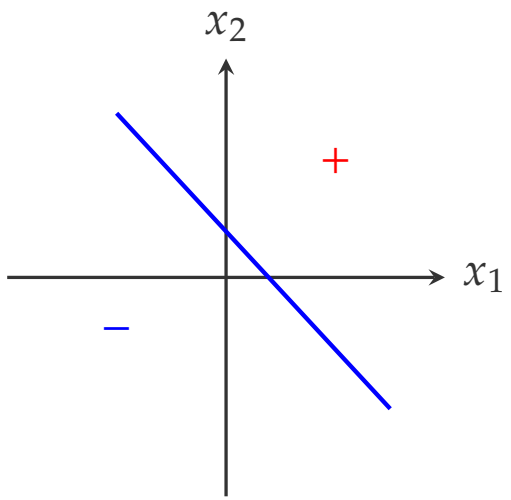


Prediction Rule

The prediction rule of a half-space predictor is based on the sign of $h(x) = \text{sign}(\langle w, x \rangle)$

$$h(x) = \begin{cases} +1 & \langle w, x \rangle > 0 \\ -1 & \langle w, x \rangle < 0 \end{cases} \tag{11}$$





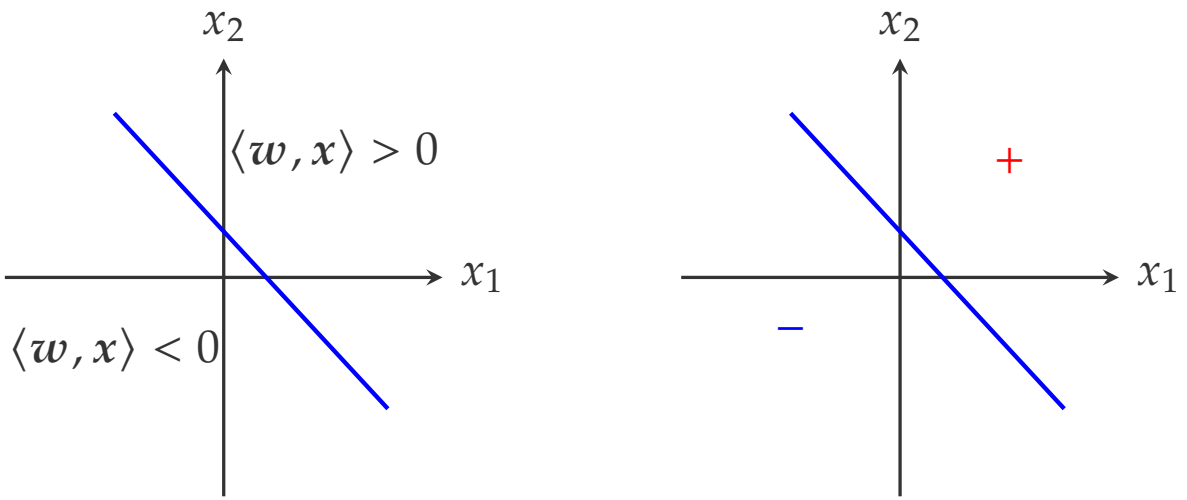
Prediction Rule

The prediction rule of a half-space predictor is based on the sign of $h(\boldsymbol{x}) = \text{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle)$

$$h(\boldsymbol{x}) = \begin{cases} +1 & \langle \boldsymbol{w}, \boldsymbol{x} \rangle > 0 \\ -1 & \langle \boldsymbol{w}, \boldsymbol{x} \rangle < 0 \end{cases} \tag{11}$$

or,

$$h(\boldsymbol{x}) = y' \quad \text{if } y' \in \{-1, +1\} \text{ and } y' \langle \boldsymbol{w}, \boldsymbol{x} \rangle > 0 \tag{12}$$



Perceptron Algorithm

The perceptron algorithm is defined as

1: **Input**: $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m))\}$
2: Initialize $\boldsymbol{w}^{(0)} = (0, \ldots, 0)$

9: **Output**: $\boldsymbol{w}^{(T)}$

Perceptron Algorithm

The perceptron algorithm is defined as

1: **Input**: $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m))\}$
2: Initialize $\boldsymbol{w}^{(0)} = (0, \ldots, 0)$
3: **for** $t = 1, 2, \cdots, T$ **do**
4: $\quad i \leftarrow t \mod m$

8: **end for**
9: **Output**: $\boldsymbol{w}^{(T)}$

Perceptron Algorithm

The perceptron algorithm is defined as

1: **Input**: $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m))\}$
2: Initialize $\boldsymbol{w}^{(0)} = (0, \ldots, 0)$
3: **for** $t = 1, 2, \cdots, T$ **do**
4: $\quad i \leftarrow t \mod m$
5: $\quad$ **if** $y_i \langle \boldsymbol{w}^{(t)}, \boldsymbol{x}_i \rangle \leq 0$ **then**
6: $\quad\quad \boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} + y_i \boldsymbol{x}_i$ // *updating rule*
7: $\quad$ **end if**
8: **end for**
9: **Output**: $\boldsymbol{w}^{(T)}$

Perceptron Algorithm

The perceptron algorithm is defined as

1: **Input**: $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m))\}$
2: Initialize $\boldsymbol{w}^{(0)} = (0, \ldots, 0)$
3: **for** $t = 1, 2, \cdots, T$ **do**
4: $\quad i \leftarrow t \mod m$
5: $\quad$ **if** $y_i \langle \boldsymbol{w}^{(t)}, \boldsymbol{x}_i \rangle \leq 0$ **then**
6: $\quad\quad \boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} + y_i \boldsymbol{x}_i$ // *updating rule*
7: $\quad$ **end if**
8: **end for**
9: **Output**: $\boldsymbol{w}^{(T)}$

Exercise: Implementing this algorithm with a smple example

Two Questions

The updating rule can be break down into two cases:

$$w^{(t+1)} \leftarrow w^{(t)} + y_i x_i \tag{13}$$

- For $y_i = +1$, $w^{(t+1)} \leftarrow w^{(t)} + x_i$
- For $y_i = -1$, $w^{(t+1)} \leftarrow w^{(t)} - x_i$

Two Questions

The updating rule can be break down into two cases:

$$w^{(t+1)} \leftarrow w^{(t)} + y_i x_i \tag{13}$$

- For $y_i = +1$, $w^{(t+1)} \leftarrow w^{(t)} + x_i$
- For $y_i = -1$, $w^{(t+1)} \leftarrow w^{(t)} - x_i$

Two questions:

- How the updating rule can help?
- How many updating steps the algorithm needs?

The Updating Rule

At time step t, given the training example (x_i, y_i) and the current weight $w^{(t)}$

$$y_i \langle w^{(t+1)}, x_i \rangle = y_i \langle w^{(t)} + y_i x_i, x_i \rangle \quad (14)$$

$$= y_i \langle w^{(t)}, x_i \rangle + \|x_i\|^2 \quad (15)$$

- $w^{(t+1)}$ gives a higher value of $y_i \langle w^{(t+1)}, x_i \rangle$ on predicting x_i than $w^{(t)}$
- the updating is affected by the norm of x_i, $\|x_i\|^2$

Theorem

Assume that $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ is separable. Let

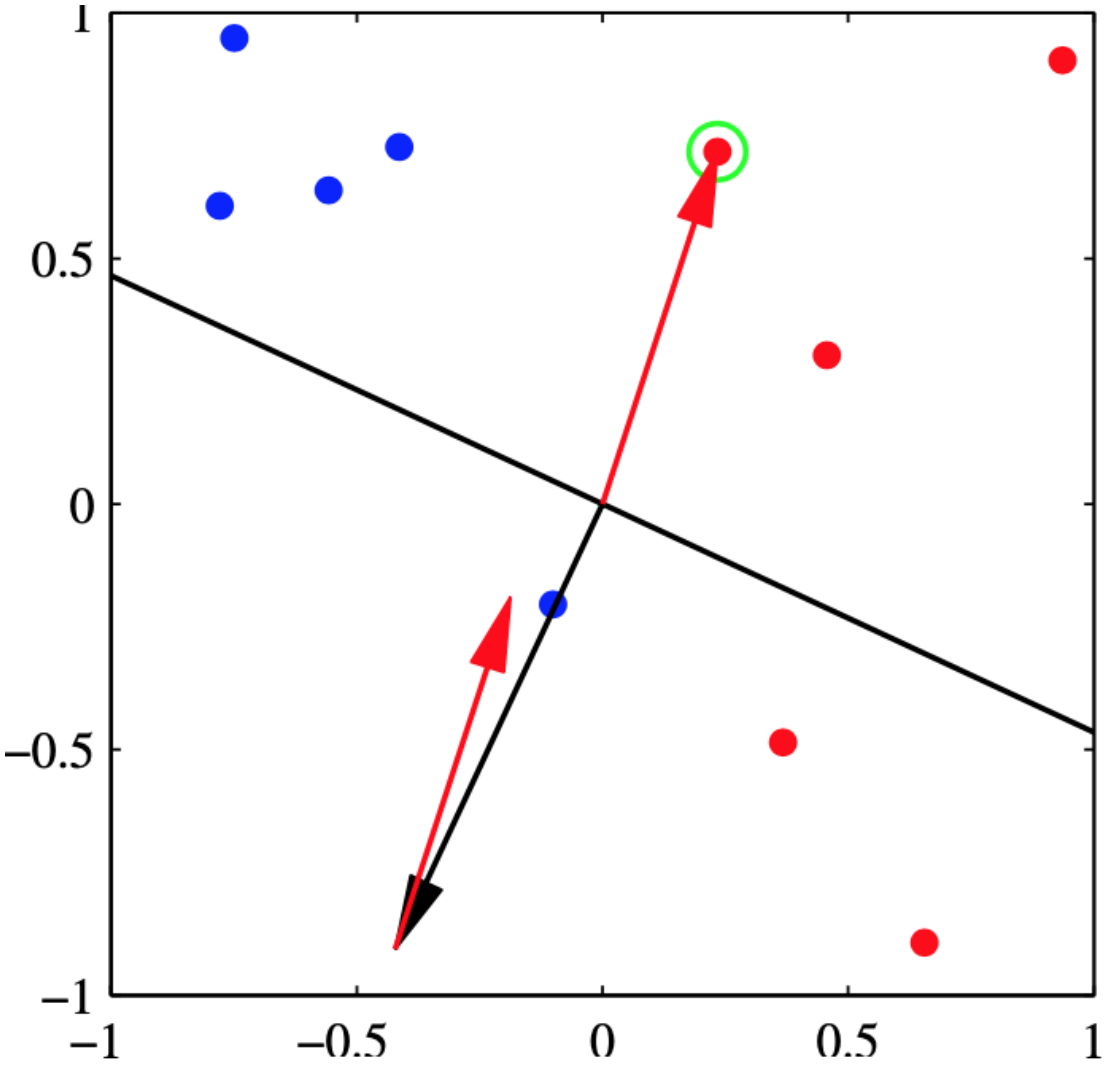
- $B = \min\{\|\boldsymbol{w}\| : \forall i \in [m], y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle \geq 1\}$, and
- $R = \max_i \|\boldsymbol{x}_i\|$.

Then, the Perceptron algorithm stops after at most $(RB)^2$ iterations, and when it stops it holds that $\forall i \in [m]$,

$$y_i \langle \boldsymbol{w}^{(t)}, \boldsymbol{x} \rangle > 0 \tag{16}$$

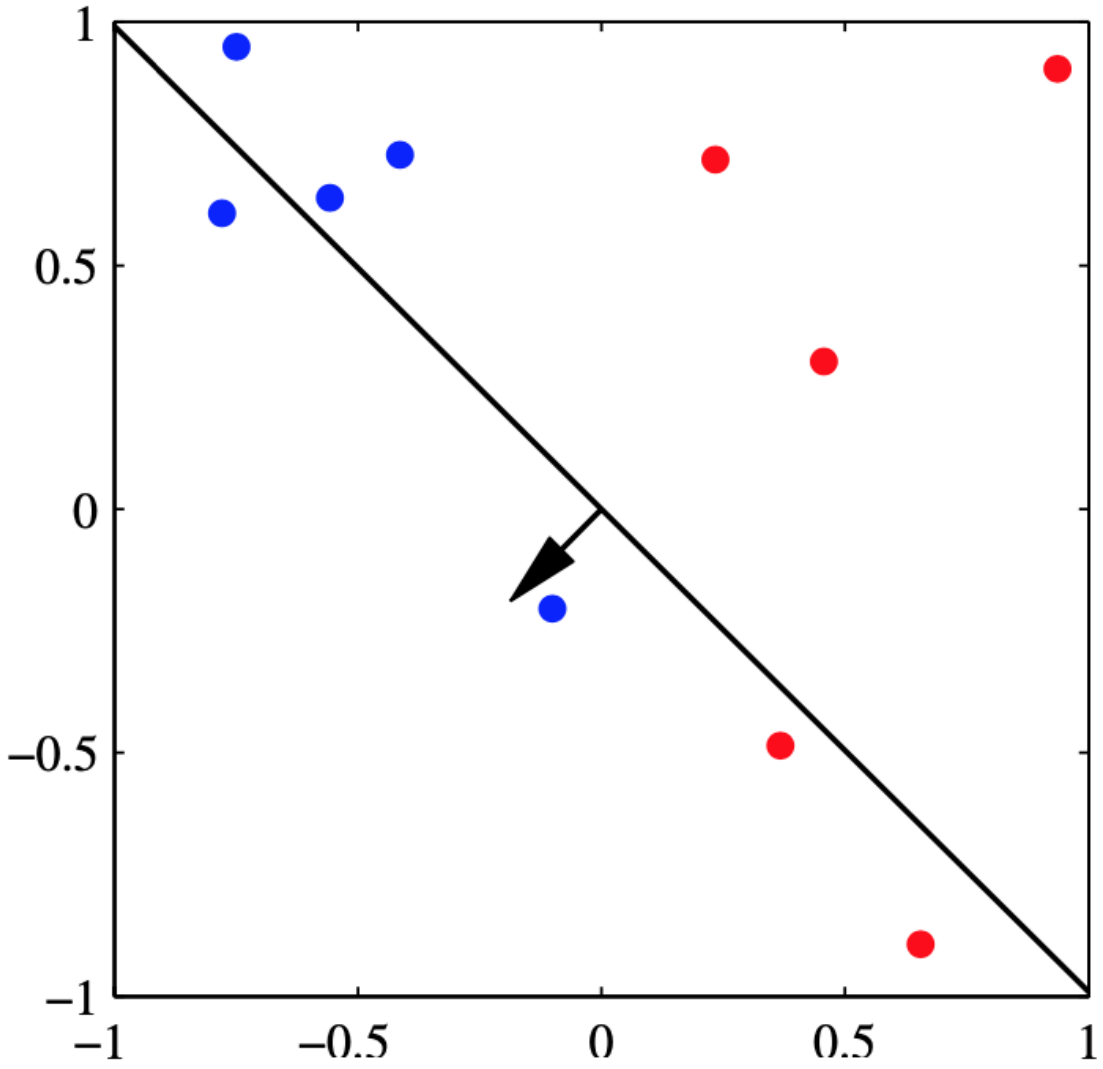
- A realizable case with infinite hypothesis space
- Finish training in finite steps

Example



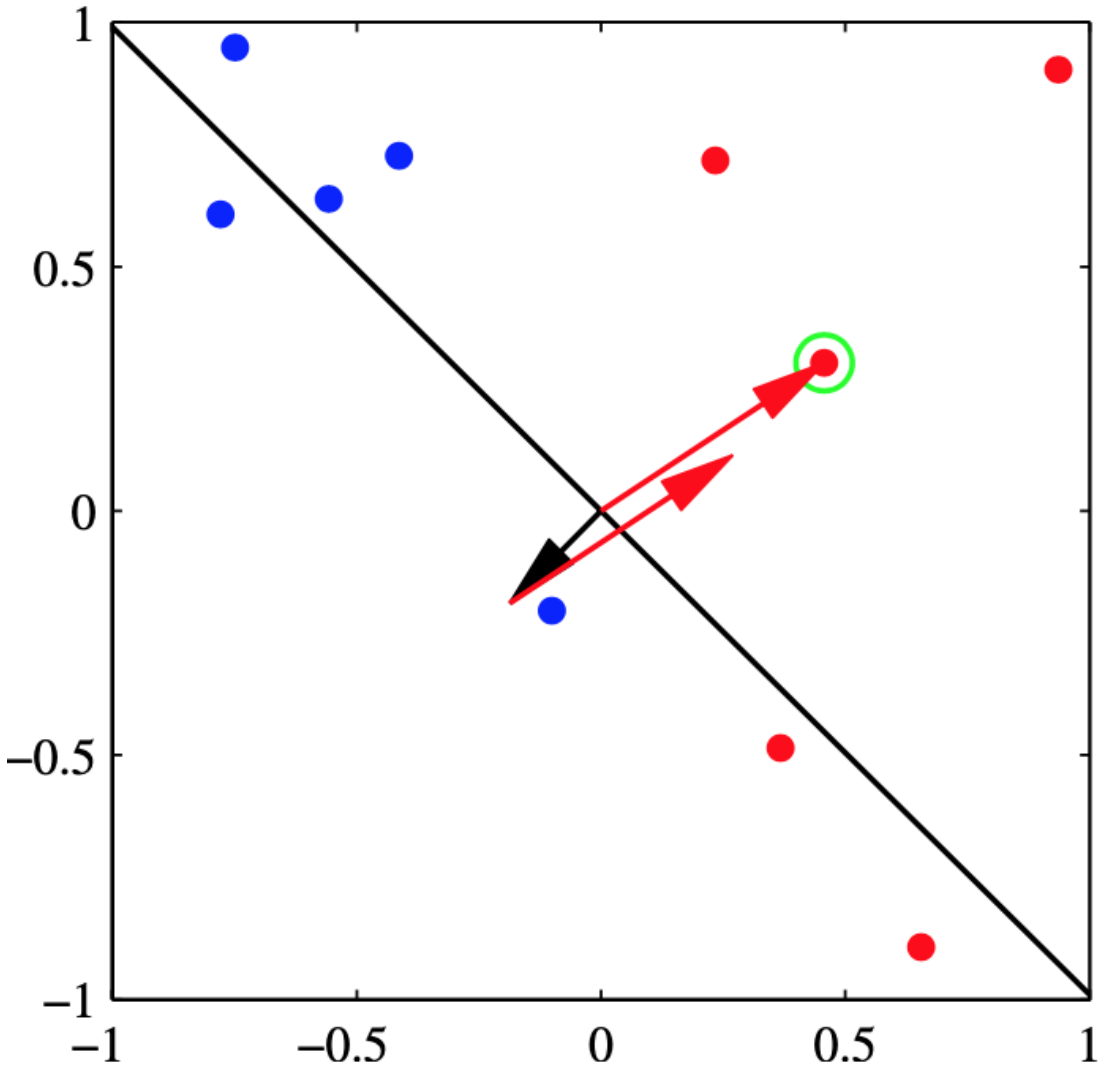
[Bishop, 2006, Page 195]

Example



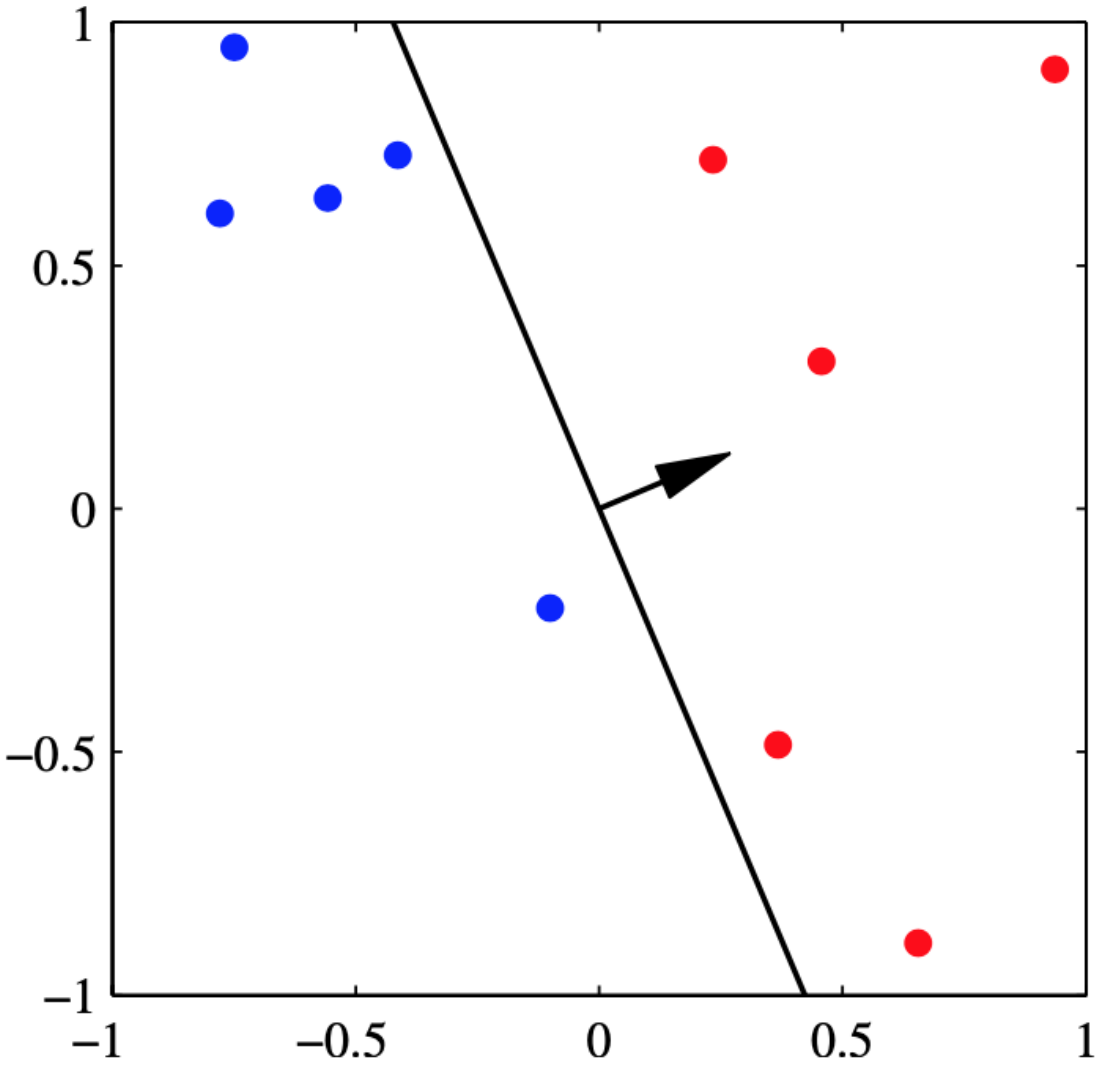
[Bishop, 2006, Page 195]

Example



[Bishop, 2006, Page 195]

Example



[Bishop, 2006, Page 195]

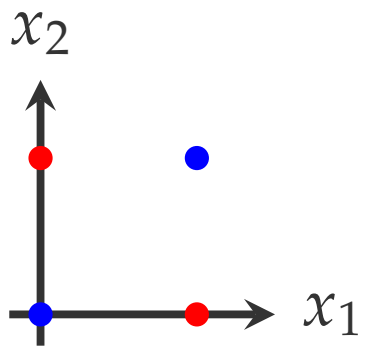
The XOR Example: a Non-separable Case

- $X_1, X_2 \in \{0, 1\}$
- the XOR operation is defined as

$$Y = X_1 \oplus X_2$$

where

$$Y = \begin{cases} 1 & X_1 \neq X_2 \\ 0 & X_1 = X_2 \end{cases}$$



The XOR Example: Further Comment

Frank Rosenblatt
1928–1969

Rosenblatt's perceptron played an important role in the history of machine learning. Initially, Rosenblatt simulated the perceptron on an IBM 704 computer at Cornell in 1957, but by the early 1960s he had built special-purpose hardware that provided a direct, parallel implementation of perceptron learning. Many of his ideas were encapsulated in "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms" published in 1962. Rosenblatt's work was criticized by Marvin Minksy, whose objections were published in the book "Perceptrons", co-authored with Seymour Papert. This book was widely misinterpreted at the time as showing that neural networks were fatally flawed and could only learn solutions for linearly separable problems. In fact, it only proved such limitations in the case of single-layer networks such as the perceptron and merely conjectured (incorrectly) that they applied to more general network models. Unfortunately, however, this book contributed to the substantial decline in research funding for neural computing, a situation that was not reversed until the mid-1980s. Today, there are many hundreds, if not thousands, of applications of neural networks in widespread use, with examples in areas such as handwriting recognition and information retrieval being used routinely by millions of people.

Logistic Regression

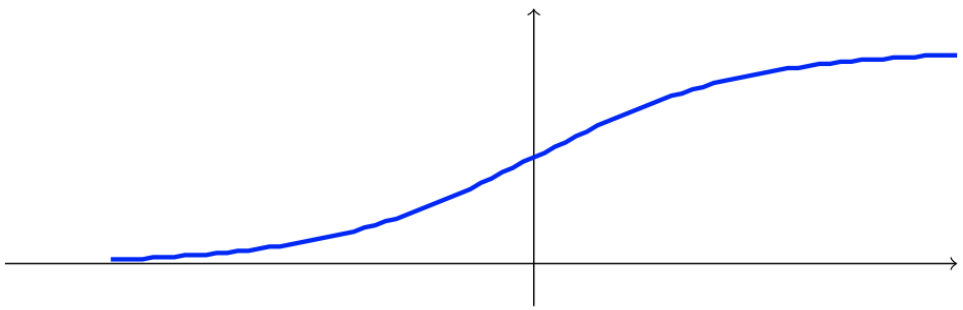
Hypothesis Class

- The hypothesis class of logistic regression is defined as

$$\mathcal{H}_{\text{LR}} = \{\sigma(\langle \boldsymbol{w}, \boldsymbol{x} \rangle) : \boldsymbol{w} \in \mathbb{R}^d\} \tag{17}$$

- The sigmoid function $\sigma(a)$ with $a \in \mathbb{R}$

$$\sigma(a) = \frac{1}{1 + \exp(-a)} \tag{18}$$



Unified Form for Logistic Predictors

- An unified form for $y \in \{-1, +1\}$

$$h(\boldsymbol{x}, y) = \frac{1}{1 + \exp(-y\langle \boldsymbol{w}, \boldsymbol{x} \rangle)} \tag{19}$$

which is similar to the half-space predictors

Unified Form for Logistic Predictors

- An unified form for $y \in \{-1, +1\}$

$$h(\boldsymbol{x}, y) = \frac{1}{1 + \exp(-y\langle \boldsymbol{w}, \boldsymbol{x} \rangle)} \tag{19}$$

  which is similar to the half-space predictors

- Prediction
  1. Compute the the values from Eq. 19 with $y \in \{-1, +1\}$
  2. Pick the y that has bigger value

$$y = \begin{cases} +1 & h(\boldsymbol{x}, +1) > h(\boldsymbol{x}, -1) \\ -1 & h(\boldsymbol{x}, +1) < h(\boldsymbol{x}, -1) \end{cases} \tag{20}$$

A Predictor

Take a close look of the uniform definition of $h(x, y)$

- When $y = +1$

$$h_w(x, +1) = \frac{1}{1 + \exp(-\langle w, x \rangle)}$$

- When $y = -1$

$$\begin{aligned} h(x, -1) &= \frac{1}{1 + \exp(\langle w, x \rangle)} \\ &= \frac{\exp(-\langle w, x \rangle)}{1 + \exp(-\langle w, x \rangle)} \\ &= 1 - \frac{1}{1 + \exp(-\langle w, x \rangle)} \\ &= 1 - h_w(x, +1) \end{aligned}$$

A Linear Classifier?

To justify this is a linear classifier, let take a look the decision boundary given by

$$h(x, +1) = h(x, -1) \tag{21}$$

Specifically, we have

$$\begin{aligned}
\frac{1}{1 + \exp(-\langle w, x \rangle)} &= \frac{1}{1 + \exp(\langle w, x \rangle)} \\
\exp(-\langle w, x \rangle) &= \exp(\langle w, x \rangle) \\
-\langle w, x \rangle &= \langle w, x \rangle \\
2\langle w, x \rangle &= 0
\end{aligned}$$

The decision boundary is a straight line

Risk

For a given training example (x, y), the risk function is defined as the negative log of $h(x, y)$

$$\begin{aligned} L(h_w, (x, y)) &= -\log \frac{1}{1 + \exp(-y\langle w, x\rangle)} \\ &= \log(1 + \exp(-y\langle w, x\rangle)) \end{aligned} \tag{22}$$

Intuitively, minimizing the risk will increase the value of $h(x, y)$

ERM

The **Empirical Risk Minimization** (ERM) problem: given the training set $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$, minimize the following objective function with respect to $\boldsymbol{w}$

$$L(h_{\boldsymbol{w}}, S) = \frac{1}{m} \sum_{i=1}^{m} \log(1 + \exp(-y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)) \tag{23}$$

- A convex function with respect to $\boldsymbol{w}$
- Minimization can be done with gradient-based optimization[1]

[1] more detail will be covered in the lecture of optimization methods

Gradient Descent

- The gradient of $L(h_{\boldsymbol{w}}, S)$ with respect to $\boldsymbol{w}$

$$\frac{dL(h_{\boldsymbol{w}}, S)}{d\boldsymbol{w}} = \frac{1}{m} \sum_{i=1}^{m} \frac{\exp(-y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)}{1 + \exp(-y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)} \cdot (-y_i \boldsymbol{x}_i) \quad (24)$$

Gradient Descent

- The gradient of $L(h_{\boldsymbol{w}}, S)$ with respect to $\boldsymbol{w}$

$$\frac{dL(h_{\boldsymbol{w}}, S)}{d\boldsymbol{w}} = \frac{1}{m}\sum_{i=1}^{m} \frac{\exp(-y_i\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle)}{1+\exp(-y_i\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle)} \cdot (-y_i\boldsymbol{x}_i) \quad (24)$$

- Gradient-based learning

$$\begin{aligned} \boldsymbol{w}^{(\text{new})} &= \boldsymbol{w}^{(\text{old})} - \eta \frac{dL(h_{\boldsymbol{w}}, S)}{d\boldsymbol{w}} \\ &= \boldsymbol{w}^{(\text{old})} + \frac{\eta}{m}\sum_{i=1}^{m} \frac{\exp(-y_i\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle)}{1+\exp(-y_i\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle)} \cdot (y_i\boldsymbol{x}_i) \end{aligned}$$

where η is the updating step size.

Gradient Descent

- The gradient of $L(h_{\boldsymbol{w}}, S)$ with respect to $\boldsymbol{w}$

$$\frac{dL(h_{\boldsymbol{w}}, S)}{d\boldsymbol{w}} = \frac{1}{m}\sum_{i=1}^{m} \frac{\exp(-y_i\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle)}{1+\exp(-y_i\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle)} \cdot (-y_i\boldsymbol{x}_i) \quad (24)$$

- Gradient-based learning

$$\begin{aligned} \boldsymbol{w}^{(\text{new})} &= \boldsymbol{w}^{(\text{old})} - \eta \frac{dL(h_{\boldsymbol{w}}, S)}{d\boldsymbol{w}} \\ &= \boldsymbol{w}^{(\text{old})} + \frac{\eta}{m}\sum_{i=1}^{m} \frac{\exp(-y_i\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle)}{1+\exp(-y_i\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle)} \cdot (y_i\boldsymbol{x}_i) \end{aligned}$$

  where η is the updating step size.

- *Exercise*: prove Eq. 24

More Analysis on Gradient Descent

Gradient-based learning

$$\boldsymbol{w}^{(\text{new})} = \boldsymbol{w}^{(\text{old})} + \frac{\eta}{m} \sum_{i=1}^{m} \frac{\exp(-y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)}{1 + \exp(-y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)} \cdot (y_i \boldsymbol{x}_i) \quad (25)$$

The update is

- directed by the true label y_i (as in the Perceptron algorithm)
- proportional to the prediction value of the opposite label (not like the Perceptron algorithm)

Updating Rules

Consider the case where the learning algorithms only take *one* training example at each time

- Logistic regression

$$\boldsymbol{w}^{(\text{new})} = \boldsymbol{w}^{(\text{old})} + \eta \cdot \frac{\exp(-y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)}{1 + \exp(-y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)} \cdot (y_i \boldsymbol{x}_i) \quad (26)$$

- Perceptron algorithm

$$\boldsymbol{w}^{(\text{new})} = \boldsymbol{w}^{(\text{old})} + y_i \boldsymbol{x}_i \quad (27)$$

only applies when the prediction is wrong

A Probabilistic View of Logistic Regression

- From a probabilistic view, logistic regression defines the probability of a possible label y given the input $\boldsymbol{x}$

$$p_{\boldsymbol{w}}(Y = y \mid \boldsymbol{x}) = \frac{1}{1 + \exp(-y\langle \boldsymbol{w}, \boldsymbol{x} \rangle)} \tag{28}$$

  where Y is a random variable with $Y \in \{-1, +1\}$

- The previous prediction rule is equivalent to

$$\hat{y} = \begin{cases} +1 & \text{if } p(Y = +1 \mid \boldsymbol{x}) > p(Y = -1 \mid \boldsymbol{x}) \\ -1 & \text{if } p(Y = +1 \mid \boldsymbol{x}) < p(Y = -1 \mid \boldsymbol{x}) \end{cases} \tag{29}$$

Likelihood Function

Given the training set $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$, the likelihood function is defined as

$$\text{Lik}(\boldsymbol{x}) = \prod_{i=1}^{m} p_{\boldsymbol{w}}(y_i \mid \boldsymbol{x}_i) \tag{30}$$

Likelihood Principle: All the information about $\boldsymbol{w}$ is contained in the likelihood function for $\boldsymbol{w}$ given S.

[Berger and Wolpert, 1988]

Maximum Likelihood Estimation

Given the training set S, learning with ERM is equivalent to the Maximum Likelihood Estimation in Statistics

- Log-likelihood function

$$\begin{aligned} \ell(\boldsymbol{w}) &= \sum_{i=1}^{m} \log p_{\boldsymbol{w}}(y_i \mid \boldsymbol{x}_i) \\ &= \sum_{i=1}^{m} \log \frac{1}{1+\exp(-y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)} \\ &= -\sum_{i=1}^{m} \log(1+\exp(-y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)) \end{aligned} \tag{31}$$

- Maximize the log-likelihood function

$$\max_{\boldsymbol{w}} \ell(\boldsymbol{w}) = \min_{\boldsymbol{w}} -\ell(\boldsymbol{w}) = \min_{\boldsymbol{w}} L(h_{\boldsymbol{w}}, S) \tag{32}$$

Gradient Descent, revisited

Recall the gradient-based learning on the previous slide

$$\begin{aligned} \boldsymbol{w}^{(\text{new})} &= \boldsymbol{w}^{(\text{old})} + \eta \sum_{i=1}^{m} \frac{\exp(-y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)}{1 + \exp(-y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)} \cdot (y_i \boldsymbol{x}_i) \\ &= \boldsymbol{w}^{(\text{old})} + \eta \sum_{i=1}^{m} (1 - p(y_i \mid \boldsymbol{x}_i)) \cdot y_i \boldsymbol{x}_i \end{aligned} \tag{33}$$

- If $p(y_i \mid \boldsymbol{x}_i) \to 0$, wrong prediction, maximal update
- If $p(y_i \mid \boldsymbol{x}_i) \to 1$, correct prediction, minimal update

Linear Regression

Hypothesis Class

- The hypothesis class of linear regression predictors is defined as

$$\mathcal{H}_{\text{reg}} = \{\boldsymbol{x} \mapsto \langle \boldsymbol{w}, \boldsymbol{x} \rangle : \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}\} \tag{34}$$

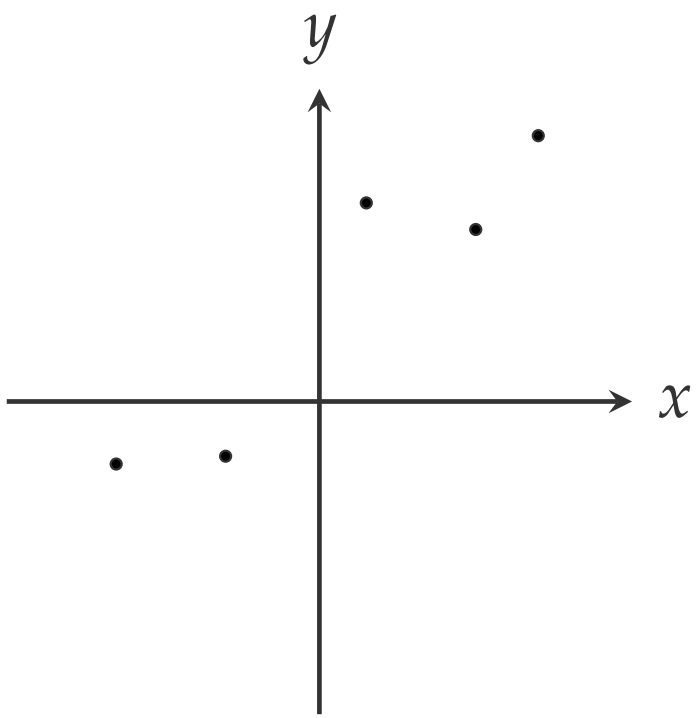
- One example hypothesis $h \in \mathcal{H}_{\text{reg}}$

$$h(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle \tag{35}$$

where $\boldsymbol{x} = (x_1, \ldots, x_n, 1)$ and $\boldsymbol{w} = (w_1, \ldots, w_n, b)$

Problem Statement

Given the training set S, in this case, $\{(x_1, y_1), \ldots, (x_5, y_5)\}$, find $h \in \mathcal{H}_{\text{reg}}$ such that $h(x)$ gives the best (linear) relation between x and y



Loss Function

- Loss function

$$L(h, (\boldsymbol{x}, y)) = (h(\boldsymbol{x}) - y)^2 = (\boldsymbol{w}^\top \boldsymbol{x} - y)^2 \tag{36}$$

- Given the training set S, the corresponding empirical risk function of linear regression is defined as

$$L(h, S) = \frac{1}{m} \sum_{i=1}^{m} (h(\boldsymbol{x}_i) - y_i)^2 \tag{37}$$

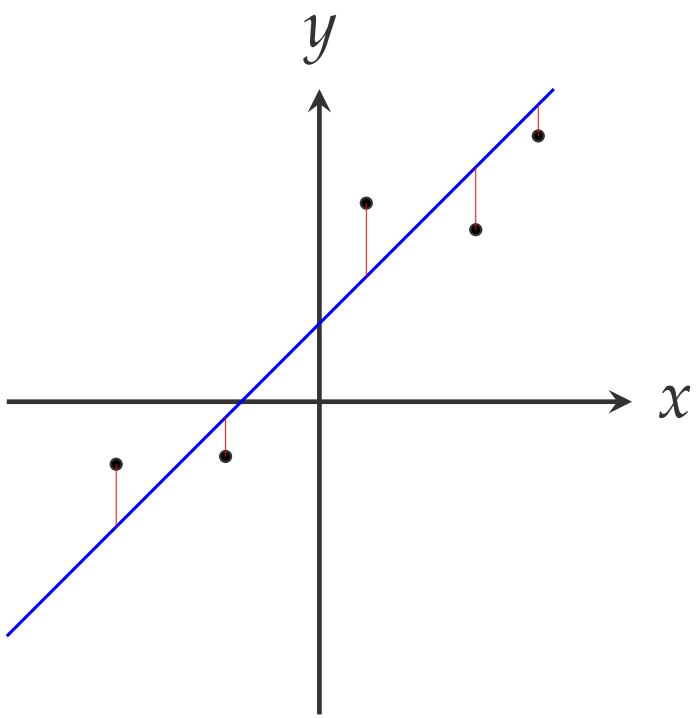
which is called *Mean Squared Error*

Visualization

For a 1-D case, the loss function

$$L(h, S) = \frac{1}{m} \sum_{i=1}^{m} (h(\boldsymbol{x}_i) - y_i)^2 \tag{38}$$

can be visualized as



Empirical Risk Minimization

- The ERM problem

$$\operatorname*{argmin}_{w} L_S(h_w) = \operatorname*{argmin}_{w} \frac{1}{m} \sum_{i=1}^{m} (\langle w, x_i \rangle - y_i)^2 \quad (39)$$

- Compute the gradient and set it to be zero

$$\frac{2}{m} \sum_{i=1}^{m} (\langle w, x_i \rangle - y_i) x_i = 0$$

$$\sum_{i=1}^{m} \langle w, x_i \rangle x_i = y_i x_i$$

Empirical Risk Minimization (II)

- Since $\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle \boldsymbol{x}_i = (\boldsymbol{w}^\top \boldsymbol{x}_i)\boldsymbol{x}_i = (\boldsymbol{x}_i \boldsymbol{x}_i^\top)\boldsymbol{w}$

$$\sum_{i=1}^{m} (\boldsymbol{x}_i \boldsymbol{x}_i^\top)\boldsymbol{w} = \sum_{i=1}^{m} y_i \boldsymbol{x}_i \tag{40}$$

- Rewrite it as $\mathbf{A}\boldsymbol{w} = \boldsymbol{b}$ with

$$\mathbf{A} = \sum_{i=1}^{m} \boldsymbol{x}_i \boldsymbol{x}_i^\top \quad \boldsymbol{b} = \sum_{i=1}^{m} y_i \boldsymbol{x}_i \tag{41}$$

Solution

- If $\mathbf{A}$ is invertible, the solution of the ERM problem is

$$w = \mathbf{A}^{-1} b \tag{42}$$

- If $\mathbf{A}$ is not invertible, consider the eigen decomposition of $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{U}^{\mathsf{T}}$, and we can compute the *generalized* inverse $\mathbf{A}^{+} = \mathbf{U}\mathbf{D}^{+}\mathbf{U}^{\mathsf{T}}$, then

$$\hat{w} = \mathbf{A}^{+} b \tag{43}$$

- With $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_i, 0, \ldots, 0)$, $\mathbf{D}^{+}$ is defined as

$$\mathbf{D}^{+} = \mathrm{diag}(\frac{1}{d_1}, \ldots, \frac{1}{d_i}, 0, \ldots, 0) \tag{44}$$

Verification of Generalized Inverse

$$\mathbf{D} = \begin{bmatrix} d_1 & & & & \\ & \ddots & & & \\ & & d_i & & \\ & & & 0 & \\ & & & & \ddots \\ & & & & & 0 \end{bmatrix} \quad \mathbf{D}^+ = \begin{bmatrix} \frac{1}{d_1} & & & & \\ & \ddots & & & \\ & & \frac{1}{d_i} & & \\ & & & 0 & \\ & & & & \ddots \\ & & & & & 0 \end{bmatrix}$$

- $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{D}^\mathsf{T}$
- $\mathbf{A}^+ = \mathbf{U}\mathbf{D}^+\mathbf{D}^\mathsf{T}$

$$\mathbf{A}\mathbf{A}^+ = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & 0 & \\ & & & & \ddots \\ & & & & & 0 \end{bmatrix} \tag{45}$$

ℓ_2 Regularization

- Another common way of addressing the non-invertible issue is to add a constraint on $\boldsymbol{w}$ as

$$L_S(h) = \frac{1}{m}\sum_{i=1}^{m}(h(\boldsymbol{x}_i) - y_i)^2 + \lambda\|\boldsymbol{w}\|^2 \tag{46}$$

  where λ is the regularization parameter
- Gradient of the new $L(h, S)$ as

$$\frac{dL_S(h)}{d\boldsymbol{w}} = \frac{2}{m}\sum_{i=1}^{m}(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle - y_i)\boldsymbol{x}_i + \lambda\boldsymbol{w} \tag{47}$$

ℓ_2 Regularization

- Solution: with the notations $\mathbf{A}$ and b defined in Eq. (41)

$$w = (\mathbf{A} + \lambda \mathbf{I})^{-1} b \tag{48}$$

- $\mathbf{A} + \lambda \mathbf{I}$ is invertible

$$\mathbf{A} + \lambda \mathbf{I} = \mathbf{U}\mathbf{D}\mathbf{U}^{\mathsf{T}} + \lambda \mathbf{I} = \mathbf{U}(\mathbf{D} + \lambda \mathbf{I})\mathbf{D}^{\mathsf{T}} \tag{49}$$

- Regularization will be further discussed in the following two topics
  - Model selection
  - Regularization and stability
- *Exercise*: Prove Eq. (48)

A Probabilistic View of Linear Regression

Consider the loss function $L_S(h)$ defined in equation 46,

$$
\begin{aligned}
\exp(-L_S(h)) &= \exp\Big\{-\frac{1}{m}\sum_{i=1}^{m}(h(\boldsymbol{x}_i)-y_i)^2-\lambda\|\boldsymbol{w}\|^2\Big\} \\
&\propto \exp\Big\{-\sum_{i=1}^{m}(h(\boldsymbol{x}_i)-y_i)^2\Big\}\cdot\exp\Big\{-\|\boldsymbol{w}\|^2\Big\} \\
&= \prod_{i=1}^{m}\exp\Big\{-(h(\boldsymbol{x}_i)-y_i)^2\Big\}\cdot\exp\Big\{-\|\boldsymbol{w}\|^2\Big\} \\
&\propto \prod_{i=1}^{m}\mathcal{N}(y_i \mid h(\boldsymbol{x}_i),\frac{1}{2})\cdot\mathcal{N}(\boldsymbol{w}\mid 0,\frac{1}{2})
\end{aligned}
$$

A Probabilistic View of Linear Regression (II)

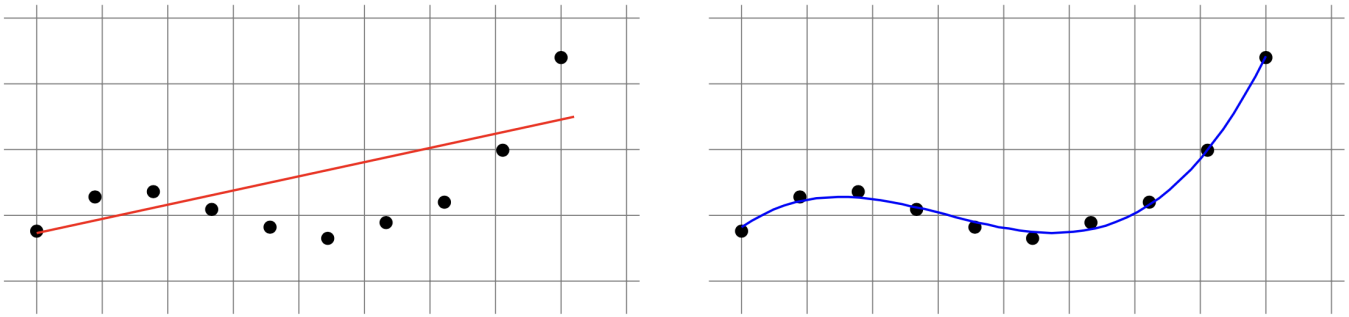
Minimize the loss function $L_S(h)$ is equivalent to maximizing the following objective function

$$\exp(-L_S(h)) \propto \prod_{i=1}^{m} \mathcal{N}(y_i \mid h(\boldsymbol{x}_i), \frac{1}{2}) \cdot \mathcal{N}(\boldsymbol{w} \mid 0, \frac{1}{2}) \qquad (50)$$

- $\prod_{i=1}^{m} \mathcal{N}(y_i \mid h(\boldsymbol{x}_i), \frac{1}{2})$: likelihood function of the data S
- $\mathcal{N}(\boldsymbol{w} \mid 0, \frac{1}{2})$: prior distribution of $\boldsymbol{w}$

Polynomial Regression

Some learning tasks require nonlinear predictors with single variable $x \in \mathbb{R}$



$$f_w(x) = w_0 + w_1 x + \cdots + w_n x^n \tag{51}$$

where $\boldsymbol{w} = (w_0, w_1, \ldots, w_n)$ is a vector of coefficients of size $n + 1$.

Reference

Berger, J. O. and Wolpert, R. L. (1988).
The likelihood principle.
IMS.

Bishop, C. M. (2006).
Pattern recognition and machine learning.
springer.

Friedman, J., Hastie, T., and Tibshirani, R. (2001).
The elements of statistical learning.
Springer.