CS 6316 Machine Learning

Linear Predictors

Yangfeng Ji

Information and Language Processing Lab
Department of Computer Science
University of Virginia

Overview

1. Review: Linear Functions
2. Perceptron
3. Logistic Regression
4. Linear Regression
5. ℓ_2 Regularization and Overfitting
6. Summary

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

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

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

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

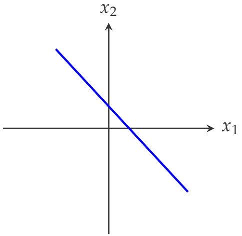
and then

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

Linear Functions

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

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



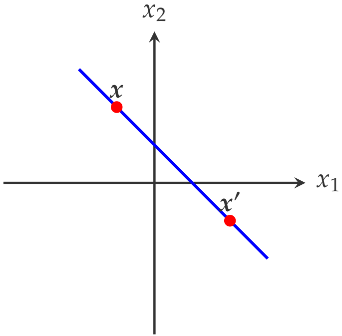
Different values of $f(\boldsymbol{x})$ map to different areas on this 2-D space. For example, the following equation defines the blue line L.

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

Properties of Linear Functions (II)

For any two points $\boldsymbol{x}$ and $\boldsymbol{x}'$ lying in the line

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



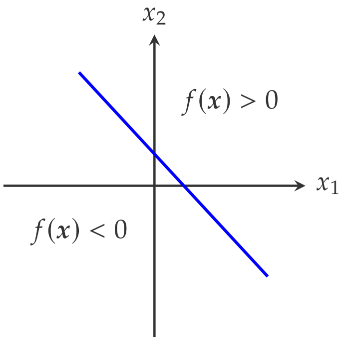
[Friedman et al., 2001, Section 4.5]

Properties of Linear Functions (III)

Furthermore,

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

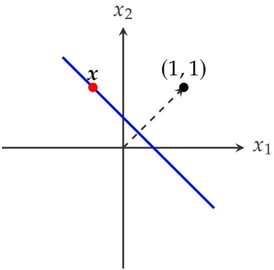
separates the 2-D space $\mathbb{R}^2$ into two half spaces



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

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

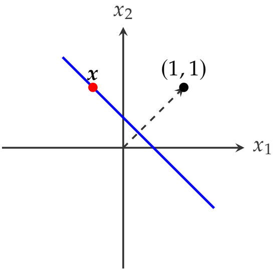


[Friedman et al., 2001, Section 4.5]

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

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



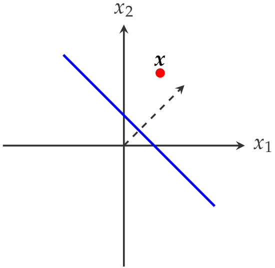
This idea can be generalized to compute the distance between a point and a line.

[Friedman et al., 2001, Section 4.5]

Properties of Linear Functions (IV)

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

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



[Friedman et al., 2001, Section 4.5]

Perceptron

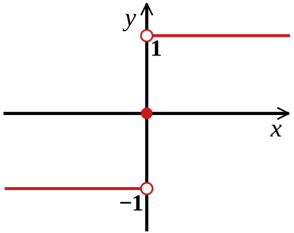
Halfspace Hypothesis Class

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

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

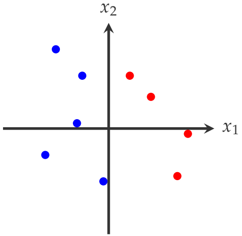
  which is an infinite hypothesis space.

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

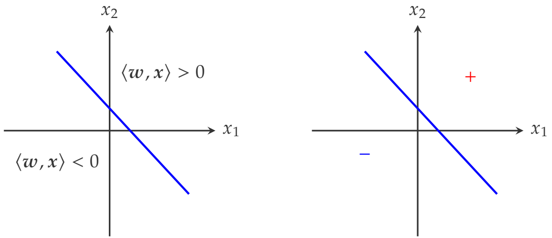


The definition of linearly separable cases is with respect to the training set S instead of $\mathscr{D}$

Prediction Rule

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

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



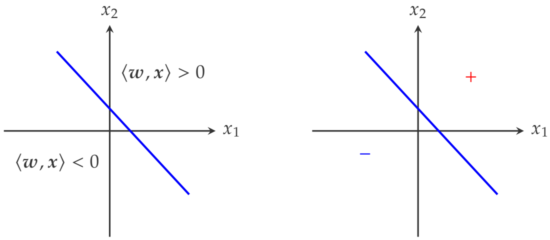
Prediction Rule

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

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

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

Perceptron Algorithm

The perceptron algorithm is defined as

```
Input: S = {(x_1, y_1), ..., (x_m, y_m))}
Initialize w^(0) = (0, ..., 0)
for t = 1, 2, ..., T do
    i ← t mod m



end for
Output: w^(T)
```

Perceptron Algorithm

The perceptron algorithm is defined as

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

Perceptron Algorithm

The perceptron algorithm is defined as

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

Exercise: Implementing this algorithm with a simple example

Two Questions

The updating rule can be break down into two cases:

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

- 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:

$$\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} + y_i \boldsymbol{x}_i \tag{13}$$

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

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 $(\boldsymbol{x}_i, y_i)$ and the current weight $\boldsymbol{w}^{(t)}$

$$\begin{aligned} y_i\langle \boldsymbol{w}^{(t+1)}, \boldsymbol{x}_i\rangle &= y_i\langle \boldsymbol{w}^{(t)} + y_i\boldsymbol{x}_i, \boldsymbol{x}_i\rangle && (14) \\ &= y_i\langle \boldsymbol{w}^{(t)}, \boldsymbol{x}_i\rangle + \|\boldsymbol{x}_i\|^2 && (15) \end{aligned}$$

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

Theorem

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

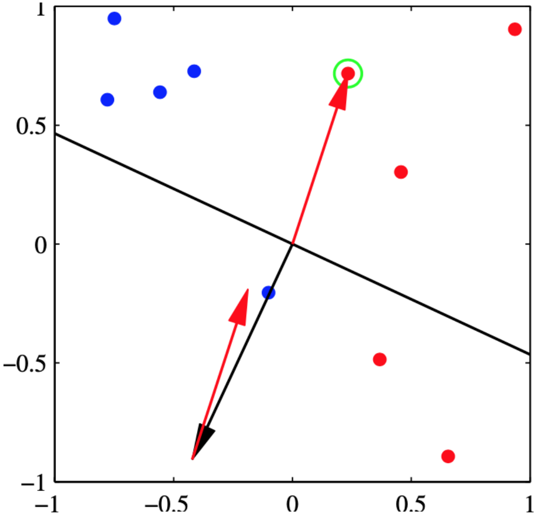
- $B = \min\{\|\boldsymbol{w}\| : \forall i \in [m], y_i\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle \geq 1\}$, and
- $R = \max_i \|\boldsymbol{x}_i\|$.

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

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

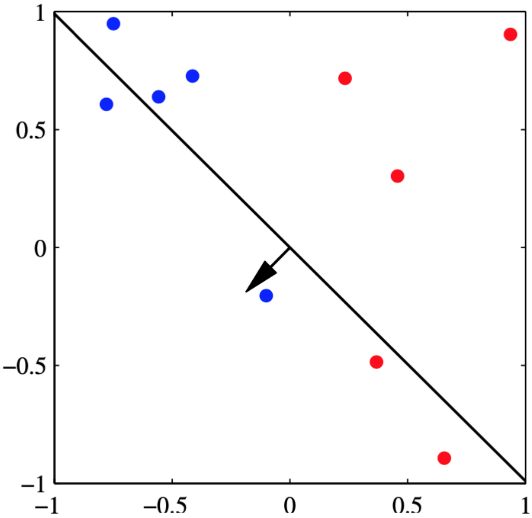
- A realizable case with infinite hypothesis space
- Finish training in finite steps

Example



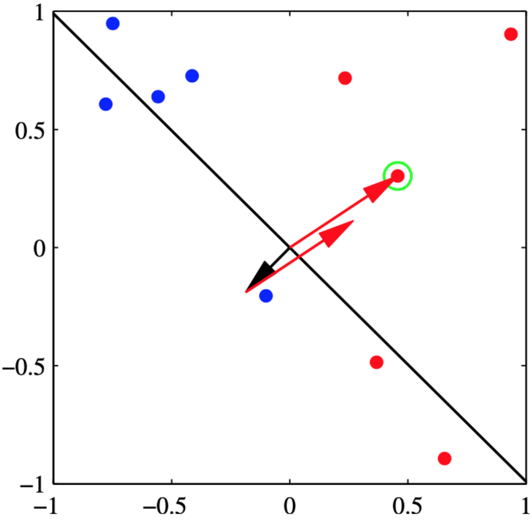
[Bishop, 2006, Page 195]

Example



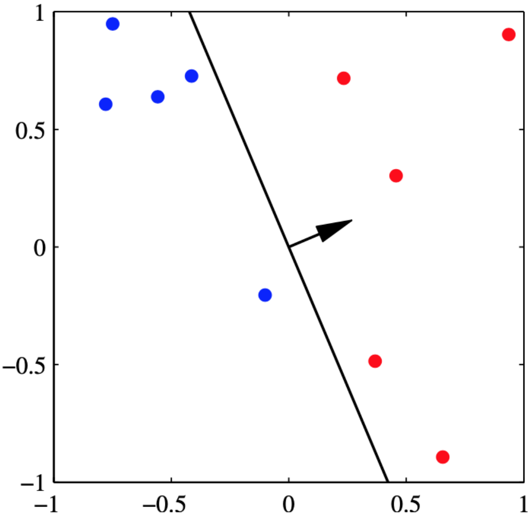
[Bishop, 2006, Page 195]

Example



[Bishop, 2006, Page 195]

Example



[Bishop, 2006, Page 195]

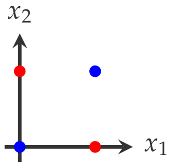
The XOR Example: a Non-separable Case

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

$$Y = X_1 \oplus X_2$$

where

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



Logistic Regression

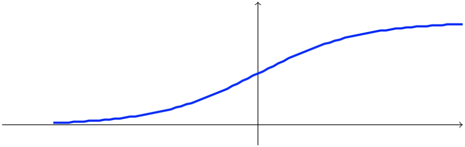
Hypothesis Class

- 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 $y \in \{-1, +1\}$

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

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 $h(\boldsymbol{x}, y)$

- When $y = +1$

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

- When $y = -1$

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

A Linear Classifier?

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

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

Specifically, we have

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

The decision boundary is a straight line

Risk/Loss Function

For a given training example $(\boldsymbol{x}, y)$, the risk/loss function is defined as the negative log of $h(\boldsymbol{x}, y)$

$$L(h_w, (\boldsymbol{x}, y)) = -\log \frac{1}{1 + \exp(-y\langle \boldsymbol{w}, \boldsymbol{x} \rangle)} = \log(1 + \exp(-y\langle \boldsymbol{w}, \boldsymbol{x} \rangle)) \quad (22)$$

Intuitively, minimizing the risk will increase the value of $h(\boldsymbol{x}, y)$

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 $\boldsymbol{w}$

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

- $L(h_{\boldsymbol{w}}, S)$ is convex function with respect to $\boldsymbol{w}$
- Estimation of $\boldsymbol{w}$: $\hat{\boldsymbol{w}} \leftarrow \text{argmin}_{\boldsymbol{w}'} L(h_{\boldsymbol{w}'}, S)$
- 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 $\boldsymbol{w}$

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

- *Exercise*: prove Eq. 24

Gradient Descent

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

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

- Gradient-based learning

$$\boldsymbol{w}^{(\text{new})} = \boldsymbol{w}^{(\text{old})} - \eta \frac{dL(h_{\boldsymbol{w}}, S)}{d\boldsymbol{w}}$$

where η is the updating step size.

- *Exercise*: prove Eq. 24

Gradient Descent

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

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

- 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

$$w^{(\text{new})} = w^{(\text{old})} + \frac{\eta}{m} \sum_{i=1}^{m} \underbrace{\frac{\exp(-y_i \langle w, x_i \rangle)}{1 + \exp(-y_i \langle w, x_i \rangle)}}_{(2)} \cdot \underbrace{(y_i x_i)}_{(1)} \tag{25}$$

For each (x_i, y_i), the update is

(1) directed by the true label y_i, as in the Perceptron algorithm

(2) 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

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

Updating Rules

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

- Logistic regression

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

- Perceptron algorithm

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

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 $\boldsymbol{x}$

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

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

A Probabilistic View of Logistic Regression

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

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

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

- The previous prediction rule is equivalent to

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

Parameter Estimation: Likelihood Function

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

$$\mathrm{Lik}(\boldsymbol{x}) = \prod_{i=1}^{m} p_w(y_i \mid \boldsymbol{x}_i) \quad (30)$$

Parameter Estimation: 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]

Parameter Estimation: Maximum Likelihood

Given the training set S,

- Log-likelihood function

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

Parameter Estimation: Maximum Likelihood

Given the training set S,

- Log-likelihood function

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

- Maximize the log-likelihood function

$$\operatorname{argmax}_{w} \ell(\boldsymbol{w}) = \operatorname{argmin}_{w} -\ell(\boldsymbol{w}) = \operatorname{argmin}_{w} L(h_{\boldsymbol{w}}, S)$$

learning with ERM is equivalent to the Maximum Likelihood Estimation (MLE) in Statistics

Gradient Descent, revisited

Recall the gradient-based learning on the previous slide

$$
\begin{aligned}
\boldsymbol{w}^{(\text{new})} &= \boldsymbol{w}^{(\text{old})} + \eta \sum_{i=1}^{m} \frac{\exp(-y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)}{1 + \exp(-y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)} \cdot (y_i \boldsymbol{x}_i) \\
&= \boldsymbol{w}^{(\text{old})} + \eta \sum_{i=1}^{m} (1 - p(y_i \mid \boldsymbol{x}_i)) \cdot y_i \boldsymbol{x}_i \qquad (32)
\end{aligned}
$$

- 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

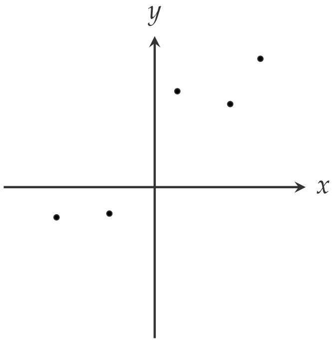
$$\mathcal{H}_{\text{reg}} = \{\langle \boldsymbol{w}, \boldsymbol{x} \rangle : \boldsymbol{w} \in \mathbb{R}^d\} \tag{33}$$

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

$$h(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle \tag{34}$$

Problem Statement

Given the training set S, in this case, $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_5, y_5)\}$, find $h \in \mathcal{H}_{\text{reg}}$ such that $h(\boldsymbol{x})$ gives the best (linear) relation between x and y



Loss Function

- Loss function

$$L(h, (\boldsymbol{x}, y)) = (h(\boldsymbol{x}) - y)^2 = (\boldsymbol{w}^\top \boldsymbol{x} - y)^2 \tag{35}$$

Loss Function

- Loss function

$$L(h, (\boldsymbol{x}, y)) = (h(\boldsymbol{x}) - y)^2 = (\boldsymbol{w}^\top \boldsymbol{x} - y)^2 \tag{35}$$

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

$$L(h, S) = \frac{1}{m} \sum_{i=1}^{m} (h(\boldsymbol{x}_i) - y_i)^2 \tag{36}$$

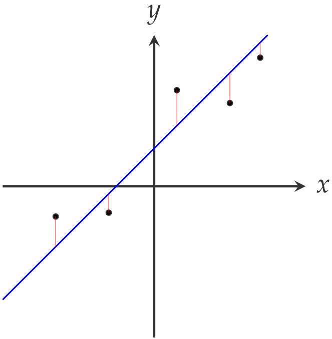
which is called **Mean Squared Error** (MSE).

Visualization

For a 1-D case, the loss function

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

can be visualized as



Empirical Risk Minimization

- The ERM problem

$$\underset{w}{\operatorname{argmin}} L_S(h_w) = \underset{w}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - y_i)^2 \tag{38}$$

- Compute the gradient and set it to be zero

$$\begin{aligned} \frac{2}{m} \sum_{i=1}^{m} (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle &- y_i)\boldsymbol{x}_i = 0 \\ \sum_{i=1}^{m} \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle \boldsymbol{x}_i &= y_i \boldsymbol{x}_i \end{aligned}$$

Empirical Risk Minimization (II)

To isolate $\boldsymbol{w}$ for solution, we have

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

$$\sum_{i=1}^{m} (\boldsymbol{x}_i \boldsymbol{x}_i^\top)\boldsymbol{w} = \sum_{i=1}^{m} y_i \boldsymbol{x}_i \tag{39}$$

- then, rewrite it as

$$\mathbf{A}\boldsymbol{w} = \boldsymbol{b} \tag{40}$$

with

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

Review: Symmetric Matrices

A symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is defined as

$$\mathbf{A}^{\top} = \mathbf{A} \tag{42}$$

or, in other words,

$$a_{i,j} = a_{j,i} \quad \forall i, j \in [n] \tag{43}$$

Comments

- The identity matrix $\mathbf{I}$ is symmetric
- A diagonal matrix is symmetric

Review: Eigen Decomposition

Every symmetric matrix $\mathbf{A}$ can be decomposed as

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\top} \tag{44}$$

with

- $\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}$ as a diagonal matrix
- $\mathbf{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n]$ is an orthogonal matrix

$$\langle \boldsymbol{u}_i, \boldsymbol{u}_i \rangle = \|\boldsymbol{u}\|_2^2 = 1 \text{ and } \langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle = 0$$

Inverse

The *inverse* of a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is denoted as $\mathbf{A}^{-1}$, which is the unique matrix such that

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{I} = \mathbf{A}\mathbf{A}^{-1} \tag{45}$$

- Not all matrices are invertible
  - Non-square matrices do not have inverses (by definition)
  - Not all square matrices are invertible
    - Not all symmetric matrices are invertible

Solution

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

$$\boldsymbol{w} = \mathbf{A}^{-1}\boldsymbol{b} \tag{46}$$

Solution

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

$$\boldsymbol{w} = \mathbf{A}^{-1}\boldsymbol{b} \tag{46}$$

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

$$\hat{\boldsymbol{w}} = \mathbf{A}^{+}\boldsymbol{b} \tag{47}$$

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

$$\mathbf{D}^{+} = \text{diag}(\frac{1}{d_1}, \ldots, \frac{1}{d_i}, 0, \ldots, 0) \tag{48}$$

Verification of Generalized Inverse

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

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

$$\mathbf{A}\mathbf{A}^+ = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} \tag{49}$$

ℓ_2 Regularization

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

$$L_{S,\ell_2}(h_{\boldsymbol{w}}) = \frac{1}{m}\sum_{i=1}^{m}(h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2 + \lambda\|\boldsymbol{w}\|^2 \qquad (50)$$

  where λ is the regularization parameter
- Gradient of the new $L_S(h_{\boldsymbol{w}})$ as

$$\frac{dL_{S,\ell_2}(h_{\boldsymbol{w}})}{d\boldsymbol{w}} = \frac{2}{m}\sum_{i=1}^{m}(\langle\boldsymbol{w},\boldsymbol{x}_i\rangle - y_i)\boldsymbol{x}_i + 2\lambda\boldsymbol{w} \qquad (51)$$

ℓ_2 Regularization

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

$$\boldsymbol{w} = (\mathbf{A} + \lambda\mathbf{I})^{-1}b \tag{52}$$

- *Exercise*: Prove Eq. (52)

ℓ_2 Regularization

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

$$\boldsymbol{w} = (\mathbf{A} + \lambda\mathbf{I})^{-1} b \tag{52}$$

- $\mathbf{A} + \lambda\mathbf{I}$ is invertible, when $d_i + \lambda \neq 0, \forall i$

$$\mathbf{A} + \lambda\mathbf{I} = \mathbf{U}\mathbf{D}\mathbf{U}^{\top} + \lambda\mathbf{I} = \mathbf{U}(\mathbf{D} + \lambda\mathbf{I})\mathbf{U}^{\top} \tag{53}$$

- *Exercise*: Prove Eq. (52)

ℓ_2 Regularization

- Solution: with the notations **A** and b defined in Eq. (41)

$$\boldsymbol{w} = (\mathbf{A} + \lambda\mathbf{I})^{-1} b \tag{52}$$

- $\mathbf{A} + \lambda\mathbf{I}$ is invertible, when $d_i + \lambda \neq 0, \forall i$

$$\mathbf{A} + \lambda\mathbf{I} = \mathbf{U}\mathbf{D}\mathbf{U}^\top + \lambda\mathbf{I} = \mathbf{U}(\mathbf{D} + \lambda\mathbf{I})\mathbf{U}^\top \tag{53}$$

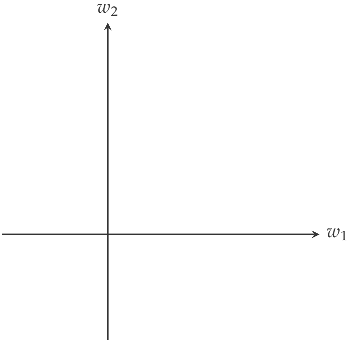
- Regularization will be further discussed in the next lecture on model selection
- *Exercise*: Prove Eq. (52)

ℓ_2 Regularization: Illustration

Consider a 2-D case, where $\boldsymbol{x} = (x_1, x_2)$ and $\boldsymbol{w} = (w_1, w_2)$

$$L_{S,\ell_2}(h_{\boldsymbol{w}}) = \frac{1}{m}\sum_{i=1}^{m}(h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2 + \lambda\|\boldsymbol{w}\|^2 \tag{54}$$

Visualization of both components with their contour plots

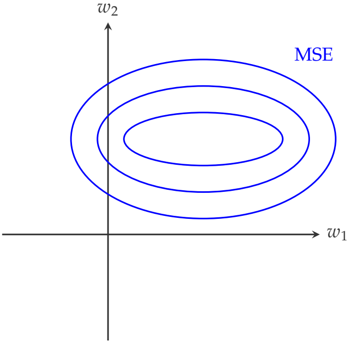


ℓ_2 Regularization: Illustration

Consider a 2-D case, where $\boldsymbol{x} = (x_1, x_2)$ and $\boldsymbol{w} = (w_1, w_2)$

$$L_{S,\ell_2}(h_{\boldsymbol{w}}) = \frac{1}{m}\sum_{i=1}^{m}(h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2 + \lambda\|\boldsymbol{w}\|^2 \quad (54)$$

Visualization of both components with their contour plots

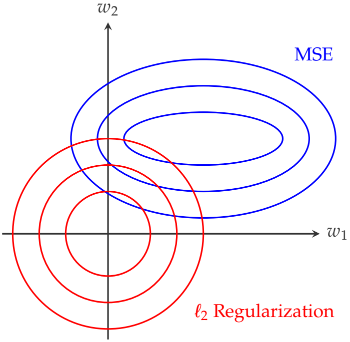


ℓ_2 Regularization: Illustration

Consider a 2-D case, where $\boldsymbol{x} = (x_1, x_2)$ and $\boldsymbol{w} = (w_1, w_2)$

$$L_{S,\ell_2}(h_{\boldsymbol{w}}) = \frac{1}{m}\sum_{i=1}^{m}(h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2 + \lambda\|\boldsymbol{w}\|^2 \qquad (54)$$

Visualization of both components with their contour plots

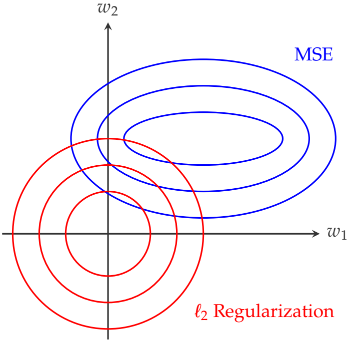


ℓ_2 Regularization: Illustration

Consider a 2-D case, where $\boldsymbol{x} = (x_1, x_2)$ and $\boldsymbol{w} = (w_1, w_2)$

$$L_{S,\ell_2}(h_{\boldsymbol{w}}) = \frac{1}{m}\sum_{i=1}^{m}(h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2 + \lambda\|\boldsymbol{w}\|^2 \tag{54}$$

Visualization of both components with their contour plots



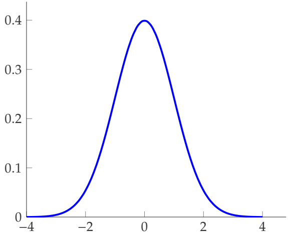
Minimizing $L_{S,\ell_2}(h_{\boldsymbol{w}})$ is to find a tradeoff between these two components

Gaussian Distribution

A random variable $X \in \mathbb{R}$ is said to follow a normal (or Gaussian) distribution $\mathcal{N}(\mu, \sigma^2)$ if its probability density function is given by

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{55}$$

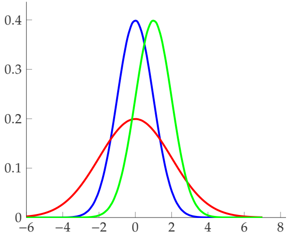
- μ: mean
- σ^2: variance
- Probability of $X \in [a, b]$: $P(a \leq X \leq b) = \int_a^b f(x)dx$



Gaussian Distribution (II)

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(- \frac{(x-\mu)^2}{2\sigma^2} \right) \tag{56}$$

There examples of Gaussian distributions



- Blue: $\mathcal{N}(0, 1)$ (standard normal distribution)
- Red: $\mathcal{N}(0, 2)$
- Green: $\mathcal{N}(1, 1)$

A Probabilistic View of Linear Regression

Consider the loss function $L_{S,\ell_2}(h_{\boldsymbol{w}})$ defined in equation 50,

$$\exp(-L_{S,\ell_2}(h_{\boldsymbol{w}})) \quad = \quad \exp\Big\{-\frac{1}{m}\sum_{i=1}^{m}(h_{\boldsymbol{w}}(\boldsymbol{x}_i)-y_i)^2-\lambda\|\boldsymbol{w}\|^2\Big\}$$

A Probabilistic View of Linear Regression

Consider the loss function $L_{S,\ell_2}(h_w)$ defined in equation 50,

$$\begin{aligned}\exp(-L_{S,\ell_2}(h_w)) &= \exp\left\{-\frac{1}{m}\sum_{i=1}^{m}(h_w(\boldsymbol{x}_i)-y_i)^2-\lambda\|\boldsymbol{w}\|^2\right\}\\ &\propto \exp\left\{-\sum_{i=1}^{m}(h_w(\boldsymbol{x}_i)-y_i)^2\right\}\cdot\exp\left\{-\lambda\|\boldsymbol{w}\|^2\right\}\end{aligned}$$

A Probabilistic View of Linear Regression

Consider the loss function $L_{S,\ell_2}(h_{\boldsymbol{w}})$ defined in equation 50,

$$\begin{aligned}
\exp(-L_{S,\ell_2}(h_{\boldsymbol{w}})) &= \exp\Big\{-\frac{1}{m}\sum_{i=1}^{m}(h_{\boldsymbol{w}}(\boldsymbol{x}_i)-y_i)^2-\lambda\|\boldsymbol{w}\|^2\Big\} \\
&\propto \exp\Big\{-\sum_{i=1}^{m}(h_{\boldsymbol{w}}(\boldsymbol{x}_i)-y_i)^2\Big\}\cdot\exp\Big\{-\lambda\|\boldsymbol{w}\|^2\Big\} \\
&= \prod_{i=1}^{m}\exp\Big\{-(h_{\boldsymbol{w}}(\boldsymbol{x}_i)-y_i)^2\Big\}\cdot\exp\Big\{-\lambda\|\boldsymbol{w}\|^2\Big\}
\end{aligned}$$

A Probabilistic View of Linear Regression

Consider the loss function $L_{S,\ell_2}(h_w)$ defined in equation 50,

$$
\begin{aligned}
\exp(-L_{S,\ell_2}(h_w)) &= \exp\Big\{-\frac{1}{m}\sum_{i=1}^{m}(h_w(\boldsymbol{x}_i)-y_i)^2-\lambda\|\boldsymbol{w}\|^2\Big\} \\
&\propto \exp\Big\{-\sum_{i=1}^{m}(h_w(\boldsymbol{x}_i)-y_i)^2\Big\}\cdot\exp\Big\{-\lambda\|\boldsymbol{w}\|^2\Big\} \\
&= \prod_{i=1}^{m}\exp\Big\{-(h_w(\boldsymbol{x}_i)-y_i)^2\Big\}\cdot\exp\Big\{-\lambda\|\boldsymbol{w}\|^2\Big\} \\
&\propto \prod_{i=1}^{m}\mathcal{N}(y_i \mid h_w(\boldsymbol{x}_i),\frac{1}{2})\cdot\mathcal{N}(\boldsymbol{w}\mid 0,\frac{1}{2\lambda})
\end{aligned}
$$

A Probabilistic View of Linear Regression (II)

Minimize the loss function $L_{S,\ell_2}(h_{\boldsymbol{w}})$ is equivalent to maximizing the following objective function

$$\exp(-L_S(h_{\boldsymbol{w}})) \propto \prod_{i=1}^{m} \mathcal{N}(y_i \mid h_{\boldsymbol{w}}(\boldsymbol{x}_i), \frac{1}{2}) \cdot \mathcal{N}(\boldsymbol{w} \mid 0, \frac{1}{2\lambda}) \tag{57}$$

- $\prod_{i=1}^{m} \mathcal{N}(y_i \mid h_{\boldsymbol{w}}(\boldsymbol{x}_i), \frac{1}{2})$: likelihood function $\prod_{i=1}^{m} p(y_i \mid \boldsymbol{x}_i; \boldsymbol{w})$
- $\mathcal{N}(\boldsymbol{w} \mid 0, \frac{1}{2\lambda})$: prior distribution $p(\boldsymbol{w})$

A Probabilistic View of Linear Regression (II)

Minimize the loss function $L_{S,\ell_2}(h_w)$ is equivalent to maximizing the following objective function

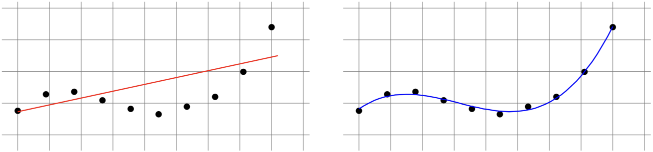
$$\exp(-L_S(h_w)) \propto \prod_{i=1}^{m} \mathcal{N}(y_i \mid h_w(x_i), \frac{1}{2}) \cdot \mathcal{N}(w \mid 0, \frac{1}{2\lambda}) \qquad (57)$$

- $\prod_{i=1}^{m} \mathcal{N}(y_i \mid h_w(x_i), \frac{1}{2})$: likelihood function $\prod_{i=1}^{m} p(y_i \mid x_i; w)$
- $\mathcal{N}(w \mid 0, \frac{1}{2\lambda})$: prior distribution $p(w)$
- Maximizing equation 57 is equivalent to the *maximum a posteriori estimation*

$$p(w \mid \{(x_i, y_i)\}_{i=1}^{m}) = \frac{p(w) \prod_{i=1}^{m} p(y_i \mid x_i; w)}{\prod_{i=1}^{m} p(y_i \mid x_i)} \qquad (58)$$

Polynomial Regression

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



$$h_{\boldsymbol{w}}(x) = w_0 + w_1 x + \cdots + w_n x^n \tag{59}$$

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

Polynomial Regression (II)

Given training examples $\{(x_i, y_i\}_{i=1}^m$, the problem of polynomial regression

$$h_{\boldsymbol{w}}(x) = w_0 + w_1 x + \cdots + w_n x^n \tag{60}$$

can be converted to a linear regression problem

$$\begin{bmatrix} 1 & x_1 & \cdots & x_1^n \\ 1 & x_2 & \cdots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & \cdots & x_m^n \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \tag{61}$$

Polynomial Regression (II)

Given training examples $\{(x_i, y_i\}_{i=1}^m$, the problem of polynomial regression

$$h_w(x) = w_0 + w_1 x + \cdots + w_n x^n \tag{60}$$

can be converted to a linear regression problem

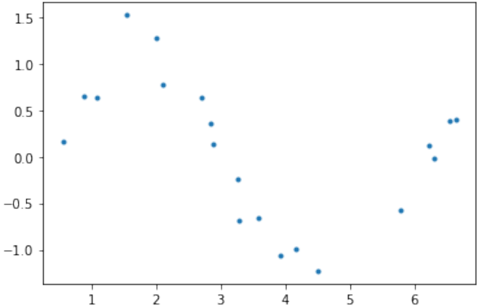
$$\begin{bmatrix} 1 & x_1 & \cdots & x_1^n \\ 1 & x_2 & \cdots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & \cdots & x_m^n \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \tag{61}$$

We will use polynomial regression as an example in the next lecture

ℓ_2 Regularization and Overfitting

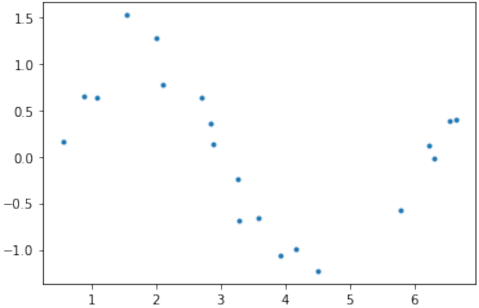
Polynomial Regression: Data

Consider the following polynomial regression problem



Polynomial Regression: Data

Consider the following polynomial regression problem



Data generation process

$$y = \sin(x) + 0.3 * \epsilon \tag{62}$$

where $\epsilon \sim \mathcal{N}(0, 1)$

Regression: Model

- We choose the hypothesis class of polynomial functions with degree 7

$$h_{\boldsymbol{w}}(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_7 x^7 \tag{63}$$

where $\{w_0, w_1, w_3, \ldots, w_7\}$ are the parameters

Regression: Model

- We choose the hypothesis class of polynomial functions with degree 7

$$h_{\boldsymbol{w}}(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_7 x^7 \tag{63}$$

where $\{w_0, w_1, w_3, \ldots, w_7\}$ are the parameters

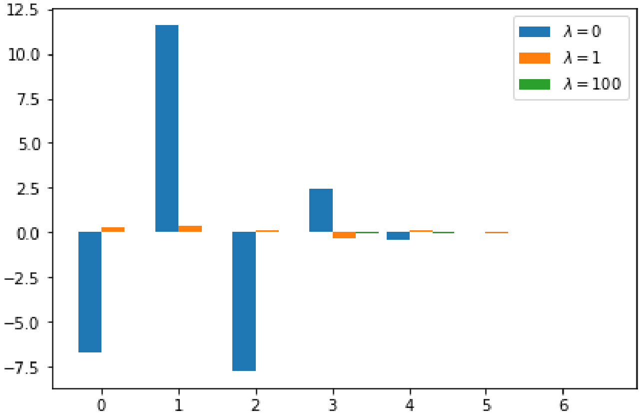
- The loss function: MSE with ℓ_2 regularization

$$L_{S,\ell_2}(h_{\boldsymbol{w}}) = \frac{1}{m}\sum_{i=1}^{m}(h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2 + \lambda\|\boldsymbol{w}\|^2 \tag{64}$$

where we can pick different values of $\lambda \in \{0, 1, 100\}$

Regression: Regularization Effects

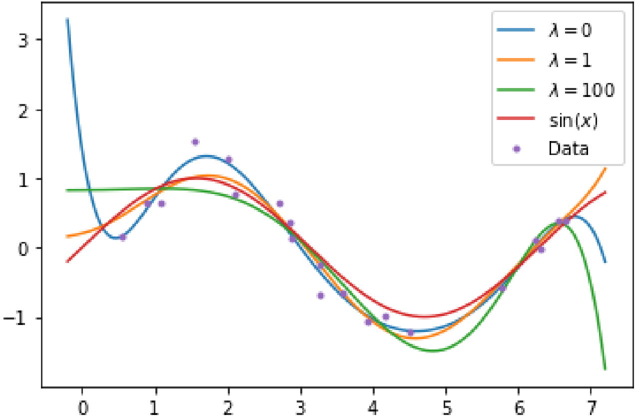
The direct effect of regularization is to constrain the coefficient to be close to zero



Larger regularization parameter, stronger effect

Regression: Regularization for Avoiding Overfitting

By forcing the coefficient to be smaller, regularization can help avoid overfitting

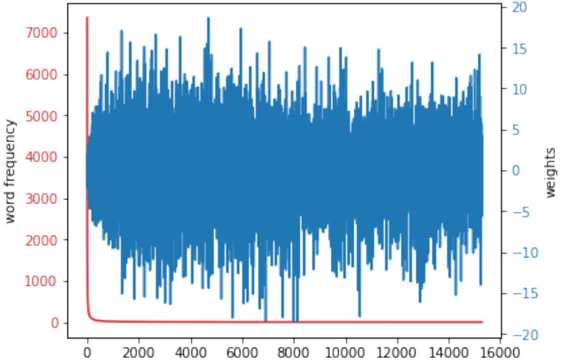


Strong regularization effect will hurt the model performance.

Regression: Learning without Regularization

In the demo code, we chose $\lambda = \frac{1}{C} = 0.001$ to approximate the case without regularization.

- Training accuracy: 99.89%
- Val accuracy: 52.21%



Classification: Weights without Regularization

Here are some word features and their classification weights from the previous model without regularization. Positive weights indicate the word feature contribute to positive sentiment classification and negative weights indicate the opposite contribution

	`interesting`	`pleasure`	`boring`	`zoe`	`write`	`workings`
Without Reg	0.011	-5.63	1.80	-5.68	-8.20	14.16

Classification: Weights without Regularization

Here are some word features and their classification weights from the previous model without regularization. Positive weights indicate the word feature contribute to positive sentiment classification and negative weights indicate the opposite contribution

	`interesting`	`pleasure`	`boring`	`zoe`	`write`	`workings`
Without Reg	0.011	-5.63	1.80	-5.68	-8.20	14.16

- NEGATIVE: `woody allen can write and deliver a one liner as well as anybody .`

Classification: Weights without Regularization

Here are some word features and their classification weights from the previous model without regularization. Positive weights indicate the word feature contribute to positive sentiment classification and negative weights indicate the opposite contribution

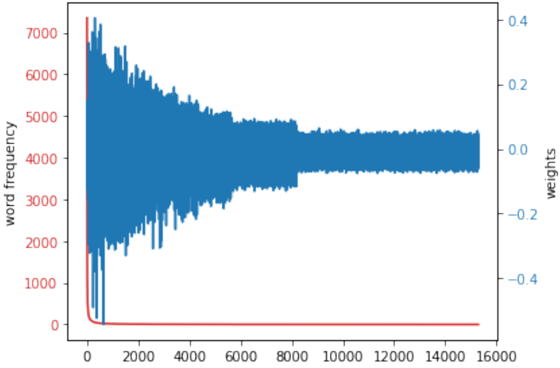
	`interesting`	`pleasure`	`boring`	`zoe`	`write`	`workings`
Without Reg	0.011	-5.63	1.80	-5.68	-8.20	14.16

- NEGATIVE: `woody allen can write and deliver a one liner as well as anybody .`
- POSITIVE: `soderbergh , like kubrick before him , may not touch the planet 's skin , but understands the workings of its spirit .`

Classification: Learning with Regularization

We chose $\lambda = \frac{1}{C} = 10^2$

- Training accuracy: 62.54%
- Val accuracy: 63.17%



Classification: Weights with Regularization

With regularization, the classification weights make more sense to us

	interesting	pleasure	boring	zoe	write	workings
Without Reg	0.011	-5.63	1.80	-5.68	-8.20	14.16
With Reg	0.16	0.36	-0.21	-0.057	-0.066	0.040

Summary

Important Concepts

- Perceptron
  - The hypothesis class (page 11)
  - Linearly separable cases (page 12)
  - Perceptron updating rule (page 14)
- Logistic regression
  - The hypothesis class (page 21 - 22)
  - Gradient-based updating rule (page 27 - 29)
  - Maximum likelihood estimation (page 32)
- Linear regression
  - The hypothesis class (page 35)
  - ℓ_2 regularization (page 46)
  - Maximum a posteriori (MAP) estimation (page 52)
- ℓ_2 Regularization and Overfitting

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.