EE2211 Introduction to Machine Learning

Lecture 6

Semester 2
2023/2024

Yueming Jin
ymjin@nus.edu.sg

Electrical and Computer Engineering Department
National University of Singapore

Acknowledgement:
EE2211 development team
(Xinchao, Helen, Thomas, Kar-Ann, Chen Khong, Robby and Haizhou

Course Contents

- Introduction and Preliminaries (Xinchao)
  - Introduction
  - Data Engineering
  - Introduction to Probability and Statistics
- Fundamental Machine Learning Algorithms I (Yueming)
  - Systems of linear equations
  - Least squares, Linear regression
  - Ridge regression, Polynomial regression
- Fundamental Machine Learning Algorithms II (Yueming)
  - Over-fitting, bias/variance trade-off
  - Optimization, Gradient descent
  - Decision Trees, Random Forest
- Performance and More Algorithms (Xinchao)
  - Performance Issues
  - K-means Clustering
  - Neural Networks

Mid-term: Lecture 1 to 5
Trial quiz
Assignment 1 & 2

Ridge Regression & Polynomial Regression

Module II Contents

- Notations, Vectors, Matrices
- Operations on Vectors and Matrices
- Systems of Linear Equations
- Functions, Derivative and Gradient
- Least Squares, Linear Regression
- Linear Regression with Multiple Outputs
- Linear Regression for Classification
- Ridge Regression
- Polynomial Regression

Review: Linear Regression
Learning of Scalar Function (Single Output)

For one sample: a linear model $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$ scalar function

For *m* samples: $\boldsymbol{f}_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w} = \mathbf{y}$

$$\mathbf{y} = \begin{bmatrix} \mathbf{x}_1^T\mathbf{w} \\ \vdots \\ \mathbf{x}_m^T\mathbf{w} \end{bmatrix} \quad \text{where} \quad \mathbf{x}_i^T = [1, x_{i,1}, \ldots, x_{i,d}]$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & \ldots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1} & \ldots & x_{m,d} \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

Objective: $\sum_{i=1}^{m}(f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \mathbf{e}^T\mathbf{e} = (\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y})$

Learning/training when $\mathbf{X}^T\mathbf{X}$ is invertible

Least square solution: $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

Prediction/testing: $y_{new} = \hat{\boldsymbol{f}}_{\mathbf{w}}(\mathbf{X}_{new}) = \mathbf{X}_{new}\hat{\mathbf{w}}$

Review: Linear Regression
Learning of Vectored Function (Multiple Outputs)

$$\mathbf{F_w}(\mathbf{X}) = \mathbf{X}\mathbf{W} = \mathbf{Y}$$

Sample 1 ⋮ Sample m

$$= \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_m^T \end{bmatrix} \mathbf{W} = \begin{bmatrix} 1 & x_{1,1} & \dots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1} & \dots & x_{m,d} \end{bmatrix} \begin{bmatrix} w_{0,1} & \dots & w_{0,h} \\ w_{1,1} & \dots & w_{1,h} \\ \vdots & \ddots & \vdots \\ w_{d,1} & \dots & w_{d,h} \end{bmatrix}$$

Sample 1's output ⋮ Sample m's output

$$= \begin{bmatrix} y_{1,1} & \dots & y_{1,h} \\ & \vdots & \\ y_{m,1} & \dots & y_{m,h} \end{bmatrix}$$

Least Squares Regression

If $\mathbf{X}^T\mathbf{X}$ is invertible, then

Learning/training: $\widehat{\mathbf{W}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$

Prediction/testing: $\hat{\mathbf{F}}_{\mathbf{w}}(\mathbf{X}_{new}) = \mathbf{X}_{new}\widehat{\mathbf{W}}$

$$\mathbf{X} \in \mathcal{R}^{m\times(d+1)},\ \mathbf{W} \in \mathcal{R}^{(d+1)\times h},\ \mathbf{Y} \in \mathcal{R}^{m\times h}$$

Linear Regression (for classification)

Linear Methods for Classification

- We have a collection of labeled examples
  - m is the size of the collection
  - $\mathbf{x}_i$ is the d-dimensional feature vector of example $i = 1, \ldots, m$
  - y_i is discrete target label (e.g., $y_i \in \{-1, +1\}$ or $\{0, 1\}$ for binary classification problems)
  - Note:
    - when y_i is **continuous** valued → a **regression problem**
    - when y_i is **discrete** valued →a **classification problem**
- Linear model: $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b$ or in compact form $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$ (having the offset term absorbed into the inner product)

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "Introduction to Applied Linear Algebra", Cambridge University Press, 2018 (chp.14)

Linear Regression (for classification)

Linear Methods for Classification

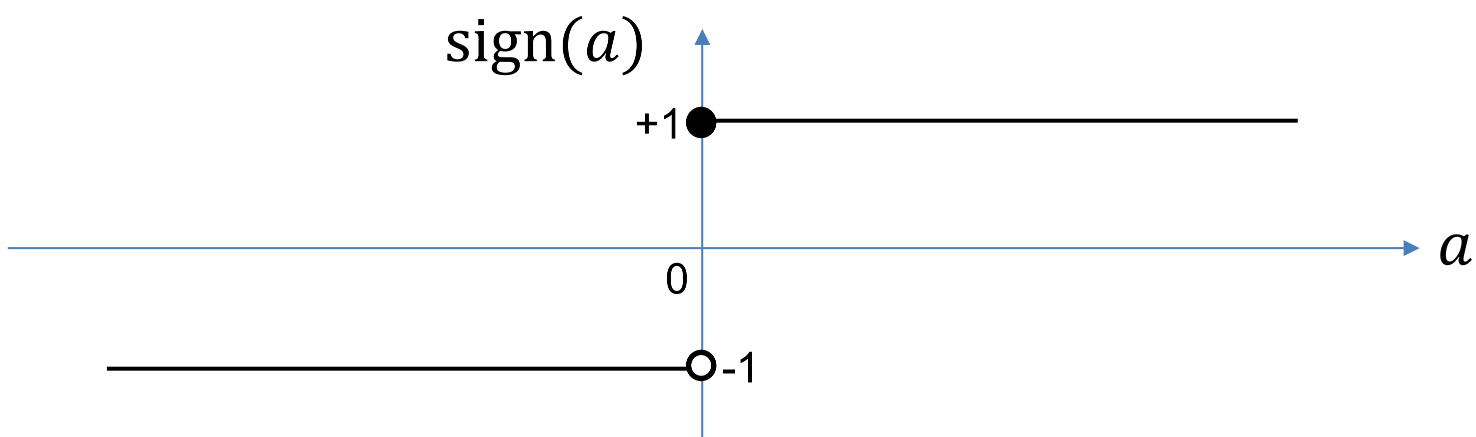
Binary Classification:

If $\mathbf{X}^T\mathbf{X}$ is invertible, then

Learning: $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$, $y_i \in \{-1, +1\}, i = 1, \dots, m$

Prediction: $\hat{f}^c_{\mathbf{w}}(\mathbf{x}_{new}) = \mathrm{sign}(\mathbf{x}_{new}^T\hat{\mathbf{w}})$ for each row $\mathbf{x}_{new}^T$ of $\mathbf{X}_{new}$

$$\mathbf{sign}(a) = +1 \text{ for } a \geq 0 \text{ and } -1 \text{ for } a < 0$$



Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "Introduction to Applied Linear Algebra", Cambridge University Press, 2018 (chp.14)

Linear Regression (for classification)

Example 1 **Training set** $\{x_i, y_i\}_{i=1}^m$

$\{x=-9\} \rightarrow \{y=-1\}$
$\{x=-7\} \rightarrow \{y=-1\}$
$\{x=-5\} \rightarrow \{y=-1\}$
$\{x=1\} \rightarrow \{y=+1\}$
$\{x=5\} \rightarrow \{y=+1\}$
$\{x=9\} \rightarrow \{y=+1\}$

Bias

$$\underbrace{\begin{bmatrix} 1 & -9 \\ 1 & -7 \\ 1 & -5 \\ 1 & 1 \\ 1 & 5 \\ 1 & 9 \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} w_0 \\ w_1 \end{bmatrix}}_{\mathbf{w}} = \underbrace{\begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \end{bmatrix}}_{\mathbf{y}}$$

This set of linear equations has NO exact solution

$$\hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

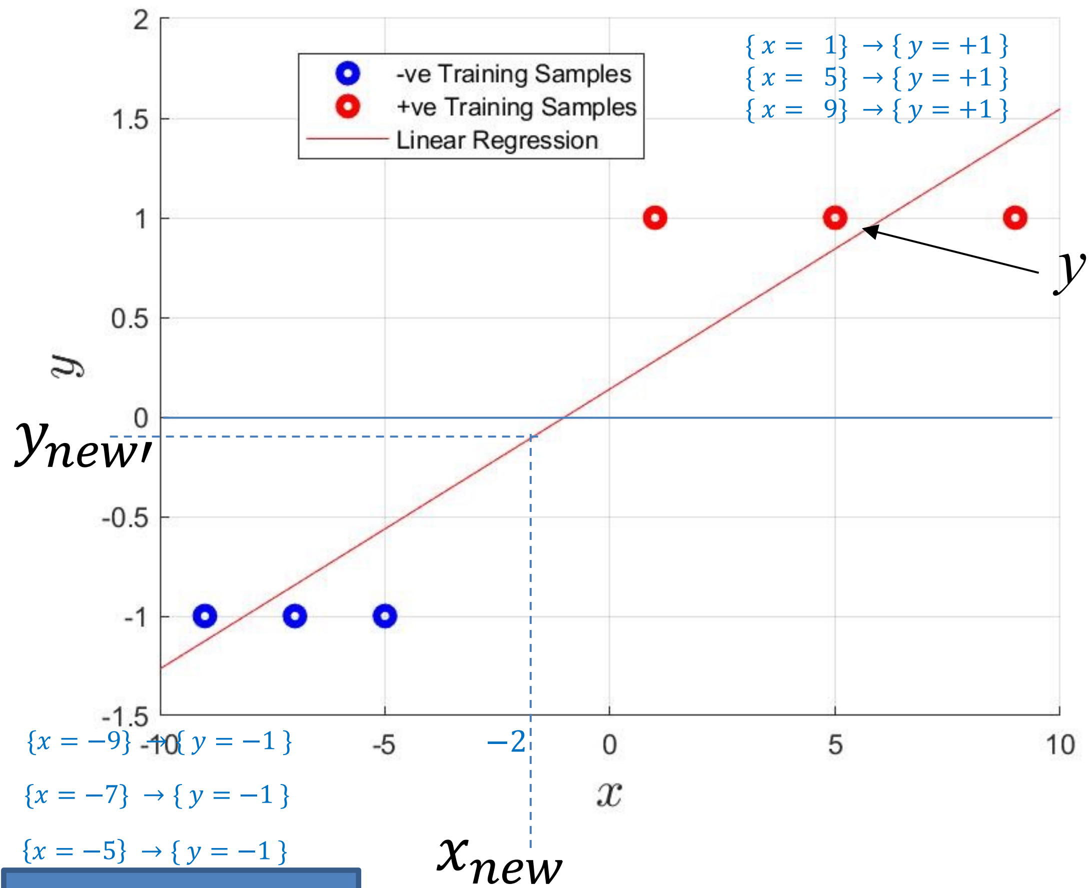
$\mathbf{X}^T\mathbf{X}$ is invertible

$$= \begin{bmatrix} 6 & -6 \\ -6 & 262 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -9 & -7 & -5 & 1 & 5 & 9 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.1406 \\ 0.1406 \end{bmatrix}$$

Least square approximation

Linear Regression (for classification)

Example 1



$$\hat{y} = \text{sign}(\mathbf{X}\hat{\mathbf{w}})$$
$$= \text{sign}\left(\mathbf{X}\begin{bmatrix}0.1406\\0.1406\end{bmatrix}\right)$$

$$y' = \mathbf{X}\hat{\mathbf{w}} = 0.1406 + 0.1406x$$

Prediction:

Test set $\{x = -2\} \rightarrow \{y = ?\}$

$$y_{new} = \hat{f}^{c}_{\mathbf{w}}(\mathbf{x}_{new}) = \text{sign}(\mathbf{x}_{new}\hat{\mathbf{w}})$$

Bias

$$= \text{sign}\left(\begin{bmatrix}1 & -2\end{bmatrix}\begin{bmatrix}0.1406\\0.1406\end{bmatrix}\right)$$

$$= \text{sign}(-0.1406) = -1$$

Python demo 1

Linear Regression for one-dimensional classification

Linear Regression (for classification)

Linear Methods for Classification

Multi-Category Classification:

If $\mathbf{X}^T\mathbf{X}$ is invertible, then

Learning: $\widehat{\mathbf{W}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}, \quad \mathbf{Y} \in \mathbf{R}^{m \times C}$

Prediction: $\hat{f}^c_{\mathbf{w}}(\mathbf{x}_{new}) = \arg\max_{k=1,\ldots,C} \left(\mathbf{x}_{new}^T \widehat{\mathbf{W}}(:,k)\right)$ for each $\mathbf{x}_{new}^T$ of $\mathbf{X}_{new}$

$\hat{y}_{new}$

Each row (of $i = 1, \ldots, m$) in $\mathbf{Y}$ has an **one-hot** encoding/assignment:

e.g., target for class-1 is labelled as $\mathbf{y}_i^T = [1, 0, 0, \ldots, 0]$ for the ith sample,

target for class-2 is labelled as $\mathbf{y}_j^T = [0, 1, 0, \ldots, 0]$ for the jth sample,

target for class-C is labelled as $\mathbf{y}_m^T = \underbrace{[0, 0, \ldots, 0, 1]}_{C}$ for the mth sample.

Ref: Hastie, Tibshirani, Friedman, "The Elements of Statistical Learning", (2nd ed., 12th printing) 2017 (chp.4)

Linear Regression (for classification)

Example 2 Three class classification

Training set $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{m}$

$\{x_1 = 1, \ \ x_2 = 1\} \rightarrow \{y_1 = 1, y_2 = 0, y_3 = 0\}$ Class 1

$\{x_1 = -1, \ x_2 = 1\} \rightarrow \{y_1 = 0, y_2 = 1, y_3 = 0\}$ Class 2

$\{x_1 = 1, \ \ x_2 = 3\} \rightarrow \{y_1 = 1, y_2 = 0, y_3 = 0\}$ Class 1

$\{x_1 = 1, \ \ x_2 = 0\} \rightarrow \{y_1 = 0, y_2 = 0, y_3 = 1\}$ Class 3

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & 0 \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & w_{3,2} & w_{3,3} \end{bmatrix}}_{\mathbf{W}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{Y}}$$

Bias → (first column of $\mathbf{X}$)

This set of linear equations has NO exact solution.

Least square approximation

$$\hat{\mathbf{W}} = \mathbf{X}^{\dagger}\mathbf{Y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

$\mathbf{X}^T\mathbf{X}$ is invertible

$$= \begin{bmatrix} 4 & 2 & 5 \\ 2 & 4 & 3 \\ 5 & 3 & 11 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & 3 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0.2857 & -0.5 & 0.2143 \\ 0.2857 & 0 & -0.2857 \end{bmatrix}$$

Linear Regression (for classification)

Example 2 Prediction

Test set $\mathbf{X}_{new}$ $\{x_1 = 6, \quad x_2 = 8\} \rightarrow \{class\ 1, 2, or\ 3?\}$

$\{x_1 = 0, \ x_2 = -1\} \rightarrow \{class\ 1, 2, or\ 3?\}$

$$\hat{\mathbf{Y}} = \mathbf{X}_{new}\hat{\mathbf{W}} = \begin{bmatrix} 1 & 6 & 8 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0.2857 & -0.5 & 0.2143 \\ 0.2857 & 0 & -0.2857 \end{bmatrix}$$

Category prediction:

$$\hat{f}^c_{\mathbf{w}}(\mathbf{X}_{new}) = \arg\max_{k=1,\ldots,C}(\hat{\mathbf{Y}}(:,k))$$

$$= \arg\max_{k=1,\ldots,C}\left(\begin{bmatrix} 4 & -2.50 & -0.50 \\ -0.2587 & 0.50 & 0.7857 \end{bmatrix}\right)$$

$$= \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

→ **Class 1**

→ **Class 3**

Python demo 2

For each row of **Y**, the **column position** of the **largest** number (across all columns for that row) determines the **class label**.

E.g. in the first row, the maximum number is 4 which is in column 1. Therefore, the resulting predicted class is 1.

Ridge Regression

Recall Linear regression

Objective: $\widehat{\mathbf{w}} = \operatorname{argmin} \sum_{i=1}^{m} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = (\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y})$

The learning computation: $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

We cannot guarantee that the matrix $\mathbf{X}^T\mathbf{X}$ is invertible

Ridge regression: shrinks the regression coefficients *w* by imposing a penalty on their size

Objective:
$$\widehat{\mathbf{w}} = \operatorname{argmin} \sum_{i=1}^{m} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 + \lambda \sum_{j=1}^{d} w_j^2$$
$$= \operatorname{argmin} (\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^T\mathbf{w}$$

Here $\lambda \geq 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of λ, the greater the amount of shrinkage.

Note: *m* samples & *d* parameters

Ridge Regression

Using a linear model:

$$\min_{\mathbf{w}} (\mathbf{Xw} - \mathbf{y})^T (\mathbf{Xw} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w}$$

Solution:

$$\frac{\partial}{\partial \mathbf{w}} \left((\mathbf{Xw} - \mathbf{y})^T (\mathbf{Xw} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w} \right) = \mathbf{0}$$

$$\Rightarrow 2\mathbf{X}^T\mathbf{Xw} - 2\mathbf{X}^T\mathbf{y} + 2\lambda\mathbf{w} = \mathbf{0}$$

$$\Rightarrow \mathbf{X}^T\mathbf{Xw} + \lambda\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

$$\Rightarrow (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

where **I** is the *dxd* identity matrix

Here on, we shall focus on single column of output $\mathbf{y}$ in derivations in the sequel

Learning: $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1} \mathbf{X}^T\mathbf{y}$

Ref: Hastie, Tibshirani, Friedman, "The Elements of Statistical Learning", (2nd ed., 12th printing) 2017 (chp.3)

Ridge Regression

Ridge Regression in Primal Form (when m > d)

$(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})$ is invertible for $\lambda > 0$,

Learning: $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\,\mathbf{X}^T\mathbf{y}$

Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{X}_{new}) = \mathbf{X}_{new}\hat{\mathbf{w}}$

Ref: Hastie, Tibshirani, Friedman, "The Elements of Statistical Learning", (2nd ed., 12th printing) 2017 (chp.3)

Ridge Regression

Ridge Regression in Dual Form (when m < d)

$(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})$ is invertible for $\lambda > 0$,

Learning: $\hat{\mathbf{w}} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\,\mathbf{y}$

Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{X}_{new}) = \mathbf{X}_{new}\hat{\mathbf{w}}$

Derivation as homework (see tutorial 6).

Hint: start off with $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{w} = \mathbf{X}^T\mathbf{y}$ and make use of $\mathbf{w} = \mathbf{X}^T\mathbf{a}$ and $\boldsymbol{a} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w}), \lambda > 0$

Polynomial Regression

Motivation: nonlinear decision surface

- Based on the sum of products of the variables
- E.g. when the input dimension is *d=2*,

a polynomial function of degree = 2 is:

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + w_{12}\,x_1x_2 + w_{11}\,x_1^2 + w_{22}\,x_2^2.$$

XOR problem

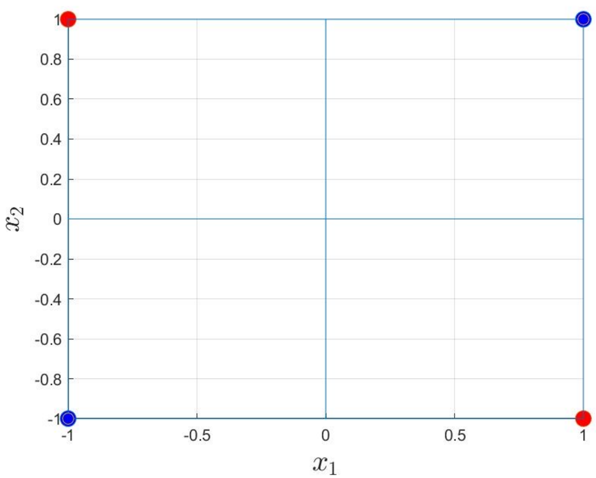
$\mathbf{x}_1 = [+1 \quad +1]^\top \qquad y_1 = +1$

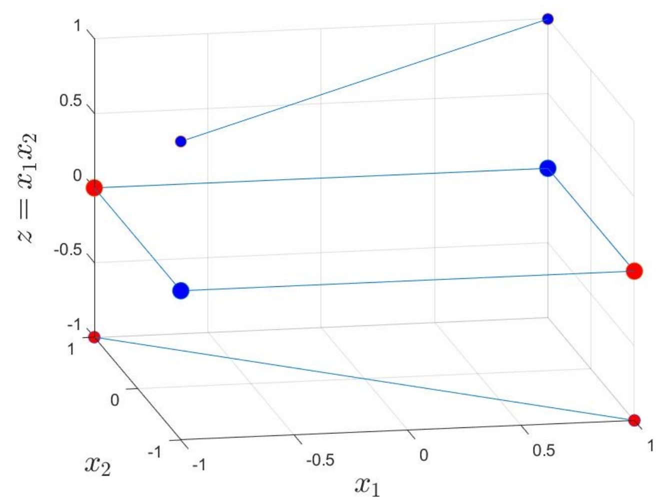
$\mathbf{x}_2 = [-1 \quad +1]^\top \qquad y_2 = -1$

$\mathbf{x}_3 = [+1 \quad -1]^\top \qquad y_3 = -1$

$\mathbf{x}_4 = [-1 \quad -1]^\top \qquad y_4 = +1$

$$f_{\mathbf{w}}(\mathbf{x}) = x_1x_2$$





Polynomial Regression

Polynomial Expansion

- The linear model $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$ can be written as

$$\begin{aligned} f_{\mathbf{w}}(\mathbf{x}) &= \mathbf{x}^T\mathbf{w} \\ &= \sum_{i=0}^{d} x_i w_i\,, \qquad x_0 = 1 \\ &= w_0 + \sum_{i=1}^{d} x_i w_i\,. \end{aligned}$$

- By including additional terms involving the products of pairs of components of $\mathbf{x}$, we obtain a quadratic model:

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} x_i x_j.$$

$C(n, r) = \frac{(n+r-1)!}{r!\,(n-1)!}$

2nd order: $f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + w_{12}x_1x_2 + w_{11}x_1^2 + w_{22}x_2^2$

3rd order: $f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + w_{12}x_1x_2 + w_{11}x_1^2 + w_{22}x_2^2 + \sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d} w_{ijk}x_ix_jx_k,\ d = 2$

Ref: Duda, Hart, and Stork, “Pattern Classification”, 2001 (Chp.5)

Polynomial Regression

1
$C(3,1)$
$C(3,2)$
$C(3,3)$

Generalized Linear Discriminant Function

- In general:

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} x_i x_j + \sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d} w_{ijk} x_i x_j x_k + \cdots$$

Weierstrass Approximation Theorem: Every continuous function defined on a closed interval [*a*, *b*] can be uniformly approximated as closely as desired by a polynomial function.

- Suppose f is a continuous real-valued function defined on the real interval $[a, b]$.
- For every $\varepsilon > 0$, there exists a polynomial p such that for all x in $[a, b]$, we have $|f(x) - p(x)| < \varepsilon$.

(Ref: https://en.wikipedia.org/wiki/Stone%E2%80%93Weierstrass_theorem)

Notes:

- For high dimensional input features (large d *value*) and high *polynomial order*, the number of polynomial terms becomes explosive! (i.e., grows exponentially)
- For high dimensional problems, polynomials of order larger than 3 is seldom used.

Ref: Duda, Hart, and Stork, "Pattern Classification", 2001 (Chp.5) online

Polynomial Regression

Generalized Linear Discriminant Function

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} x_i x_j + \sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d} w_{ijk} x_i x_j x_k + \cdots$$

$$\boldsymbol{f}_{\mathbf{w}}(\mathbf{x}) = \mathbf{P}\mathbf{w}$$

(Note: $\mathbf{P} \triangleq \mathbf{P}(\mathbf{X})$ for symbol simplicity)

$$= \begin{bmatrix} \boldsymbol{p}_1^T \mathbf{w} \\ \vdots \\ \boldsymbol{p}_m^T \mathbf{w} \end{bmatrix}$$

where $\boldsymbol{p}_l^T \mathbf{w} = [1, x_{l,1}, \ldots, x_{l,d}, \ldots, x_{l,i}x_{l,j}, \ldots, x_{l,i}x_{l,j}x_{l,k}, \ldots] \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \\ \vdots \\ w_{ij} \\ \vdots \\ w_{ijk} \\ \vdots \end{bmatrix}$

$l = 1, \ldots, m$; d denotes the dimension of input features; m denotes the number of samples

Ref: Duda, Hart, and Stork, "Pattern Classification", 2001 (Chp.5)

Example 3

Training set $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{m}$

$\{x_1 = 0,\ x_2 = 0\} \rightarrow \{y = -1\}$
$\{x_1 = 1,\ x_2 = 1\} \rightarrow \{y = -1\}$
$\{x_1 = 1,\ x_2 = 0\} \rightarrow \{y = +1\}$
$\{x_1 = 0,\ x_2 = 1\} \rightarrow \{y = +1\}$

2nd order polynomial model

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + w_{12}\, x_1 x_2 + w_{11}\, x_1^2 + w_{22}\, x_2^2$$

$$= \begin{bmatrix} 1 & x_1 & x_2 & x_1 x_2 & x_1^2 & x_2^2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_{12} \\ w_{11} \\ w_{22} \end{bmatrix}$$

Python demo 3

Stack the 4 training samples as a matrix

$$\mathbf{P} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & x_{1,1}x_{1,2} & x_{1,1}^2 & x_{1,2}^2 \\ 1 & x_{2,1} & x_{2,2} & x_{2,1}x_{2,2} & x_{2,1}^2 & x_{2,2}^2 \\ 1 & x_{3,1} & x_{3,2} & x_{3,1}x_{3,2} & x_{3,1}^2 & x_{3,2}^2 \\ 1 & x_{4,1} & x_{4,2} & x_{4,1}x_{4,2} & x_{4,1}^2 & x_{4,2}^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Polynomial Regression

Summary

Ridge Regression in Primal Form (m > d)

For $\lambda > 0$,

Learning: $\hat{\mathbf{w}} = (\mathbf{P}^T\mathbf{P} + \lambda\mathbf{I})^{-1}\,\mathbf{P}^T\mathbf{y}$

Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \mathbf{P}_{new}\hat{\mathbf{w}}$

Ridge Regression in Dual Form (m < d)

For $\lambda > 0$,

Learning: $\hat{\mathbf{w}} = \mathbf{P}^T(\mathbf{P}\mathbf{P}^T + \lambda\mathbf{I})^{-1}\,\mathbf{y}$

Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \mathbf{P}_{new}\hat{\mathbf{w}}$

Note: Change **X** to **P** with reference to slides 15/16; m & d refers to the size of **P** (not **X**)

Polynomial Regression

Summary

For Regression Applications

- Learn **continuous** valued y using either primal form or dual form
- Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \mathbf{P}_{new}\hat{\mathbf{w}}$

For Classification Applications

- Learn **discrete** valued y ($y \in \{-1, +1\}$) or $\mathbf{Y}$ (one-hot) using either primal form or dual form
- Binary Prediction: $\hat{f}^{c}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \text{sign}(\mathbf{P}_{new}\hat{\mathbf{w}})$
- Multi-Category Prediction: $\hat{f}^{c}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \arg\max_{k=1,\ldots,C}(\mathbf{P}_{new}\widehat{\mathbf{W}}(:,k))$

Example 3 (cont'd)

Training set

$\{x_1 = 0, x_2 = 0\} \rightarrow \{y = -1\}$
$\{x_1 = 1, x_2 = 1\} \rightarrow \{y = -1\}$
$\{x_1 = 1, x_2 = 0\} \rightarrow \{y = +1\}$
$\{x_1 = 0, x_2 = 1\} \rightarrow \{y = +1\}$

2nd order polynomial model

$$\mathbf{P} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & x_{1,1}x_{1,2} & x_{1,1}^2 & x_{1,2}^2 \\ 1 & x_{2,1} & x_{2,2} & x_{2,1}x_{2,2} & x_{2,1}^2 & x_{2,2}^2 \\ 1 & x_{3,1} & x_{3,2} & x_{3,1}x_{3,2} & x_{3,1}^2 & x_{3,2}^2 \\ 1 & x_{4,1} & x_{4,2} & x_{4,1}x_{4,2} & x_{4,1}^2 & x_{4,2}^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\hat{\mathbf{w}} = \mathbf{P}^T(\mathbf{P}\mathbf{P}^T)^{-1}\mathbf{y}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 6 & 3 & 3 \\ 1 & 3 & 3 & 1 \\ 1 & 3 & 1 & 3 \end{bmatrix}^{-1} \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -4 \\ 1 \\ 1 \end{bmatrix}$$

Python demo 3

Example 3 (cont'd) Prediction

Test set

Test point 1: $\{x_1 = 0.1, x_2 = 0.1\} \rightarrow \{y = \text{class } -1 \text{ or } +1?\}$

Test point 2: $\{x_1 = 0.9, x_2 = 0.9\} \rightarrow \{y = \text{class } -1 \text{ or } +1?\}$

Test point 3: $\{x_1 = 0.1, x_2 = 0.9\} \rightarrow \{y = \text{class } -1 \text{ or } +1?\}$

Test point 4: $\{x_1 = 0.9, x_2 = 0.1\} \rightarrow \{y = \text{class } -1 \text{ or } +1?\}$

$$\hat{\mathbf{y}} = \mathbf{P}_{new}\,\hat{\mathbf{w}}$$

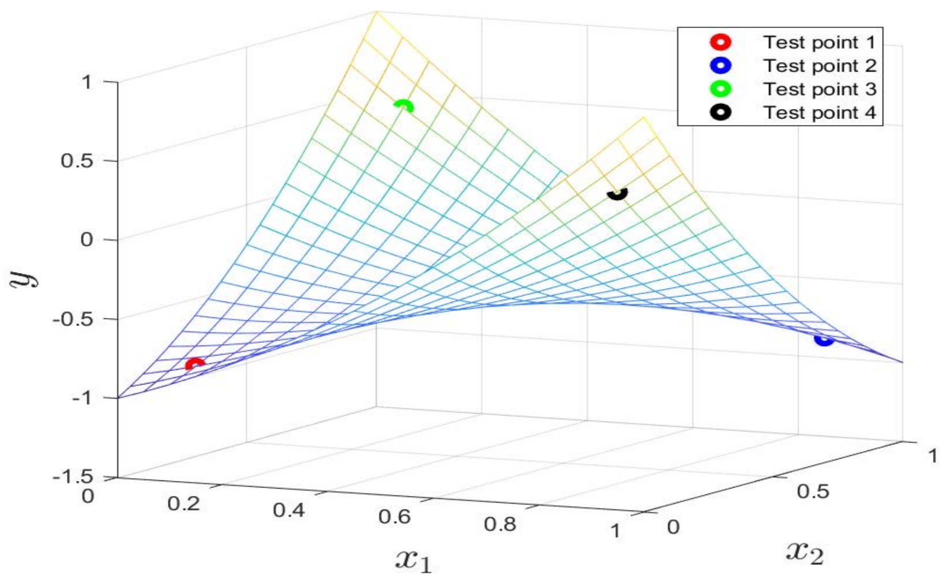
$$= \begin{bmatrix} 1 & 0.1 & 0.1 & 0.01 & 0.01 & 0.01 \\ 1 & 0.9 & 0.9 & 0.81 & 0.81 & 0.81 \\ 1 & 0.1 & 0.9 & 0.09 & 0.01 & 0.81 \\ 1 & 0.9 & 0.1 & 0.09 & 0.81 & 0.01 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ 1 \\ -4 \\ 1 \\ 1 \end{bmatrix}$$

$[1 \quad x_1 \quad x_2 \quad x_1x_2 \quad x_1^2 \quad x_2^2]$

$$\hat{f}^{c}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \text{sign}(\hat{\mathbf{y}}) = \text{sign}\left(\begin{bmatrix} -0.82 \\ -0.82 \\ 0.46 \\ 0.46 \end{bmatrix}\right)$$

$$= \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$

- Class -1
- Class -1
- Class $+1$
- Class $+1$



Poll on PollEv.com/ymjin
Just “**skip**” if you are required to do registration

Summary

Mid-term: Lecture 1 to 5
Trial quiz
Assignment 1 & 2

- Notations, Vectors, Matrices
- Operations on Vectors and Matrices
- Systems of Linear Equations $f_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w} = \mathbf{y}$
- Functions, Derivative and Gradient
- Least Squares, Linear Regression with Single and Multiple Outputs
- Learning of vectored function, binary and multi-category classification
- Ridge Regression: penalty term, primal and dual forms
- Polynomial Regression: nonlinear decision boundary

Primal form Learning: $\hat{\mathbf{w}} = (\mathbf{P}^T\mathbf{P} + \lambda\mathbf{I})^{-1}\,\mathbf{P}^T\mathbf{y}$

Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \mathbf{P}_{new}\hat{\mathbf{w}}$

Dual form Learning: $\hat{\mathbf{w}} = \mathbf{P}^T(\mathbf{P}\mathbf{P}^T + \lambda\mathbf{I})^{-1}\,\mathbf{y}$

Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \mathbf{P}_{new}\hat{\mathbf{w}}$

Hint: python packages: sklearn.preprocessing (PolynomialFeatures), np.sign, sklearn.model_selection (train_test_split), sklearn.preprocessing (OneHotEncoder)