EE2211 Lecture 6: Linear Classification, Ridge Regression, Polynomial Regression

Vincent Y. F. Tan

Department of Electrical and Computer Engineering, NUS

EE2211 Spring 2023

Acknowledgements to

Xinchao, Helen, Thomas, Kar Ann, Chen Khong, Robby, and Haizhou

Outline

Review of Linear Regression

- (Learning/Training) Given a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, the least squares solution (with offset) is

$$\overline{\mathbf{w}}^* = \begin{bmatrix} b^* \\ \mathbf{w}^* \end{bmatrix} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \in \mathbb{R}^{d+1}$$

where the design matrix and target vector are

$$\mathbf{X} = \begin{bmatrix} - \overline{\mathbf{x}}_1^\top - \\ - \overline{\mathbf{x}}_2^\top - \\ \vdots \\ - \overline{\mathbf{x}}_m^\top - \end{bmatrix} = \begin{bmatrix} 1 & - \mathbf{x}_1^\top - \\ 1 & - \mathbf{x}_2^\top - \\ \vdots & \vdots \\ 1 & - \mathbf{x}_m^\top - \end{bmatrix} \in \mathbb{R}^{m \times (d+1)} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \in \mathbb{R}^m.$$

- (Prediction/Testing) Given a new feature vector (sample, example) $\mathbf{x}_{\text{new}}$, the prediction based on the least squares solution is

$$\hat{y}_{\text{new}} = \begin{bmatrix} 1 \\ \mathbf{x}_{\text{new}} \end{bmatrix}^\top \overline{\mathbf{w}}^* = b^* + \mathbf{x}_{\text{new}}^\top \mathbf{w}^*.$$

Review of Linear Regression With Multiple Outputs

- Suppose there are h outputs we want to predict (above $h = 3$).
- Given a dataset $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$ where $\mathbf{x}_i \in \mathbb{R}^d$ (column vector) and $\mathbf{y}_i \in \mathbb{R}^{1\times h}$ (row vector), the model to be used is

$$\underbrace{\begin{bmatrix} y_{1,1} & y_{1,2} & \dots & y_{1,h} \\ y_{2,1} & y_{2,2} & \dots & y_{2,h} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m,1} & y_{m,2} & \dots & y_{m,h} \end{bmatrix}}_{\mathbf{Y}\in\mathbb{R}^{m\times h}} = \underbrace{\begin{bmatrix} 1 & x_{1,1} & \dots & x_{1,d} \\ 1 & x_{2,1} & \dots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1} & \dots & x_{m,d} \end{bmatrix}}_{\mathbf{X}\in\mathbb{R}^{m\times(d+1)}} \underbrace{\begin{bmatrix} b_1 & b_2 & \dots & b_h \\ w_{1,1} & w_{1,2} & \dots & w_{1,h} \\ \vdots & \vdots & \ddots & \vdots \\ w_{d,1} & w_{d,2} & \dots & w_{d,h} \end{bmatrix}}_{\overline{\mathbf{W}}\in\mathbb{R}^{(d+1)\times h}}$$

- When $h = 1$, this particularizes to standard linear regression.
- This is exactly h separate linear regression problems.

Review of Linear Regression With Multiple Outputs

- (Training/Learning) Least Squares Solution

$$\overline{\mathbf{W}}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} \in \mathbb{R}^{(d+1)\times h}.$$

- (Testing/Prediction) Given a new feature vector $\mathbf{x}_{\text{new}} \in \mathbb{R}^d$, we can predict its h outputs as

$$\hat{\mathbf{y}}_{\text{new}} = \begin{bmatrix} 1 \\ \mathbf{x}_{\text{new}} \end{bmatrix}^\top \overline{\mathbf{W}}^* \in \mathbb{R}^{1\times h}$$

- The k-th ($1 \le k \le h$) component of $\hat{\mathbf{y}}_{\text{new}}$ is the prediction of the k-th output based the dataset $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$.

Outline

- We have a collection of m labelled samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ where
  1. $\mathbf{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,d}]^\top \in \mathbb{R}^d$ is the i-th feature vector;
  2. y_i is a discrete label.
- In binary classification, we can encode y_i as $y_i = +1$ (positive class) and $y_i = -1$ (negative class).
- m is the number of data samples (feature vectors) in the dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$.
- d is the dimension of each data sample, i.e., length of each $\mathbf{x}_i$.
- We assume the affine model

$$f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} + b = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}^\top \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix} = \overline{\mathbf{x}}^\top \overline{\mathbf{w}},$$

where

$$\overline{\mathbf{x}} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix} \in \mathbb{R}^{d+1} \qquad \overline{\mathbf{w}} = \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix} \in \mathbb{R}^{d+1}$$

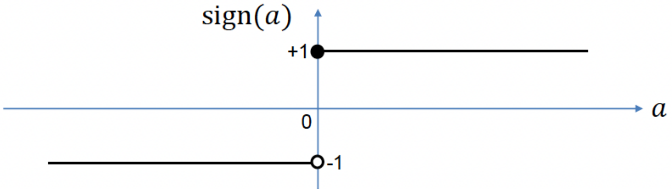
Linear Models for Classification

- The main idea is to treat binary classification as regression where each label y_i can only take on -1 or $+1$.
- If in testing/prediction, $\bar{\mathbf{x}}_{\text{new}}^\top \bar{\mathbf{w}}^*$ is positive (resp. negative), predict that $\hat{y}_{\text{new}} = +1$ (resp. $\hat{y}_{\text{new}} = -1$). For example, distinguishing between cats and dogs.
- (Learning/Training) Given a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ (where each $y_i \in \{+1, -1\}$), learn the weights using least squares

$$\bar{\mathbf{w}}^* = \begin{bmatrix} b^* \\ \mathbf{w}^* \end{bmatrix} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \in \mathbb{R}^{d+1}.$$

- (Prediction/Testing) Given a new data sample $\mathbf{x}_{\text{new}} \in \mathbb{R}^d$, its predicted label is

$$\hat{y}_{\text{new}} = \mathrm{sign}\left(\bar{\mathbf{x}}_{\text{new}}^\top \bar{\mathbf{w}}^*\right) = \mathrm{sign}\left(\begin{bmatrix} 1 \\ \mathbf{x}_{\text{new}} \end{bmatrix}^\top \bar{\mathbf{w}}^*\right) \in \{+1, -1\}.$$



For example,

- If the raw prediction $\overline{\mathbf{x}}_{\text{new}}^{\top}\overline{\mathbf{w}}^{*} = 0.2$, the predicted class is $+1$;
- If the raw prediction $\overline{\mathbf{x}}_{\text{new}}^{\top}\overline{\mathbf{w}}^{*} = -0.8$, the predicted class is -1;
- If the raw prediction $\overline{\mathbf{x}}_{\text{new}}^{\top}\overline{\mathbf{w}}^{*} = 0.0$, we declare error.

Numerical Example for Binary Classification

- Dataset $(\mathbf{x}_i, y_i), i = 1, 2, 3, 4$ includes the samples

$$\mathbf{x}_1 = -7, \quad \mathbf{x}_2 = -5, \quad \mathbf{x}_3 = 1, \quad \mathbf{x}_4 = 5$$
$$y_1 = -1, \quad y_2 = -1, \quad y_3 = +1, \quad y_4 = +1$$

- Here, $m = 4$ and $d = 1$ (scalar features).
- Design matrix and target vector are

$$\mathbf{X} = \begin{bmatrix} 1 & -7 \\ 1 & -5 \\ 1 & 1 \\ 1 & 5 \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$

- The linear system $\mathbf{X}\overline{\mathbf{w}} = \mathbf{y}$ is overdetermined and there is no solution for $\overline{\mathbf{w}}$ because

$$\text{rank}(\mathbf{X}) < \text{rank}(\tilde{\mathbf{X}}).$$

Numerical Example for Binary Classification

- Using some numerical software, we can find the least square approximation

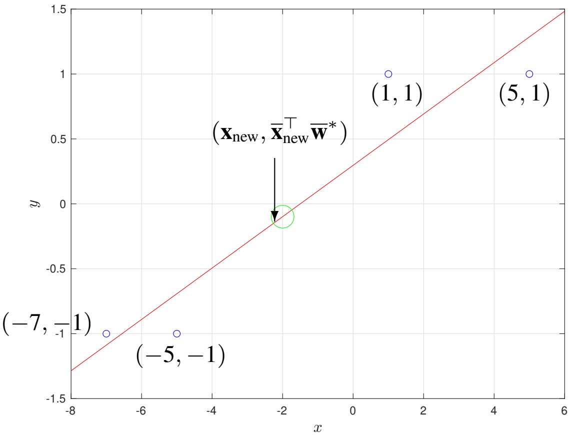
$$\overline{\mathbf{w}}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \begin{bmatrix} 0.2967 \\ 0.1978 \end{bmatrix}$$

- If we want to predict what's the label for $\mathbf{x}_{\text{new}} = -2$, we plug $\mathbf{x}_{\text{new}} = -2$ into the learned affine model to get

$$\begin{aligned} \hat{y}_{\text{new}} &= \text{sign}\left(\begin{bmatrix} 1 \\ \mathbf{x}_{\text{new}} \end{bmatrix}^\top \overline{\mathbf{w}}^* \right) \\ &= \text{sign}\left(1 \times (0.2967) + (-2) \times (0.1978)\right) \\ &= \text{sign}(-0.0989) = -1. \end{aligned}$$

- So we predict that the label of the new test point $\mathbf{x}_{\text{new}} = -2$ is $\hat{y}_{\text{new}} = -1$ (negative class). [Python demo]

Numerical Example for Binary Classification



The predicted label of new point $\mathbf{x}_{\mathrm{new}}$ is $\mathrm{sign}(\overline{\mathbf{x}}_{\mathrm{new}}^\top \overline{\mathbf{w}}^*) = -1$ as $\overline{\mathbf{x}}_{\mathrm{new}}^\top \overline{\mathbf{w}}^*$ is negative.

Python Demo 1

```python
# EE2211 Lecture 6 Demo 1 Binary classification
import numpy as np
from numpy.linalg import inv

X = np.array([[1,-7], [1,-5], [1,1], [1,5]])
y = np.array([[-1], [-1], [1], [1]])
## Linear regression for classification
w = inv(X.T @ X) @ X.T @ y
print("Estimated w")
print(w)
print("\n")

Xt = np.array([[1,-2]])
y_predict = Xt @ w
print("Predicted y")
print(y_predict)
y_class_predict = np.sign(y_predict)
print("Predicted y class")
print(y_class_predict)
```

Linear Models for Multi-Class Classification

- Suppose we want to distinguish between cats, dogs and birds. These are labelled as 1, 2, 3 respectively.
- Idea is to do one-hot encoding of the labels, say $\{1, 2, \ldots, C\}$, where $C > 2$ is the number of classes.
- If sample i has class 1, its label vector is

$$\mathbf{y}_i = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \end{bmatrix}$$

- If sample i has class 2, its label vector is

$$\mathbf{y}_i = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \end{bmatrix}$$

- If sample i has class C, its label vector is

$$\mathbf{y}_i = \begin{bmatrix} 0 & 0 & 0 & \ldots & 1 \end{bmatrix}$$

Linear Models for Multi-Class Classification

- Stack all these label vectors into the $m \times C$ label matrix

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_m \end{bmatrix} = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,C} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,C} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m,1} & y_{m,2} & \cdots & y_{m,C} \end{bmatrix}$$

- This is a $\{0, 1\}$-valued matrix with m (number of samples) rows and C (number of classes) columns.
- Essentially, we are doing C separate linear classification problems.
- Each determining the "likelihood" of whether we are in class $k \in \{1, 2, \ldots, C\}$.

Linear Models for Multi-Class Classification

- (Training/Learning) The design matrix $\mathbf{X}$ is the same. If it has full column rank, find the least squares solution

$$\overline{\mathbf{W}}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} \in \mathbb{R}^{(d+1)\times C}.$$

- (Testing/Prediction) Given a new feature vector $\mathbf{x}_{\text{new}} \in \mathbb{R}^d$, we can predict its class as

$$\hat{y}_{\text{new}} = \underset{k\in\{1,2,\ldots,C\}}{\arg\max} \left(\begin{bmatrix} 1 \\ \mathbf{x}_{\text{new}} \end{bmatrix}^\top \overline{\mathbf{W}}^*[:,k] \right) \in \{1,2,\ldots,C\}$$

where $\overline{\mathbf{W}}^*[:,k] \in \mathbb{R}^{d+1}$ is the k-column of $\overline{\mathbf{W}}^*$.

Numerical Example for Multi-Class Classification

- Our $m = 4$ feature vectors are

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{x}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad \mathbf{x}_2 = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad \mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

  Each is of dimension $d = 2$.
- The raw classes (there are $C = 3$ of them) are

$$y_1 = \text{cat}, \quad y_2 = \text{dog}, \quad y_3 = \text{cat}, \quad y_4 = \text{bird}.$$

- First encode the raw classes into numerical classes, e.g.,

$$y_1 = 1, \quad y_2 = 2, \quad y_3 = 1, \quad y_4 = 3.$$

  Thus $\text{cat} \equiv 1$, $\text{dog} \equiv 2$, $\text{bird} \equiv 3$.
- One-hot encoding in operation!

$$\mathbf{y}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \ \mathbf{y}_2 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \ \mathbf{y}_3 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \ \mathbf{y}_4 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix},$$

Numerical Example for Multi-Class Classification

- Design matrix (with bias all-ones column) and target matrix are

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & 0 \end{bmatrix} \in \mathbb{R}^{m \times (d+1)} \qquad \mathbf{Y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{m \times C}.$$

  Please check that you know where these numbers came from.

- (Training/Learning) Least squares approximation

$$\overline{\mathbf{W}}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0.2857 & -0.5 & 0.2143 \\ 0.2857 & 0 & -0.2857 \end{bmatrix} \in \mathbb{R}^{(d+1) \times C}$$

Numerical Example for Multi-Class Classification

- (Prediction/Testing) Given a new sample $\mathbf{x}_{\text{new}} = \begin{bmatrix} 0 & -1 \end{bmatrix}^\top$.
- For each $k = 1, 2, 3$, calculate $\begin{bmatrix} 1 \\ \mathbf{x}_{\text{new}} \end{bmatrix}^\top \overline{\mathbf{W}}^*[:, k]$.
- We obtain

$$\begin{bmatrix} 1 \\ \mathbf{x}_{\text{new}} \end{bmatrix}^\top \overline{\mathbf{W}}^*[:, 1] = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}^\top \begin{bmatrix} 0 \\ 0.2857 \\ 0.2857 \end{bmatrix} = -0.2857$$

$$\begin{bmatrix} 1 \\ \mathbf{x}_{\text{new}} \end{bmatrix}^\top \overline{\mathbf{W}}^*[:, 2] = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}^\top \begin{bmatrix} 0.5 \\ -0.5 \\ 0 \end{bmatrix} = 0.5,$$

$$\begin{bmatrix} 1 \\ \mathbf{x}_{\text{new}} \end{bmatrix}^\top \overline{\mathbf{W}}^*[:, 3] = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}^\top \begin{bmatrix} 0.5 \\ 0.2143 \\ -0.2857 \end{bmatrix} = 0.7857$$

- (Prediction/Testing) Its predicted class is

$$\hat{y}_{\text{new}} = \underset{k \in \{1,2,3\}}{\arg\max} \left(\begin{bmatrix} 1 \\ \mathbf{x}_{\text{new}} \end{bmatrix}^\top \overline{\mathbf{W}}^*[:, k] \right) = 3 \in \{1, 2, 3\}$$

The column position $k \in \{1, 2, 3\}$ of the largest number determines the predicted class label. [Python Demo]

Python Demo 2: Setting Up and One-Hot Encoding

```python
# EE2211 Lecture 6 Demo 2 Multi-class classification
import numpy as np
from numpy.linalg import inv
from sklearn.preprocessing import OneHotEncoder
X = np.array([[1, 1, 1], [1, -1, 1], [1, 1, 3], [1, 1, 0]])
y_class = np.array([[1], [2], [1], [3]])
y_onehot = np.array([[1, 0, 0], [0, 1, 0], [1, 0, 0], [0, 0, 1]])
print("One-hot encoding manual")
print(y_class)
print(y_onehot)
print("\n")

print("One-hot encoding function")
onehot_encoder=OneHotEncoder(sparse=False)
print(onehot_encoder)
Ytr_onehot = onehot_encoder.fit_transform(y_class)
print(Ytr_onehot)
```

Python Demo 2: Training and Testing

```
## Linear Classification
print("Estimated W")
W = inv(X.T @ X) @ X.T @ Ytr_onehot
print(W)
X_test = np.array([[1, 0, -1]])
yt_est = X_test@W;
print("\n")
print("Test")
print(yt_est)
yt_class = [[1 if y == max(x) else 0 for y in x] for x in yt_est ]
print("\n")
print("class label test")
print(yt_class)

print("\n")
print("class label test using argmax")
print(np.argmax(yt_est)+1)
```

Outline

Motivation for Ridge Regression

- I was involved in the Manchester Asthma & Allergy Study (MAAS)
- About $m \approx 1000$ children (subjects are expensive to recruit)
- Number of variables $d \approx 10^6$ (modern equipment can acquire huge amounts of data)
- Environmental, Physiological and Genetic variables (e.g., Single Nucleotide Polymorphisms or SNPs)






Motivation for Ridge Regression

- This is the case of modern datasets which have many variables/attributes (d is large) and few samples (m is small).
- What happens to the least squares estimate?

$$\bar{\mathbf{w}}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \in \mathbb{R}^{d+1}?$$

Recall that this was obtained from minimizing

$$J(\bar{\mathbf{w}}) = \sum_{i=1}^{m} \left(f_{\mathbf{w},b}(\mathbf{x}_i) - y_i \right)^2 = (\mathbf{X}\bar{\mathbf{w}} - \mathbf{y})^\top (\mathbf{X}\bar{\mathbf{w}} - \mathbf{y})$$

over $\bar{\mathbf{w}} = \left[b, \mathbf{w}^\top \right]^\top \in \mathbb{R}^{d+1}$.

- The design matrix $\mathbf{X} \in \mathbb{R}^{m \times (d+1)}$ is very "wide".
- $\mathbf{X}$ is highly unlikely to have full column rank.
- $(\mathbf{X}^\top \mathbf{X})^{-1}$ does not exist.
- We need to mitigate this problem.

Motivation for Ridge Regression

New Objective Function for Ridge Regression

- For a fixed $\lambda \geq 0$, consider

$$J(\overline{\mathbf{w}}) = \sum_{i=1}^{m} \left(f_{\mathbf{w},b}(\mathbf{x}_i) - y_i\right)^2 + \lambda \sum_{i=0}^{d} w_j^2$$
$$= (\mathbf{X}\overline{\mathbf{w}} - \mathbf{y})^\top (\mathbf{X}\overline{\mathbf{w}} - \mathbf{y}) + \lambda \overline{\mathbf{w}}^\top \overline{\mathbf{w}}$$

  Note that $w_0 = b$, the offset or bias.
- The term $\lambda \overline{\mathbf{w}}^\top \overline{\mathbf{w}}$ encourages the weight vector to have small components (also known as shrinkage.
- The new objective results in ridge regression or Tikhonov regularization.
- When $\lambda = 0$, we recover usual linear regression.

Solution for Ridge Regression

Solution for Ridge Regression

- Recall that we wish to solve

$$\overline{\mathbf{w}}^* = \underset{\overline{\mathbf{w}}=[b,\mathbf{w}]^\top}{\arg\min} \ (\mathbf{X}\overline{\mathbf{w}} - \mathbf{y})^\top (\mathbf{X}\overline{\mathbf{w}} - \mathbf{y}) + \lambda \overline{\mathbf{w}}^\top \overline{\mathbf{w}}.$$

- Expanding the objective, we obtain

$$\begin{aligned}
&(\mathbf{X}\overline{\mathbf{w}} - \mathbf{y})^\top (\mathbf{X}\overline{\mathbf{w}} - \mathbf{y}) + \lambda \overline{\mathbf{w}}^\top \overline{\mathbf{w}} \\
&= \overline{\mathbf{w}}^\top \mathbf{X}^\top \mathbf{X}\overline{\mathbf{w}} - \overline{\mathbf{w}}^\top \mathbf{X}^\top \mathbf{y} - \mathbf{y}^\top \mathbf{X}\overline{\mathbf{w}} + \mathbf{y}^\top \mathbf{y} + \lambda \overline{\mathbf{w}}^\top \overline{\mathbf{w}} \\
&= \overline{\mathbf{w}}^\top \mathbf{X}^\top \mathbf{X}\overline{\mathbf{w}} + \overline{\mathbf{w}}^\top (\lambda \mathbf{I}) \overline{\mathbf{w}} - 2\overline{\mathbf{w}}^\top (\mathbf{X}^\top \mathbf{y}) + \mathbf{y}^\top \mathbf{y} \\
&= \overline{\mathbf{w}}^\top (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}) \overline{\mathbf{w}} - 2\overline{\mathbf{w}}^\top (\mathbf{X}^\top \mathbf{y}) + \mathbf{y}^\top \mathbf{y}
\end{aligned}$$

- Differentiating w.r.t. $\overline{\mathbf{w}}$ and setting the result to zero yields

$$2(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})\overline{\mathbf{w}}^* = 2(\mathbf{X}^\top \mathbf{y}) \quad \Longleftrightarrow \quad \overline{\mathbf{w}}^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}.$$

- Voila! For any $\lambda > 0$, $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}$ is always invertible (why?) so the calculation above is legitimate.

Ridge Regression in Primal Form

- Training/Learning: Minimizing the ridge regression objective $J(\overline{\mathbf{w}}) = (\mathbf{X}\overline{\mathbf{w}} - \mathbf{y})^\top (\mathbf{X}\overline{\mathbf{w}} - \mathbf{y}) + \lambda \overline{\mathbf{w}}^\top \overline{\mathbf{w}}$ yields

$$\overline{\mathbf{w}}^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}.$$

- Testing/Prediction: Given a new test sample $\mathbf{x}_{\text{new}}$, its prediction is

$$\hat{y}_{\text{new}} = \begin{bmatrix} 1 \\ \mathbf{x}_{\text{new}} \end{bmatrix}^\top \overline{\mathbf{w}}^*.$$

Ridge Regression in Primal Form

- The solution is known as the

$$\text{[Primal Form]} \qquad \overline{\mathbf{w}}^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_{d+1})^{-1} \mathbf{X}^\top \mathbf{y}.$$

  Use $\mathbf{I}_{d+1}$ to emphasize that the identity matrix is of size $(d+1) \times (d+1)$.
- What's the problem with inverting the $(d+1) \times (d+1)$ matrix $\mathbf{X}^\top \mathbf{X} + \lambda_{d+1} \mathbf{I}$?
- $d > m$ is very large. Inverting the $(d+1) \times (d+1)$ matrix is not advisable!
- This takes $\approx d^3$ operations (multiplications and additions). **[You don't need to know why.]**
- If $m > d$, we can still use

$$\overline{\mathbf{w}}^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_{d+1})^{-1} \mathbf{X}^\top \mathbf{y}.$$

Ridge Regression in Dual Form

- Fact: For every $\lambda > 0$,

$$(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_{d+1})^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_m)^{-1} \mathbf{y}. \qquad \text{(P-D)}$$

- Training/Learning: So when $d > m$ (modern datasets), we use the

$$\text{[Dual Form]} \qquad \overline{\mathbf{w}}^* = \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_m)^{-1} \mathbf{y}.$$

- Testing/Prediction: Given a new test sample $\mathbf{x}_{\text{new}}$, its prediction is

$$\hat{y}_{\text{new}} = \begin{bmatrix} 1 \\ \mathbf{x}_{\text{new}} \end{bmatrix}^\top \overline{\mathbf{w}}^*.$$

- To show (P-D), we use the Woodbury formula

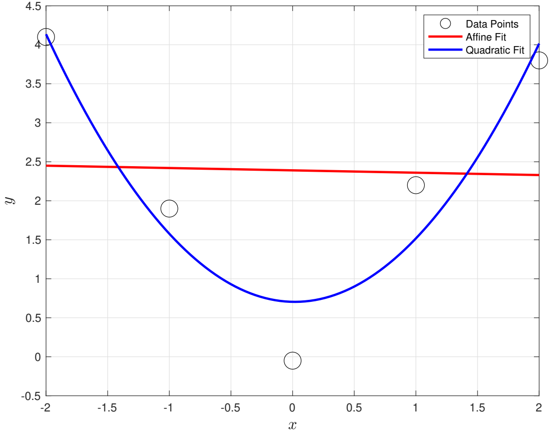
$$(\mathbf{I} + \mathbf{U}\mathbf{V})^{-1} = \mathbf{I} - \mathbf{U}(\mathbf{I} + \mathbf{V}\mathbf{U})^{-1}\mathbf{V}.$$

This will be an exercise in Tutorial 6.

Outline

Motivation for Polynomial Regression

Sometimes affine functions do not do a good job!

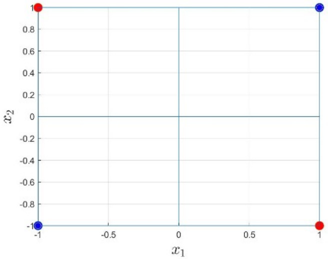


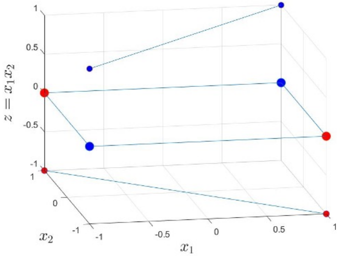
Data points come from a quadratic. Class of affine functions is not sufficiently rich.

Motivation for Polynomial Regression

XOR dataset in $d = 2$ dimensions.

$$\mathbf{x}_1 = \begin{bmatrix} +1 & +1 \end{bmatrix}^\top \quad \mathbf{x}_2 = \begin{bmatrix} -1 & +1 \end{bmatrix}^\top \quad \mathbf{x}_3 = \begin{bmatrix} +1 & -1 \end{bmatrix}^\top \quad \mathbf{x}_4 = \begin{bmatrix} -1 & -1 \end{bmatrix}^\top$$





- No linear/affine classifier can separate the training samples without error.
- The quadratic function $f(x_1, x_2) = x_1x_2$ (product of first and second components) can separate the training samples without error.

Polynomials

- We would like to model nonlinear decision boundaries or surfaces.
- A polynomial function of order 2 with $d = 1$ variables

$$f_{\mathbf{w}}(x) = w_0 + w_1 x + w_2 x^2 \qquad \mathbf{w} = (w_0, w_1, w_2)$$

- A polynomial function of order p with $d = 1$ variables

$$f_{\mathbf{w}}(x) = w_0 + w_1 x + w_2 x^2 + \ldots + w_p x^p \qquad \mathbf{w} = (w_0, w_1, \ldots, w_p)$$

- A polynomial function of order 1 with $d = 2$ variables

$$f_{\mathbf{w}}(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_2 \qquad \mathbf{w} = (w_0, w_1, w_2)$$

- A polynomial function of order 2 with $d = 2$ variables

$$f_{\mathbf{w}}(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_2 + w_{1,2} x_1 x_2 + w_{1,1} x_1^2 + w_{2,2} x_2^2$$
$$\mathbf{w} = (w_0, w_1, w_2, w_{1,2}, w_{1,1}, w_{2,2})$$

- For example, a polynomial function of order 2 in dimension $d = 2$

$$f_{\mathbf{w}}(x_1, x_2) = w_0 + w_1 x_1^{1} + w_2 x_2^{1} + w_{1,2} x_1^{1} x_2^{1} + w_{1,1} x_1^{2} + w_{2,2} x_2^{2}$$
$$\mathbf{w} = (w_0, w_1, w_2, w_{1,2}, w_{1,1}, w_{2,2})$$

Each term in $f_{\mathbf{w}}(x_1, x_2)$ is called a monomial. The maximum sum of powers (degree) of the x_1, x_2 terms is 2, e.g.,

$$\deg(w_2 x_2^{1}) = 0 + 1 = 1$$
$$\deg(w_{1,2} x_1^{1} x_2^{1}) = 1 + 1 = 2$$
$$\deg(w_{2,2} x_2^{2}) = 0 + 2 = 2$$

- In general, for d-variable quadratic (order-2) model,

$$f_{\mathbf{w}}(x_1, x_2, \ldots, x_d) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{1 \le i \le j \le d} w_{i,j} x_i x_j.$$

[Optional to know] How many terms are there here?

Polynomials

- For d-variable, cubic model,

$$f_{\mathbf{w}}(x_1, x_2, \ldots, x_d) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{1 \le i \le j \le d} w_{i,j} x_i x_j + \sum_{1 \le i \le j \le k \le d} w_{i,j,k} x_i x_j x_k$$

[Optional to know] How many terms are there here?

$$\binom{d-1}{0} + \binom{d}{1} + \binom{d+1}{2} + \binom{d+2}{3} = \binom{d+3}{3}.$$

- For a d-variable, order-p polynomial, there are

$$\binom{d+p}{p} \quad \text{terms.}$$

- The point is that if d and/or p is large, this is a very large number.

Polynomial Regression

- Generalized Linear Discriminant Function

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

- Noting that $x_{l,i}$ is the i-th ($1 \le i \le d$) component of the l-th ($1 \le l \le m$) sample, we can stack this into

$$f_{\mathbf{w}}(\mathbf{x}) = \mathbf{P}\mathbf{w} = \begin{bmatrix} \mathbf{p}_1^\top \mathbf{w} \\ \vdots \\ \mathbf{p}_m^\top \mathbf{w} \end{bmatrix}$$

and

$$\mathbf{p}_l^\top \mathbf{w} = \begin{bmatrix} 1 & x_{l,1} & \ldots & x_{l,d} & \ldots & x_{l,i}x_{l,j} & \ldots & x_{l,i}x_{l,j}x_{l,k} & \ldots \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \\ \vdots \\ w_{i,j} \\ \vdots \\ w_{i,j,k} \\ \vdots \end{bmatrix}$$

Polynomial Regression

- Note that the polynomial matrix

$$\mathbf{P} = \mathbf{P}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m) = \begin{bmatrix} - \mathbf{p}_1^\top - \\ - \mathbf{p}_2^\top - \\ \vdots \\ - \mathbf{p}_m^\top - \end{bmatrix} \in \mathbb{R}^{m \times \binom{d+p}{p}}$$

  is a function of the data samples $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$.
- For an d-variable, order-p polynomial, the matrix $\mathbf{P}$ is of size $m \times \binom{d+p}{p}$.
- When we do not use a polynomial, then for a d-variable, order-1 polynomial (affine model), $\mathbf{P}$ is of size $m \times \binom{d+1}{1} = m \times (d+1)$.
- Offset term $w_0 = b$ is automatically taken into account in an order-1 polynomial.

- Data are

$$\mathbf{x}_1 = \begin{bmatrix}+1 & +1\end{bmatrix}^\top \quad \mathbf{x}_2 = \begin{bmatrix}-1 & +1\end{bmatrix}^\top \quad \mathbf{x}_3 = \begin{bmatrix}+1 & -1\end{bmatrix}^\top \quad \mathbf{x}_4 = \begin{bmatrix}-1 & -1\end{bmatrix}^\top$$

  and $y_1 = y_4 = +1$, $y_2 = y_3 = -1$.

- Second-order polynomial in $d = 2$ variables

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + w_{1,2} x_1 x_2 + w_{1,1} x_1^2 + w_{2,2} x_2^2 = \mathbf{p}^\top \mathbf{w}$$

  where

$$\mathbf{w} = \begin{bmatrix} w_0 & w_1 & w_2 & w_{1,2} & w_{1,1} & w_{2,2} \end{bmatrix}$$
$$\mathbf{p} = \begin{bmatrix} 1 & x_1 & x_2 & x_1 x_2 & x_1^2 & x_2^2 \end{bmatrix}$$

- Can stack the 4 training samples into the polynomial matrix

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

- Notice that the magenta column perfectly distinguishes the training points. [Python Demo]

Summary of Polynomial Regression

- Ridge regression in primal form (when $m > d' = \binom{p+d}{p}$)
  - Learning/Training:

$$\mathbf{w}^* = (\mathbf{P}^\top \mathbf{P} + \lambda \mathbf{I})^{-1} \mathbf{P}^\top \mathbf{y}$$

  - Prediction/Testing: Given a new sample $\mathbf{x}_{\text{new}}$

$$\hat{y}_{\text{new}} = \mathbf{p}_{\text{new}}^\top \mathbf{w}^*$$

  where $\mathbf{p}_{\text{new}}$ is the polynomial vector associated to $\mathbf{x}_{\text{new}}$.

- Ridge regression in dual form (when $m < d' = \binom{p+d}{p}$)
  - Learning/Training:

$$\mathbf{w}^* = \mathbf{P}^\top (\mathbf{P}\mathbf{P}^\top + \lambda \mathbf{I})^{-1} \mathbf{y}$$

  - Prediction/Testing: Given a new sample $\mathbf{x}_{\text{new}}$

$$\hat{y}_{\text{new}} = \mathbf{p}_{\text{new}}^\top \mathbf{w}^*.$$

Summary of Polynomial Regression

- For regression applications:
  - Learn continuous-valued y by using either primal or dual forms
  - Prediction:

$$\hat{y}_{\text{new}} = \mathbf{p}_{\text{new}}^{\top} \mathbf{w}^{*}.$$

- For classification applications:
  - Learn discrete-valued $y \in \{-1, +1\}$ (for binary classification) or one-hot encoded $\mathbf{Y}$ (for $y \in \{1, 2, \ldots, C\}$ for multi-class classification) using either primal or dual forms
  - Binary prediction

$$\hat{y}_{\text{new}} = \operatorname{sign}\left(\mathbf{p}_{\text{new}}^{\top} \mathbf{w}^{*}\right)$$

  - Multi-class prediction

$$\hat{y}_{\text{new}} = \underset{k \in \{1,2,\ldots,C\}}{\arg\max} \left(\mathbf{p}_{\text{new}}^{\top} \mathbf{W}^{*}[:, k]\right)$$

The XOR Example Revisited

- We can compute the weight vector (with $\lambda = 0$)

$$\mathbf{w}^* = \mathbf{P}^\top(\mathbf{P}\mathbf{P}^\top)^{-1}\mathbf{y} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Recall that

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}.$$

- Note that $\mathbf{w}^*$ picks out the coefficient $w_{1,2}$ corresponding x_1x_2.

The XOR Example Revisited

- Given a new test sample $\mathbf{x}_{\text{new}} = \begin{bmatrix} 0.2 & 0.5 \end{bmatrix}^\top$, the polynomial vector associated to $\mathbf{x}_{\text{new}}$ is

$$\begin{aligned}\mathbf{p}_{\text{new}} &= \begin{bmatrix} 1 & x_{\text{new},1} & x_{\text{new},2} & x_{\text{new},1}x_{\text{new},2} & x_{\text{new},1}^2 & x_{\text{new},2}^2 \end{bmatrix}^\top \\ &= \begin{bmatrix} 1 & 0.2 & 0.5 & 0.1 & 0.04 & 0.25 \end{bmatrix}^\top\end{aligned}$$

- Its predicted label is

$$\begin{aligned}\hat{y}_{\text{new}} &= \text{sign}\left(\mathbf{p}_{\text{new}}^\top \mathbf{w}^*\right) \\ &= \text{sign}(0 \times 1 + 0 \times 0.2 + 0 \times 0.5 + 1 \times 0.1 + 0 \times 0.04 + 0 \times 0.25) \\ &= 1.\end{aligned}$$

- Intuitively this is because the product of $\mathbf{x}_{\text{new}}$'s coordinates is positive. [Python Demo]

Python Demo 3: Training/Learning

```
#EE2211 Lecture 6 Demo 3 Training/Learning
import numpy as np
from numpy.linalg import inv
from numpy.linalg import matrix_rank
from sklearn.preprocessing import PolynomialFeatures
X = np.array([ [1, 1], [-1, 1], [1, -1], [-1, -1]])
y = np.array([[1], [-1], [-1], [1]])
## Generate polynomial features
order = 2
poly = PolynomialFeatures(order)
print(poly)
P = poly.fit_transform(X)
print("matrix P")
print(P)
print("Under-determined system")
#print(matrix_rank(P))
#PY = np.vstack((P.T, y.T))
#print(matrix_rank(PY.T))

## dual solution m < d (without ridge)
w_dual = P.T @ inv(P @ P.T) @ y
print("Unique constrained solution, no ridge")
print(w_dual)
```

Python Demo 3: Prediction/Testing

```
#testing
print("Prediction")
#Xnew= np.array([ [0.2, 0.5]])
# Two test points
Xnew= np.array([ [0.2, 0.5], [-0.9, 0.7]])
Pnew = poly.fit_transform(Xnew)
Ynew=Pnew@w_dual
print(Ynew)
print(np.sign(Ynew))
```

Summary

- Primal Form
  - Learning/Training

$$\mathbf{w}^* = (\mathbf{P}^\top \mathbf{P} + \lambda \mathbf{I})^{-1} \mathbf{P} \mathbf{y}$$

  - Prediction/Testing

$$\hat{y}_{\text{new}} = \mathbf{p}_{\text{new}}^\top \mathbf{w}^*$$

- Dual Form
  - Learning/Training

$$\mathbf{w}^* = \mathbf{P}^\top (\mathbf{P} \mathbf{P}^\top + \lambda \mathbf{I})^{-1} \mathbf{y}$$

  - Prediction/Testing

$$\hat{y}_{\text{new}} = \mathbf{p}_{\text{new}}^\top \mathbf{w}^*$$

- Useful Python packages and functions

`sklearn.preprocessing PolynomialFeatures`, `np.sign`,
`sklearn.model_selection train_test_split`,
`sklearn.preprocessing OneHotEncoder`