THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

DSAA2011 Machine Learning

L2 Supervised Learning: Regression and Classification I

Dr. Zixin Zhong

Data Science and Analytics Thrust
Information Hub
Hong Kong University of Sceience and Technology (Guangzhou)

February 18, 2025

Syllabus

Week #	Topic	Lecturer
1	Introduction + course Info	Zixin
2-3	**Supervised learning: regression and classification**	Zixin
4	Model evaluation and choice	Zixin
5	Feature selection	Zixin
6	Boosting methods	Weikai
	Midterm-29 March (Sat): save your day and mark it on calendar!	
7-8	Unsupervised learning: clustering	Weikai
9	Active learning	Weikai
10-11	Markov and graphical models	Weikai
12-13	Online learning	Zixin
14	Final exam	

Exercise in lab

- You may submit your solution set via Canvas: ‘Assignments’ > ‘Lab note’.
  - We may randomly select some and provide feedback.
  - Your grade for this course will not be affected.

Recap of Lecture 1

- Definition and taxonomy of machine learning
- Mathematical tools

Taxonomy of Machine Learning (A Simplistic View)

- What type of data?
  - **Supervised learning** - labeled data: e.g., prediction
  - Semi-supervised learning
  - **Unsupervised learning** - unlabeled data: e.g., clustering
  - **Reinforcement learning** - environment feedback: e.g., multi-armed bandit
- When do we collect data?
  - **Offline learning**
  - **Online learning**

Mathematical tools

- Set and function
  - Set
  - Function: differentiable, (strictly) convex/concave, linear and affine
  - Local and global extrema, partial derivative and gradient vector

Mathematical tools

- Set and function
  - Set
  - Function: differentiable, (strictly) convex/concave, linear and affine
  - Local and global extrema, partial derivative and gradient vector
    - ★ $f(x)$ is differentiable and (strictly)convex: x_0 such that $f'(x_0) = 0$ is the (unique) minimizer of function f.
    - ★ $f(x)$ is differentiable and (strictly) concave: x_0 such that $f'(x_0) = 0$ is the (unique) maximizer of function f.

Mathematical tools

- Set and function
  - Set
  - Function: differentiable, (strictly) convex/concave, linear and affine
  - Local and global extrema, partial derivative and gradient vector
- Probability and estimation
  - Discrete and continuous random variables, event
  - Sum rule and product rule
  - Bayes' rule
  - Independence, expectation and variance
  - **Maximum likelihood estimation (MLE) (unformal):**
    Find the parameter set that maximizes the probability the given dataset is collected
  - Exponential distribution: memory-less

Bayes' rule

$$p(y|x) = \frac{p(x|y)p(y)}{\sum_{y' \in \mathcal{Y}} p(x|y')p(y')} \propto p(x|y)p(y) \text{ (when } x \text{ is fixed)}$$

- **Prior:** beliefs or knowledge about y before observing the data x
- **Likelihood:** how likely the observed x is, given a particular value of y
- **Posterior:** updated belief about y after incorporating the prior and the likelihood of the observed data

Function of random variable

If g is a function from the domain of X to $\mathbb{R}$, we can obtain the expectation of $Y = g(X)$ in the same way.

Proof. I. Discrete case:

$$\mathbb{E}Y = \mathbb{E}[g(X)] = \sum_{y} y p_Y(y) = \sum_{i=1}^{m} g(x) p_X(x).$$

Function of random variable

If g is a function from the domain of X to $\mathbb{R}$, we can obtain the expectation of $Y = g(X)$ in the same way.

Proof. I. Discrete case:

$$\mathbb{E}Y = \mathbb{E}[g(X)] = \sum_{y} y p_Y(y) = \sum_{i=1}^{m} g(x) p_X(x).$$

can be shown as below:

$$\mathbb{E}[g(X)] = \sum_{i=1}^{m} g(x) p_X(x) = \sum_{y} \sum_{x:g(x)=y} g(x) \Pr(X = x)$$

Function of random variable

If g is a function from the domain of X to $\mathbb{R}$, we can obtain the expectation of $Y = g(X)$ in the same way.

Proof. I. Discrete case:

$$\mathbb{E}Y = \mathbb{E}[g(X)] = \sum_{y} y p_Y(y) = \sum_{i=1}^{m} g(x) p_X(x).$$

can be shown as below:

$$\begin{aligned}\mathbb{E}[g(X)] &= \sum_{i=1}^{m} g(x) p_X(x) = \sum_{y} \sum_{x:g(x)=y} g(x) \Pr(X = x) \\ &= \sum_{y} \sum_{x:g(x)=y} y \Pr(X = x) = \sum_{y} \sum_{x:g(x)=y} y \Pr(Y = y, X = x)\end{aligned}$$

Function of random variable

If g is a function from the domain of X to $\mathbb{R}$, we can obtain the expectation of $Y = g(X)$ in the same way.

Proof. I. Discrete case:

$$\mathbb{E}Y = \mathbb{E}[g(X)] = \sum_y y p_Y(y) = \sum_{i=1}^{m} g(x) p_X(x).$$

can be shown as below:

$$\begin{aligned}
\mathbb{E}[g(X)] &= \sum_{i=1}^{m} g(x) p_X(x) = \sum_y \sum_{x:g(x)=y} g(x) \Pr(X = x) \\
&= \sum_y \sum_{x:g(x)=y} y \Pr(X = x) = \sum_y \sum_{x:g(x)=y} y \Pr(Y = y, X = x) \\
&= \sum_y y \sum_{x:g(x)=y} \Pr(Y = y, X = x) = \sum_y y \Pr(Y = y) = \sum_y y p_Y(y) \mathbb{E}Y.
\end{aligned}$$

Function of random variable

Proof (continued). II. Continuous case:

$$\mathbb{E}Y = \mathbb{E}[g(X)] = \int_{\mathbb{R}} y f_Y(y)\mathrm{d}y = \int_{\mathbb{R}} g(x) f_X(x)\mathrm{d}x.$$

Refer to Theorem 4.1.1 in the book ' Probability and Statistics'.

Function of random variable

If g is a function from the domain of X to $\mathbb{R}$, we can obtain the expectation of $Y = g(X)$ in the same way.

$$\text{(Discrete case)} \qquad \mathbb{E}Y = \mathbb{E}[g(X)] = \sum_{y} y p_Y(y) = \sum_{i=1}^{m} g(x) p_X(x),$$

$$\text{(Continuous case)} \quad \mathbb{E}Y = \mathbb{E}[g(X)] = \int_{\mathbb{R}} y f_Y(y) \mathrm{d}y = \int_{\mathbb{R}} g(x) f_X(x) \mathrm{d}x.$$

Maximum likelihood estimation (MLE)

$$\hat{\theta}_{\mathrm{ML}} = \underset{\theta\in(0,1)}{\arg\max}\ \underbrace{p_{X_1,X_2,\ldots,X_m}(X_1,X_2,\ldots,X_m;\theta)}_{\text{Likelihood}}$$

$$= \underset{\theta\in(0,1)}{\arg\max}\ \underbrace{\log p_{X_1,X_2,\ldots,X_m}(X_1,X_2,\ldots,X_m;\theta)}_{\text{Log likelihood}}$$

$$= \underset{\theta\in(0,1)}{\arg\max} \prod_{i=1}^{m} p_X(X_i;\theta)\ \text{(when } X_i\text{'s are independent)}$$

- Likelihood: probability of observing the dataset $\mathcal{D}$ given a set of parameters θ
- Maximum: seeks the set of parameters θ that maximizes this likelihood function, i.e., makes the observed dataset as "likely" as possible under the model
- Concept of MLE: linear regression, logistic regression, ...

Maximum likelihood estimation (MLE)

- **Consistency.** as the sample size increases to infinity, the estimator will converge to the true parameter value: $\hat{\theta}_{\mathrm{ML}} \xrightarrow{p} \theta$ as $n \to \infty$.
  - We say $X_n \xrightarrow{p} X$ as $n \to \infty$ if

$$\lim_{n\to\infty} \Pr(|X_n - X| > \varepsilon) = 0 \ \forall \varepsilon > 0.$$

Maximum likelihood estimation (MLE)

- **Consistency.** as the sample size increases to infinity, the estimator will converge to the true parameter value: $\hat{\theta}_{\mathrm{ML}} \xrightarrow{p} \theta$ as $n \to \infty$.
  - We say $X_n \xrightarrow{p} X$ as $n \to \infty$ if

$$\lim_{n\to\infty} \Pr(|X_n - X| > \varepsilon) = 0 \ \forall \varepsilon > 0.$$

- **Unbiasedness.** MLE is not necessarily unbiased, but in certain cases, it can be unbiased.
  - We say an estimator of a given parameter is unbiased if its expected value is equal to the true value of the parameter.

Maximum likelihood estimation (MLE)

- **Efficiency.** MLE is asymptotically efficient in large samples, i.e., it achieves the lowest possible variance among all unbiased estimators.
  - Cramér-Rao Lower Bound (CRLB): theoretical lower bound for the variance of any unbiased estimator.

Maximum likelihood estimation (MLE)

- **Efficiency.** MLE is asymptotically efficient in large samples, i.e., it achieves the lowest possible variance among all unbiased estimators.
  - Cramér-Rao Lower Bound (CRLB): theoretical lower bound for the variance of any unbiased estimator.
- **Asymptotic Normality.** MLE is asymptotically normal: $\sqrt{n}(\hat{\theta}_{\mathrm{ML}}) \xrightarrow{d} \mathcal{N}(0, I(\theta)^{-1})$.
  - $I(\theta)$: Fisher information of θ (larger information $\Longrightarrow$ smaller variance).
  - We say $X_n \xrightarrow{d} X$ as $n \to \infty$ if

$$\lim_{n\to\infty} F_{X_n}(x) = F_X(x)\ \forall x \text{ at which } F_X(x) \text{ is continuous.}$$

Maximum likelihood estimation (MLE)

- **Efficiency.** MLE is asymptotically efficient in large samples, i.e., it achieves the lowest possible variance among all unbiased estimators.
  - Cramér-Rao Lower Bound (CRLB): theoretical lower bound for the variance of any unbiased estimator.
- **Asymptotic Normality.** MLE is asymptotically normal: $\sqrt{n}(\hat{\theta}_{\mathrm{ML}}) \xrightarrow{d} \mathcal{N}(0, I(\theta)^{-1})$.
  - $I(\theta)$: Fisher information of θ (larger information $\Longrightarrow$ smaller variance).
  - We say $X_n \xrightarrow{d} X$ as $n \to \infty$ if

$$\lim_{n\to\infty} F_{X_n}(x) = F_X(x) \ \forall x \text{ at which } F_X(x) \text{ is continuous.}$$

- **Model Sensitivity.** MLE is sensitive to model assumptions, and incorrect assumptions can lead to biased or inconsistent estimates.

Maximum likelihood estimation (MLE)

- **Consistency.** as the sample size increases to infinity, the estimator will converge to the true parameter value: $\hat{\theta}_{\mathrm{ML}} \xrightarrow{p} \theta$ as $n \to \infty$.
- **Unbiasedness.** MLE is not necessarily unbiased, but in certain cases, it can be unbiased.
- **Efficiency.** MLE is asymptotically efficient in large samples, i.e., it achieves the lowest possible variance among all unbiased estimators.
- **Asymptotic Normality.** MLE is asymptotically normal: $\sqrt{n}(\hat{\theta}_{\mathrm{ML}}) \xrightarrow{d} \mathcal{N}(0, I(\theta)^{-1})$.
- **Model Sensitivity.** MLE is sensitive to model assumptions, and incorrect assumptions can lead to biased or inconsistent estimates.

Mathematical tools

- Set and function
  - Set
  - Function: differentiable, (strictly) convex/concave, linear and affine
  - Local and global extrema, partial derivative and gradient vector
- Probability and estimation
  - Discrete and continuous random variables, event
  - Sum rule and product rule
  - Bayes' rule
  - Independence, expectation and variance
  - **Maximum likelihood estimation (MLE) (unformal):**
    Find the parameter set that maximizes the probability the given dataset is collected
  - Exponential distribution: memory-less
- Systems of linear equations
  - Vector: linear independence
  - Matrix: rank, kernel, range, dimension, invertible
  - Solution to linear system: existence and uniqueness, Gaussian-elimination method

Solutions to linear systems

◎ **Method**: apply Guassian-elimination method to the augmented matrix $\tilde{\boldsymbol{X}} = [\boldsymbol{X}\ \ \boldsymbol{y}]$.

$$\boldsymbol{X}\boldsymbol{w} = \boldsymbol{y} \quad \text{or} \quad [\underline{x}_1\ \underline{x}_2\ \ \ldots\ \ \underline{x}_d] \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{bmatrix}, \text{ where } \underline{x}_i = \begin{bmatrix} x_{1,i} \\ x_{2,i} \\ \vdots \\ x_{m,i} \end{bmatrix}. \tag{0.1}$$

Theorem 0.1 (Rouché-Capelli Theorem)

- *The system in* (0.1) *admits a unique solution if and only if* $\mathrm{rank}(\boldsymbol{X}) = \mathrm{rank}(\tilde{\boldsymbol{X}}) = d$;
- *The system in* (0.1) *has no solution if and only if* $\mathrm{rank}(\boldsymbol{X}) < \mathrm{rank}(\tilde{\boldsymbol{X}})$;
- *The system in* (0.1) *has infinitely many solutions if and only if* $\mathrm{rank}(\boldsymbol{X}) = \mathrm{rank}(\tilde{\boldsymbol{X}}) < d$.

Recommended materials for mathematical tools

- Set and function
  - 'The Matrix Cookbook' [http://matrixcookbook.com] by Kaare Brandt Petersen and Michael Syskind Pedersen
- Probability and estimation
  - 'A First Course in Probability' by Sheldon Ross
  - 'Probability and Statistics' by Morris H. DeGroot and Mark J. Schervish
  - 'Probability Theory' by Achim Klenke
- Linear algebra:
  - 'Introduction to Linear Algebra' by Gilbert Strang (`https://math.mit.edu/~gs/linearalgebra/ila5/indexila5.html`)

◦ Found in Canvas ('Modules' > 'Recommended materials') or online

Outline

1. Least squares and linear regression
2. Linear classification
3. Polynomial regression
4. Ridge regression

Outline

1. Least squares and linear regression
2. Linear classification
3. Polynomial regression
4. Ridge regression

Motivation for linear regression

How can we predict our academic performance in the coming semester?

Hours studied

Sleep hours

Extracurricular activities

Previous scores

Motivation for linear regression

- Consider five indicators (data set from Kaggle):
  (a) Hours Studied x_1: total number of hours of study;
  (b) Previous Scores x_2: scores obtained in previous tests;
  (c) Extracurricular Activities x_3: participation in extracurricular activities (Yes/No);
  (d) Sleep Hours x_4: average number of hours of sleep per day;
  (e) Sample question papers practiced x_5: number of sample question papers practiced.
- Which factor is most important for determining the student's performance?

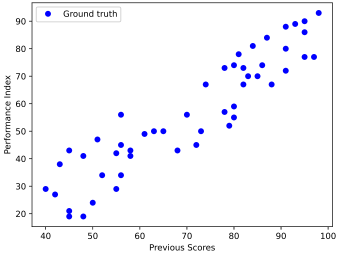
Motivation for linear regression

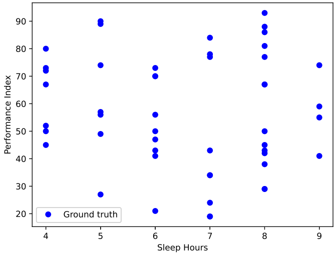
- Consider five indicators (data set from Kaggle):
  (a) Hours Studied x_1: total number of hours of study;
  (b) Previous Scores x_2: scores obtained in previous tests;
  (c) Extracurricular Activities x_3: participation in extracurricular activities (Yes/No);
  (d) Sleep Hours x_4: average number of hours of sleep per day;
  (e) Sample question papers practiced x_5: number of sample question papers practiced.
- Which factor is most important for determining the student's performance?
- Data is of the form

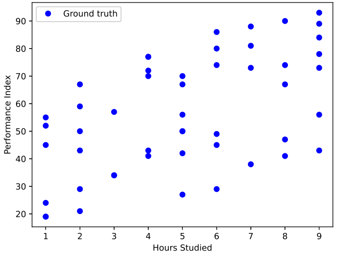
$$\mathbf{x}_i = \begin{bmatrix} x_{i,1} \\ x_{i,2} \\ x_{i,3} \\ x_{i,4} \\ x_{i,5} \end{bmatrix} \quad \text{and} \quad y_i \quad \text{for} \quad i \in \{1, 2, \ldots, 1000\}$$

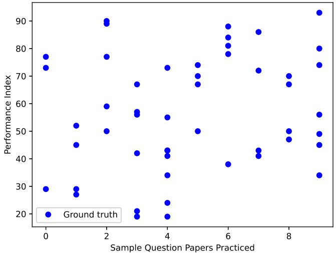
  where $x_{i,1}$ is the number of study hours of student i (etc.) and y_i is the student's performance index.

Motivation for linear regression









Linear regression

- Linear regression is a linear approach for modelling the relationship between a scalar response y and one or more explanatory variables (or attributes, or features) $\mathbf{x}$.
- We have a dataset $\{(\mathbf{x}_i, y_i) : i = 1, \ldots, m\}$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ are the feature vector and target of the i-th sample respectively.
- Without the offset, we can form the design matrix and the target vector

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_m^\top \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,d} \end{bmatrix} \in \mathbb{R}^{m \times d} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \in \mathbb{R}^m$$

- We wish to find $\mathbf{w} \in \mathbb{R}^d$ satisfying (or approximately satisfying) the linear system

$$\mathbf{X}\mathbf{w} = \mathbf{y}.$$

Linear regression (with offset)

- m: size of the dataset
- d: dimension/length of each feature vector (input)
- y_i: scalar or real-valued target/output (e.g., height, exam marks)

Goal:

- Design a function/model/regressor $f_{\mathbf{w},b}$ as a linear combination of the features in $\mathbf{x}$, i.e.,

$$f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b,$$

  where $\mathbf{w} \in \mathbb{R}^d$, the unknown, is the d-dimensional weight vector and b is the bias or offset.
- The notation $f_{\mathbf{w},b}$ means that the model is parametrized by two quantities $\mathbf{w}$ and b.
- Note that the model can also be more compactly written as

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

Objective (loss) function in linear regression

- We wish to minimize the error e_i between the prediction $f_{\mathbf{w},b}(\mathbf{x}_i)$ and the target, where

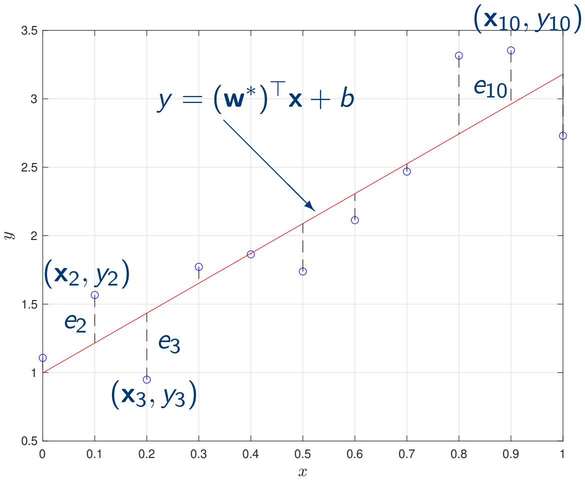
$$e_i = f_{\mathbf{w},b}(\mathbf{x}_i) - y_i.$$

- We average the square of the errors over all training samples. This defines the objective or loss function

$$\mathrm{Loss}(\mathbf{w}, b) = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\mathbf{w},b}(\mathbf{x}_i) - y_i\right)^2.$$

- $\mathrm{Loss}(\mathbf{w}, b)$ is known as the (squared or ℓ_2) **loss** or **objective function**.
- $\left(f_{\mathbf{w},b}(\mathbf{x}_i) - y_i\right)^2$ is also called the per-sample loss or objective function and is a measure of the difference or penalty between the prediction $f_{\mathbf{w},b}(\mathbf{x}_i)$ and the target y_i.

Objective (loss) function in linear regression



- Minimize the sum of squares of the errors e_i, i.e. $\sum_{i=1}^{11} e_i^2$.
- $\mathbf{x}$ is a scalar here, but can be a vector in general.

Objective (loss) function in linear regression

- Define $\overline{\mathbf{w}} \in \mathbb{R}^{d+1}$ as the $(d+1)$-dimensional vector that concatenates b and $\mathbf{w}$, i.e.,

$$\overline{\mathbf{w}} = \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} b \\ w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}.$$

- Similarly, define $\bar{\mathbf{x}}_i \in \mathbb{R}^{d+1}$ as the $(d+1)$-dimensional vector that concatenates 1 and $\mathbf{x}_i$

$$\bar{\mathbf{x}}_i = \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} 1 \\ x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,d} \end{bmatrix}.$$

Objective (loss) function in linear regression

- We wish to find $\overline{\mathbf{w}}^* = [b^*, \mathbf{w}^*]^\top \in \mathbb{R}^{d+1}$ that minimizes

$$\overline{\mathbf{w}}^* = \underset{\overline{\mathbf{w}}=[b,\mathbf{w}]^\top}{\arg\min} \mathrm{Loss}(\mathbf{w}, b)$$

where the ℓ_2 or squared loss is

$$\mathrm{Loss}(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} (f_{\mathbf{w},b}(\mathbf{x}_i) - y_i)^2$$

- The $1/m$ does not affect the solution so we can choose to include or exclude it.

Objective (loss) function in linear regression

- Note that
$$f_{\mathbf{w},b}(\mathbf{x}_i) - y_i = \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix}^\top \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix} - y_i = \overline{\mathbf{x}}_i^\top \overline{\mathbf{w}} - y_i,$$
so that
$$\sum_{i=1}^{m} (f_{\mathbf{w},b}(\mathbf{x}_i) - y_i)^2 = \sum_{i=1}^{m} \left(\overline{\mathbf{x}}_i^\top \overline{\mathbf{w}} - y_i\right)^2.$$
In other words,
$$\sum_{i=1}^{m} (f_{\mathbf{w},b}(\mathbf{x}_i) - y_i)^2 = (\mathbf{X}\overline{\mathbf{w}} - \mathbf{y})^\top (\mathbf{X}\overline{\mathbf{w}} - \mathbf{y})$$
- The design matrix is now the $m \times (d+1)$ matrix
$$\mathbf{X} = \begin{bmatrix} \overline{\mathbf{x}}_1^\top \\ \overline{\mathbf{x}}_2^\top \\ \vdots \\ \overline{\mathbf{x}}_m^\top \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{x}_1^\top \\ 1 & \mathbf{x}_2^\top \\ \vdots & \vdots \\ 1 & \mathbf{x}_m^\top \end{bmatrix} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1} & x_{m,2} & \cdots & x_{m,d} \end{bmatrix} \in \mathbb{R}^{m\times(d+1)}$$

Optimizing the loss function in linear regression

$$\overline{\mathbf{w}} = \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} b \\ w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}. \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \in \mathbb{R}^m$$

- The objective function is now simplified to

$$\begin{aligned} J(\overline{\mathbf{w}}) = (\mathbf{X}\overline{\mathbf{w}} - \mathbf{y})^\top(\mathbf{X}\overline{\mathbf{w}} - \mathbf{y}) &= \overline{\mathbf{w}}^\top\mathbf{X}^\top\mathbf{X}\overline{\mathbf{w}} - \underbrace{\overline{\mathbf{w}}^\top\mathbf{X}^\top\mathbf{y}}_{M} - \underbrace{\mathbf{y}^\top\mathbf{X}\overline{\mathbf{w}}}_{M\text{^}T} + \mathbf{y}^\top\mathbf{y} \\ &= \overline{\mathbf{w}}^\top\mathbf{X}^\top\mathbf{X}\overline{\mathbf{w}} - 2\overline{\mathbf{w}}^\top(\mathbf{X}^\top\mathbf{y}) + \mathbf{y}^\top\mathbf{y} \end{aligned}$$

  The terms in blue are the same. Why? To prove M is symmetric

  Because the size of matrix M is 1×1

- Differentiating this w.r.t. $\overline{\mathbf{w}}$,

$$\nabla_{\overline{\mathbf{w}}} J(\overline{\mathbf{w}}) = 2\mathbf{X}^\top\mathbf{X}\overline{\mathbf{w}} - 2\mathbf{X}^\top\mathbf{y}.$$

- Setting this to zero yields

$$2\mathbf{X}^\top\mathbf{X}\overline{\mathbf{w}}^* = 2\mathbf{X}^\top\mathbf{y}.$$

- If $\mathbf{X}$ has full column rank, $\mathbf{X}^\top\mathbf{X}$ is invertible and

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

This is the least squares solution. Is it a global or local minimum?

Least squares: training and prediction

- In summary, given a dataset $(\mathbf{x}_i, y_i)$ for $i = 1, 2, \ldots, m$, form the design matrix and target vector

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

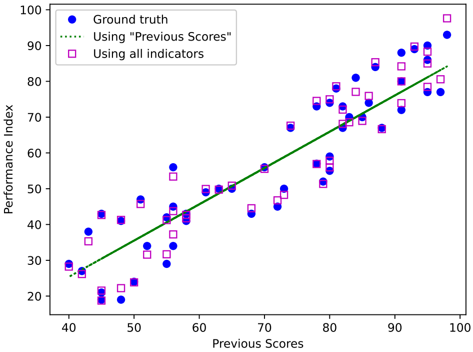
- Training/Learning:

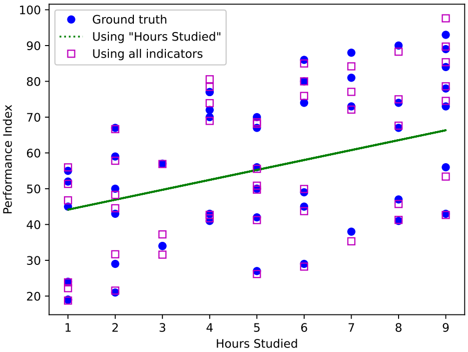
$$\overline{\mathbf{w}}^* = \begin{bmatrix} b^* \\ \mathbf{w}^* \end{bmatrix} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$

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

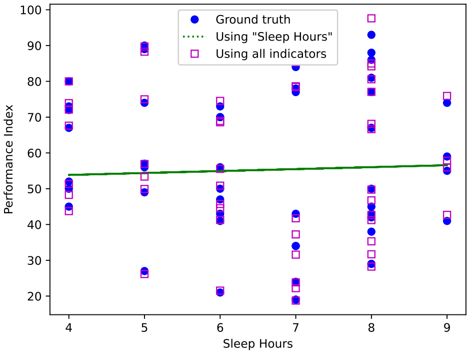
$$\hat{y}_{\mathrm{new}} = \begin{bmatrix} 1 \\ \mathbf{x}_{\mathrm{new}} \end{bmatrix}^\top \overline{\mathbf{w}}^* = b^* + \mathbf{x}_{\mathrm{new}}^\top \mathbf{w}^*.$$

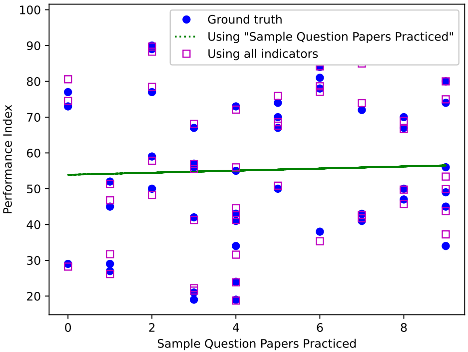
Linear regression: academic performance





Linear regression: academic performance





Linear regression: example 1

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

$$\mathbf{x}_1 = -7, \quad \mathbf{x}_2 = -5, \quad \mathbf{x}_3 = 1, \quad \mathbf{x}_4 = 5$$
$$y_1 = -6, \quad y_2 = -4, \quad y_3 = -1, \quad y_4 = 4$$

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

$$\mathbf{X} = \begin{bmatrix} 1 & -7 \\ 1 & -5 \\ 1 & 1 \\ 1 & 5 \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} -6 \\ -4 \\ -1 \\ 4 \end{bmatrix}$$

Solutions to linear systems

◎ **Method**: apply Guassian-elimination method to the augmented matrix $\tilde{\boldsymbol{X}}$.

$$\boldsymbol{X}\boldsymbol{w} = \boldsymbol{y} \quad \text{or} \quad [\underline{x}_1 \ \underline{x}_2 \ \ldots \ \underline{x}_d] \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{bmatrix}, \text{ where } \underline{x}_i = \begin{bmatrix} x_{1,i} \\ x_{2,i} \\ \vdots \\ x_{m,i} \end{bmatrix}. \tag{0.1}$$

Theorem 1.1 (Rouché-Capelli Theorem)

- *The system in* (0.1) *admits a unique solution if and only if* $\mathrm{rank}(\boldsymbol{X}) = \mathrm{rank}(\tilde{\boldsymbol{X}}) = d$;
- *The system in* (0.1) *has no solution if and only if* $\mathrm{rank}(\boldsymbol{X}) < \mathrm{rank}(\tilde{\boldsymbol{X}})$;
- *The system in* (0.1) *has infinitely many solutions if and only if* $\mathrm{rank}(\boldsymbol{X}) = \mathrm{rank}(\tilde{\boldsymbol{X}}) < d$.

Linear regression: example 1

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

$$\mathbf{x}_1 = -7, \quad \mathbf{x}_2 = -5, \quad \mathbf{x}_3 = 1, \quad \mathbf{x}_4 = 5$$
$$y_1 = -6, \quad y_2 = -4, \quad y_3 = -1, \quad y_4 = 4$$

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

$$\mathbf{X} = \begin{bmatrix} 1 & -7 \\ 1 & -5 \\ 1 & 1 \\ 1 & 5 \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} -6 \\ -4 \\ -1 \\ 4 \end{bmatrix}$$

⊙ Exercise: Is there a solution to the linear system $\mathbf{X}\overline{\mathbf{w}} = \mathbf{y}$? How many?

Linear regression: example 1

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

$$\mathbf{x}_1 = -7, \quad \mathbf{x}_2 = -5, \quad \mathbf{x}_3 = 1, \quad \mathbf{x}_4 = 5$$
$$y_1 = -6, \quad y_2 = -4, \quad y_3 = -1, \quad y_4 = 4$$

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

$$\mathbf{X} = \begin{bmatrix} 1 & -7 \\ 1 & -5 \\ 1 & 1 \\ 1 & 5 \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} -6 \\ -4 \\ -1 \\ 4 \end{bmatrix}$$

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

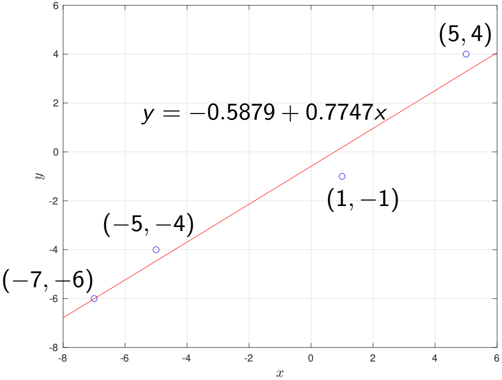
$$\mathrm{rank}(\mathbf{X}) < \mathrm{rank}(\tilde{\mathbf{X}}) \text{ where } \tilde{\mathbf{X}} = [\mathbf{X} \ y].$$

Linear regression: example 1 (training)

- Using some numerical software, we can find

$$\overline{\mathbf{w}}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \begin{bmatrix} -0.5879 \\ 0.7747 \end{bmatrix}$$

- We can plot the points and the least squares line.

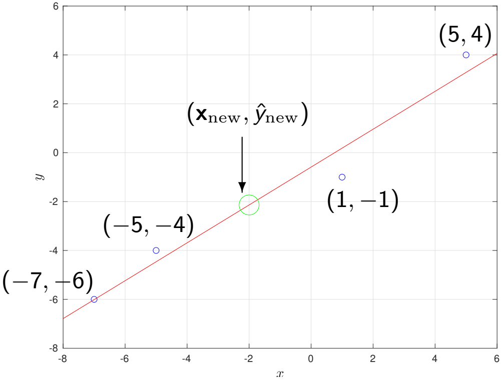


Linear regression: example 1 (prediction)

- Suppose we want to predict the value of y_{new} when $\mathbf{x}_{\text{new}} = -2$. Then we plug $\mathbf{x}_{\text{new}} = -2$ into model to get

$$\hat{y}_{\text{new}} = \begin{bmatrix} 1 \\ \mathbf{x}_{\text{new}} \end{bmatrix}^\top \overline{\mathbf{w}}^* = 1 \times (-0.5879) + (-2) \times (0.7747) = -2.1374$$

- Pictorially,



Linear regression: example 2

- Now our feature vectors are

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{x}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad \mathbf{x}_3 = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad \mathbf{x}_4 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

  and targets are

$$y_1 = 1 \quad y_2 = 0 \quad y_3 = 2 \quad y_4 = -1.$$

- The design matrix and target vector are

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 2 \\ -1 \end{bmatrix}.$$

⊙ Exercise: Is there a solution to the linear system $\mathbf{X}\overline{\mathbf{w}} = \mathbf{y}$? How many?

Solutions to linear systems

◎ **Method**: apply Guassian-elimination method to the augmented matrix $\tilde{\boldsymbol{X}}$.

$$\boldsymbol{X}\boldsymbol{w} = \boldsymbol{y} \quad \text{or} \quad [\underline{x}_1 \ \underline{x}_2 \ \dots \ \underline{x}_d] \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{bmatrix}, \text{ where } \underline{x}_i = \begin{bmatrix} x_{1,i} \\ x_{2,i} \\ \vdots \\ x_{m,i} \end{bmatrix}. \tag{0.1}$$

Theorem 1.2 (Rouché-Capelli Theorem)

- *The system in* (0.1) *admits a unique solution if and only if* $\mathrm{rank}(\boldsymbol{X}) = \mathrm{rank}(\tilde{\boldsymbol{X}}) = d$;
- *The system in* (0.1) *has no solution if and only if* $\mathrm{rank}(\boldsymbol{X}) < \mathrm{rank}(\tilde{\boldsymbol{X}})$;
- *The system in* (0.1) *has infinitely many solutions if and only if* $\mathrm{rank}(\boldsymbol{X}) = \mathrm{rank}(\tilde{\boldsymbol{X}}) < d$.

Linear regression: example 2

- Now our feature vectors are

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{x}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad \mathbf{x}_3 = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad \mathbf{x}_4 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

  and targets are

$$y_1 = 1 \quad y_2 = 0 \quad y_3 = 2 \quad y_4 = -1.$$

- The design matrix and target vector are

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 2 \\ -1 \end{bmatrix}.$$

- (Answer) Note that $3 = \operatorname{rank}(\mathbf{X}) < \operatorname{rank}(\tilde{\mathbf{X}}) = 4$ so the overdetermined system does not have a solution.

Linear regression: example 2 (training & prediction)

- But we can check that $\mathbf{X}$ has full column rank and so the least squares solution exists and is given by

$$\overline{\mathbf{w}}^* = \begin{bmatrix} b^* \\ \mathbf{w}^* \end{bmatrix} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \begin{bmatrix} -0.7500 \\ 0.1786 \\ 0.9286 \end{bmatrix}$$

This is the training or learning step.

Linear regression: example 2 (training & prediction)

- But we can check that $\mathbf{X}$ has full column rank and so the least squares solution exists and is given by

$$\overline{\mathbf{w}}^* = \begin{bmatrix} b^* \\ \mathbf{w}^* \end{bmatrix} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \begin{bmatrix} -0.7500 \\ 0.1786 \\ 0.9286 \end{bmatrix}$$

  This is the training or learning step.

- If we want to make predictions for $\mathbf{x}_{\text{new}} = [0, -1]^\top$, we use the model

$$\hat{y}_{\text{new}} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}^\top \overline{\mathbf{w}}^* = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}^\top \begin{bmatrix} -0.7500 \\ 0.1786 \\ 0.9286 \end{bmatrix} = -1.6786.$$

  This is the prediction step.

Learning vector-valued linear functions

- Suppose there are h outputs we want to predict (above $h = 3$).

Learning vector-valued linear functions

- 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

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

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

Learning vector-valued linear functions: objective

- Our loss function is a generalization of the previous study

$$\mathrm{Loss}(\overline{\mathbf{W}}) = \mathrm{Loss}(\mathbf{W}, \mathbf{b}) = \sum_{k=1}^{h} (\mathbf{X}\overline{\mathbf{w}}_k - \mathbf{y}^{(k)})^\top (\mathbf{X}\overline{\mathbf{w}}_k - \mathbf{y}^{(k)})$$

where for each $1 \le k \le h$,

$$\overline{\mathbf{w}}_k = \begin{bmatrix} b_k \\ w_{1,k} \\ \vdots \\ w_{d,k} \end{bmatrix} \in \mathbb{R}^{d+1} \quad \text{and} \quad \mathbf{y}^{(k)} = \begin{bmatrix} y_{1,k} \\ y_{2,k} \\ \vdots \\ y_{m,k} \end{bmatrix} \in \mathbb{R}^m$$

are the k-th columns of $\mathbf{W}$ and $\mathbf{Y}$ respectively.

Learning vector-valued linear functions: objective

- Our loss function is a generalization of the previous study

$$\mathrm{Loss}(\overline{\mathbf{W}}) = \mathrm{Loss}(\mathbf{W}, \mathbf{b}) = \sum_{k=1}^{h} (\mathbf{X}\overline{\mathbf{w}}_k - \mathbf{y}^{(k)})^\top (\mathbf{X}\overline{\mathbf{w}}_k - \mathbf{y}^{(k)})$$

  where for each $1 \leq k \leq h$,

$$\overline{\mathbf{w}}_k = \begin{bmatrix} b_k \\ w_{1,k} \\ \vdots \\ w_{d,k} \end{bmatrix} \in \mathbb{R}^{d+1} \quad \text{and} \quad \mathbf{y}^{(k)} = \begin{bmatrix} y_{1,k} \\ y_{2,k} \\ \vdots \\ y_{m,k} \end{bmatrix} \in \mathbb{R}^m$$

  are the k-th columns of $\mathbf{W}$ and $\mathbf{Y}$ respectively.
- We are aggregating or summing the contributions of the errors from each of the h prediction tasks.
- Our goal is to find

$$\overline{\mathbf{W}}^* = \arg\min_{\mathbf{W}, \mathbf{b}} \mathrm{Loss}(\overline{\mathbf{W}}) \quad \text{where} \quad \overline{\mathbf{W}} = \begin{bmatrix} \mathbf{b}^\top \\ \mathbf{W} \end{bmatrix}.$$

Learning vector-valued linear functions: training

- Objective:

$$\overline{\mathbf{W}}^* = \underset{\mathbf{W},\mathbf{b}}{\arg\min}\, \mathrm{Loss}(\overline{\mathbf{W}}) \quad \text{where} \quad \overline{\mathbf{W}} = \begin{bmatrix} \mathbf{b}^\top \\ \mathbf{W} \end{bmatrix}.$$

- By differentiating with respect to each column $\overline{\mathbf{w}}_k$ and setting the result to zero, we find that the least squares solution is

$$\overline{\mathbf{W}}^* = \begin{bmatrix} \overline{\mathbf{w}}_1^* & \overline{\mathbf{w}}_2^* & \dots & \overline{\mathbf{w}}_h^* \end{bmatrix} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} \in \mathbb{R}^{(d+1)\times h}.$$

  This may be an exercise in a tutorial.
- In this new setting, what condition does $\mathbf{X}$ have to satisfy for $\overline{\mathbf{W}}^*$ to exist?

Learning vector-valued linear functions: training

- Objective:

$$\overline{\mathbf{W}}^* = \arg\min_{\mathbf{W},\mathbf{b}} \mathrm{Loss}(\overline{\mathbf{W}}) \quad \text{where} \quad \overline{\mathbf{W}} = \begin{bmatrix} \mathbf{b}^\top \\ \mathbf{W} \end{bmatrix}.$$

- By differentiating with respect to each column $\overline{\mathbf{w}}_k$ and setting the result to zero, we find that the least squares solution is

$$\overline{\mathbf{W}}^* = \begin{bmatrix} \overline{\mathbf{w}}_1^* & \overline{\mathbf{w}}_2^* & \ldots & \overline{\mathbf{w}}_h^* \end{bmatrix} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y} \in \mathbb{R}^{(d+1)\times h}.$$

  This may be an exercise in a tutorial.
- In this new setting, what condition does $\mathbf{X}$ have to satisfy for $\overline{\mathbf{W}}^*$ to exist?
- We need $(\mathbf{X}^\top\mathbf{X})^{-1}$ to exist, which means that $\mathbf{X}$ has to have full column rank.

Learning vector-valued linear functions: prediction

- Given a dataset $\{(\mathbf{x}_i, \mathbf{y}_i)\}$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $\mathbf{y}_i \in \mathbb{R}^{1\times h}$ and $1 \le i \le m$, we can use the above procedure to learn the least squares solution

$$\overline{\mathbf{W}}^* = \begin{bmatrix} \overline{\mathbf{w}}_1^* & \overline{\mathbf{w}}_2^* & \dots & \overline{\mathbf{w}}_h^* \end{bmatrix} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} \in \mathbb{R}^{(d+1)\times h}.$$

- Given a new sample $\mathbf{x}_{\mathrm{new}} \in \mathbb{R}^d$, the predictions are contained in the row vector

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

Linear regression: example 3

- Now our feature vectors are

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{x}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad \mathbf{x}_3 = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad \mathbf{x}_4 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

  and targets are

$$\mathbf{y}_1 = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad \mathbf{y}_2 = \begin{bmatrix} 0 & 1 \end{bmatrix} \quad \mathbf{y}_3 = \begin{bmatrix} 2 & -1 \end{bmatrix} \quad \mathbf{y}_4 = \begin{bmatrix} -1 & 3 \end{bmatrix}$$

- Here, $m = 4$, $d = 2$, $h = 2$.
- The design matrix and target matrix are

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{Y} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 2 & -1 \\ -1 & 3 \end{bmatrix}.$$

- Note that the first regression problem here (corresponding to the first components of each $\mathbf{y}_i$) is exactly the same as that in Linear Regression Example 2 on Slide 41.

Linear regression: example 4 (training & prediction)

- We have already checked that $\mathbf{X}$ has full column rank. Hence, the least squares solution is

$$\overline{\mathbf{W}}^{*} = (\mathbf{X}^{\top}\mathbf{X})^{-1}\mathbf{X}^{\top}\mathbf{Y} = \begin{bmatrix} -0.7500 & 2.2500 \\ 0.1786 & 0.0357 \\ 0.9286 & 1.2143 \end{bmatrix} \in \mathbb{R}^{(d+1)\times h}.$$

Linear regression: example 4 (training & prediction)

- We have already checked that $\mathbf{X}$ has full column rank. Hence, the least squares solution is

$$\overline{\mathbf{W}}^* = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{Y} = \begin{bmatrix} -0.7500 & 2.2500 \\ 0.1786 & 0.0357 \\ 0.9286 & 1.2143 \end{bmatrix} \in \mathbb{R}^{(d+1)\times h}.$$

- Now, someone gave us a new sample $\mathbf{x}_{\mathrm{new}} = [0, -1]^\top$. The predicted output is

$$\hat{\mathbf{y}}_{\mathrm{new}} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}^\top \overline{\mathbf{W}}^* = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}^\top \begin{bmatrix} -0.7500 & 2.2500 \\ 0.1786 & 0.0357 \\ 0.9286 & 1.2143 \end{bmatrix} = \begin{bmatrix} -1.6786 & 3.4643 \end{bmatrix}$$

  The first prediction -1.6786 corresponds to that in Linear Regression Example 2 on Slide 42.

- This is the prediction step.

Summary

- (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

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

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

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

Linear regression

Any question about the linear regression model?

`Supervised learning`

Objective (loss) function in linear regression

$$\mathrm{Loss}(\mathbf{w}, b) = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\mathbf{w},b}(\mathbf{x}_i) - y_i\right)^2 = \frac{1}{m}\sum_{i=1}^{m}\left(\mathbf{w}^\top x_i + b - y_i\right)^2.$$

Q: Why?

Objective (loss) function in linear regression

$$\mathrm{Loss}(\mathbf{w}, b) = \frac{1}{m}\sum_{i=1}^{m}(f_{\mathbf{w},b}(\mathbf{x}_i) - y_i)^2 = \frac{1}{m}\sum_{i=1}^{m}\left(\mathbf{w}^\top x_i + b - y_i\right)^2.$$

Q: Why?

A: Assumption

$$y_i = \mathbf{w}^\top x_i + b + e_i$$

- y_i: dependent variable (target).
- $\mathbf{x}_i$: matrix of independent variables (design matrix/features).
- $\mathbf{W}$: vector of coefficients (parameters) that we want to estimate.
- e_i: error term,

Objective (loss) function in linear regression

$$\mathrm{Loss}(\mathbf{w}, b) = \frac{1}{m}\sum_{i=1}^{m}(f_{\mathbf{w},b}(\mathbf{x}_i) - y_i)^2 = \frac{1}{m}\sum_{i=1}^{m}\left(\mathbf{w}^\top x_i + b - y_i\right)^2.$$

Q: Why?

A: Assumption

$$y_i = \mathbf{w}^\top x_i + b + e_i$$

- y_i: dependent variable (target).
- $\mathbf{x}_i$: matrix of independent variables (design matrix/features).
- $\mathbf{W}$: vector of coefficients (parameters) that we want to estimate.
- e_i: error term, usually assumed to be normally distributed, i.e., $e_i \sim \mathcal{N}(0, \sigma^2)$.

MLE and linear regression

- Given that $y_i = \mathbf{w}^\top \mathbf{x}_i + b + e_i$ for each data point i, and assuming $e_i \sim \mathcal{N}\left(0, \sigma^2\right)$

MLE and linear regression

- Given that $y_i = \mathbf{w}^\top \mathbf{x}_i + b + e_i$ for each data point i, and assuming $e_i \sim \mathcal{N}\left(0, \sigma^2\right)$
- Probability density function (PDF) of y_i given x_i is

$$p\left(y_i \mid \mathbf{x}_i; \mathbf{W}, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(y_i - \mathbf{W}^\top \mathbf{x}_i\right)^2}{2\sigma^2}\right)$$

MLE and linear regression

- Given that $y_i = \mathbf{w}^\top \mathbf{x}_i + b + e_i$ for each data point i, and assuming $e_i \sim \mathcal{N}\left(0, \sigma^2\right)$
- Probability density function (PDF) of y_i given x_i is

$$p\left(y_i \mid \mathbf{x}_i; \mathbf{W}, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(y_i - \mathbf{W}^\top \mathbf{x}_i\right)^2}{2\sigma^2}\right)$$

- Likelihood function for the entire dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ is

$$L\left(\mathbf{W}, \sigma^2 \mid \{y_i, \mathbf{x}_i\}\right) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(y_i - \mathbf{W}^\top \mathbf{x}_i\right)^2}{2\sigma^2}\right)$$

MLE and linear regression

- Given that $y_i = \mathbf{w}^\top \mathbf{x}_i + b + e_i$ for each data point i, and assuming $e_i \sim \mathcal{N}(0, \sigma^2)$
- Probability density function (PDF) of y_i given x_i is

$$p\left(y_i \mid \mathbf{x}_i; \mathbf{W}, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(y_i - \mathbf{W}^\top \mathbf{x}_i\right)^2}{2\sigma^2}\right)$$

- Likelihood function for the entire dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ is

$$L\left(\mathbf{W}, \sigma^2 \mid \{y_i, \mathbf{x}_i\}\right) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(y_i - \mathbf{W}^\top \mathbf{x}_i\right)^2}{2\sigma^2}\right)$$

- Log-Likelihood Function

$$\log L\left(\mathbf{W}, \sigma^2 \mid \{y_i, \mathbf{x}_i\}\right) = -\frac{n}{2} \log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_i - \mathbf{W}^\top \mathbf{x}_i\right)^2$$

MLE and linear regression

- Maximizing the Log-Likelihood: take the derivative of the log-likelihood function with respect to $\mathbf{W}$ and set it equal to zero:

$$\frac{\partial}{\partial \mathbf{W}} \log L\left(\mathbf{W}, \sigma^2 \mid \{y_i, \mathbf{x}_i\}\right) = \frac{1}{\sigma^2} \sum_{i=1}^{n} \left(y_i - \mathbf{W}^\top \mathbf{x}_i\right) \mathbf{x}_i = 0$$

MLE and linear regression

- Maximizing the Log-Likelihood: take the derivative of the log-likelihood function with respect to $\mathbf{W}$ and set it equal to zero:

$$\frac{\partial}{\partial \mathbf{W}} \log L\left(\mathbf{W}, \sigma^2 \mid \{y_i, \mathbf{x}_i\}\right) = \frac{1}{\sigma^2} \sum_{i=1}^{n} \left(y_i - \mathbf{W}^\top \mathbf{x}_i\right) \mathbf{x}_i = 0$$

- If $\mathbf{X}$ has full column rank, $\mathbf{X}^\top \mathbf{X}$ is invertible and

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

is the least squares solution.

Outline

Linear models for classification

- **Main idea:** to treat binary classification as regression where each label y_i can only take on -1 or $+1$.
- If in testing/prediction, $\overline{\mathbf{x}}_{\mathrm{new}}^{\top}\overline{\mathbf{w}}^{*}$ is positive (resp. negative), predict that $\hat{y}_{\mathrm{new}} = +1$ (resp. $\hat{y}_{\mathrm{new}} = -1$). For example, distinguishing between cats and dogs.

Linear models for classification

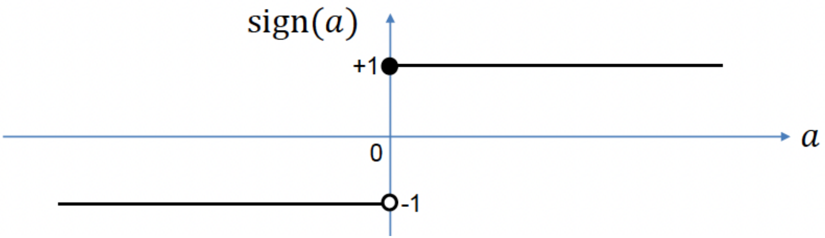
- **Main idea:** to treat binary classification as regression where each label y_i can only take on -1 or $+1$.
- If in testing/prediction, $\overline{\mathbf{x}}_{\text{new}}^\top \overline{\mathbf{w}}^*$ is positive (resp. negative), predict that $\hat{y}_{\text{new}} = +1$ (resp. $\hat{y}_{\text{new}} = -1$). 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

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

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

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

The sign function



For example,

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

$$\mathrm{rank}(\mathbf{X}) < \mathrm{rank}(\tilde{\mathbf{X}}) \text{ where } \tilde{\mathbf{X}} = [\mathbf{X}\mathbf{y}].$$

Numerical example for binary classification

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

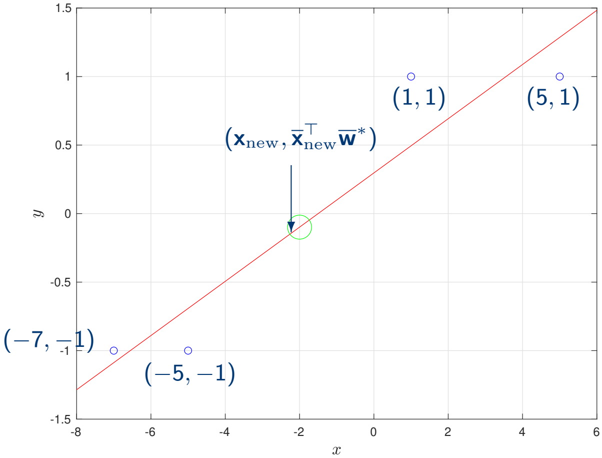
$$\overline{\mathbf{w}}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \begin{bmatrix} 0.2967 \\ 0.1978 \end{bmatrix}.$$

- 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).

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: linear classification

```python
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)
print("\n")

y_class_predict = np.sign(y_predict)
print("Predicted y class")
print(y_class_predict)
```

Linear models for multi-class classification

- **Main idea for binary classification:** to treat binary classification as regression where each label y_i can only take on -1 or $+1$.
  - **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

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

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

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

- How can we apply linear models for multi-class classification? Any guess?

Linear models for multi-class classification

- Suppose we want to distinguish between cats, dogs and birds. These are labelled as 1, 2, 3 respectively.

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:** 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

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

- (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

$$\begin{bmatrix} 1 \\ \mathbf{x}_{\text{new}} \end{bmatrix}^\top \overline{\mathbf{W}}^*[:, 1] = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}^\top \begin{bmatrix} 0 \\ 0.2857 \\ 0.2857 \end{bmatrix} = -0.2857, \quad \begin{bmatrix} 1 \\ \mathbf{x}_{\text{new}} \end{bmatrix}^\top \overline{\mathbf{W}}^*[:, 2] = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}^\top \begin{bmatrix} 0.5 \\ -0.5 \\ 0 \end{bmatrix} = 0.5,$$

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

- Its predicted class is

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

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

Python demo: setting up and one-hot encoding

```python
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)
```

- sparse=False: determine the datatype of output matrix
- version 1.2 of OneHotEncoder: sparse was renamed to sparse_output

Python demo: training and testing

```python
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("Predicted class label test using argmax")
print(np.argmax(yt_est)+1)
```

Python demo: training and testing

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

- Check: is $\mathbf{X}^\top\mathbf{X}$ invertible?

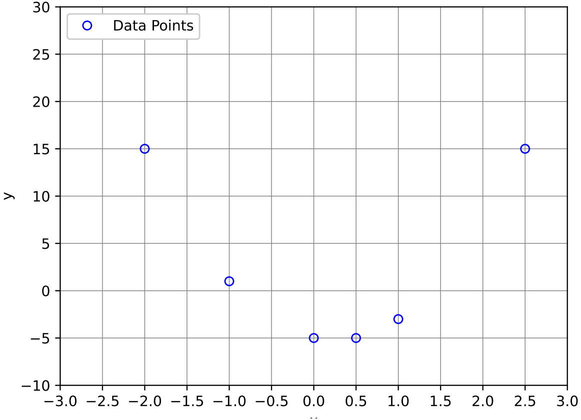
Raises:

LinAlgError

If *a* is not square or inversion fails.

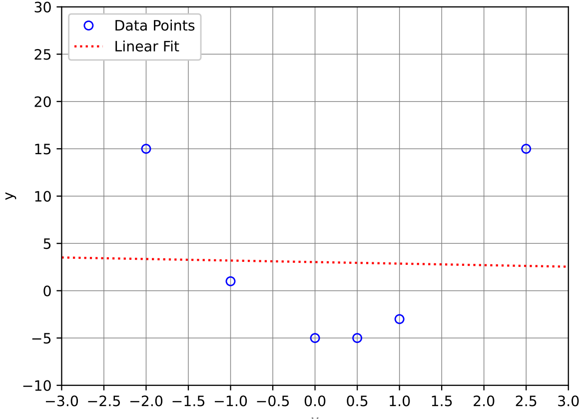
Outline

Motivation for polynomial regression



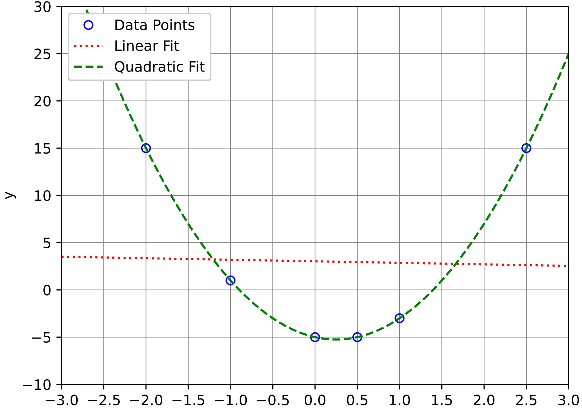
Motivation for polynomial regression

Sometimes affine functions do not do a good job!



Motivation for polynomial regression

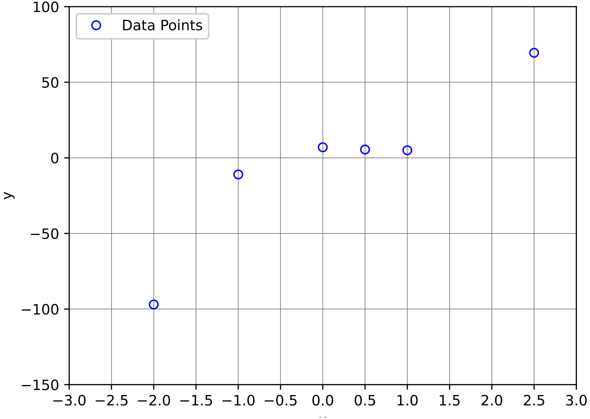
Sometimes affine functions do not do a good job!



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

Motivation for Polynomial Regression

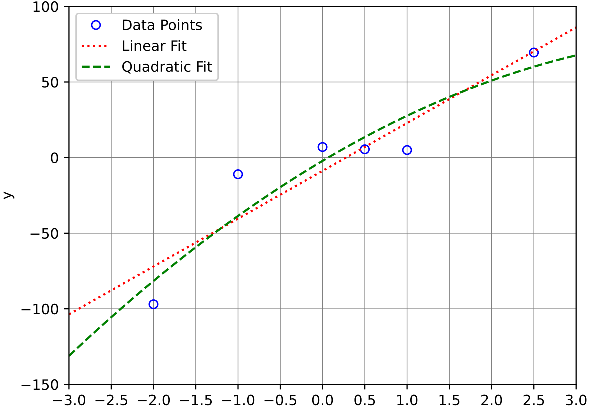
Sometimes affine functions do not do a good job!



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

Motivation for Polynomial Regression

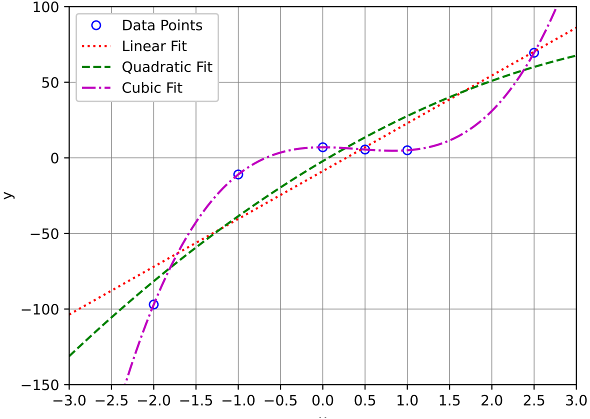
Sometimes affine functions do not do a good job!



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

Motivation for Polynomial Regression

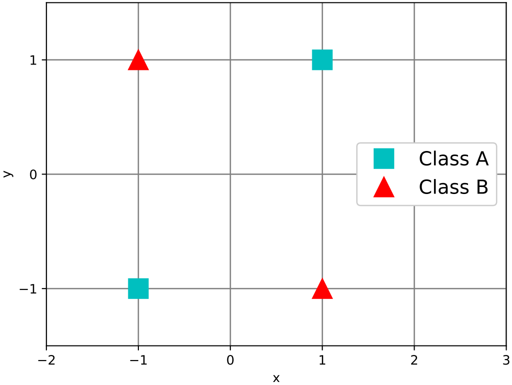
Sometimes affine functions do not do a good job!



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

Motivation for polynomial regression

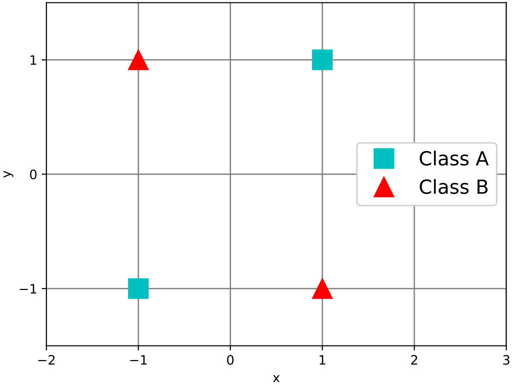
XOR dataset in $d = 2$ dimensions.



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

Motivation for polynomial regression

XOR dataset in $d = 2$ dimensions.



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

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

Polynomials

- We would like to model **nonlinear decision boundaries** or surfaces.

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

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

Polynomials

- 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.,

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

Polynomials

- 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.,

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

- 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?

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

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

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

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.

The XOR example revisited

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

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 pink column perfectly distinguishes the training points.

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 $x_1 x_2$.

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

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

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

Python demo for XOR: training/learning

```
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 for XOR: prediction/testing

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

Summary of polynomial regression

- **Learning/Training**:

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

  where

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

- **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/classification

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

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

- 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

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

  - Multi-class prediction

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

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) $\mathbf{x}_{\mathrm{new}}$, the prediction based on the least squares solution is

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

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

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

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

Review of Linear Regression with Multiple Outputs

- **Learning/Training**: Least Squares Solution

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

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

Review of Linear Regression with Multiple Outputs

- **Learning/Training**: Least Squares Solution

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

- **Prediction/Testing**: 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 $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$.

◎ Is the matrix $\mathbf{X}^\top \mathbf{X}$ invertible?

Review of polynomial regression

- **Learning/Training**:

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

where

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

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

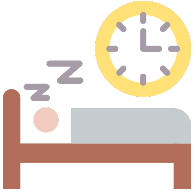
$$\hat{y}_{\text{new}} = \mathbf{p}_{\text{new}}^\top \mathbf{w}^*.$$

◎ Is the matrix $\mathbf{P}^\top\mathbf{P}$ invertible?

Motivation for ridge regression

How can we predict our academic performance in the coming semester?

Hours studied



Sleep hours

Extracurricular activities

Previous scores

Motivation for ridge regression

How can we predict our academic performance in the coming semester?

Hours studied

Sleep hours

Extracurricular activities

Previous scores

- Subject
- Commute time
- Age
- Male/Female
- Family income
- ......

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?

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

Recall that this was obtained from minimizing

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

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

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?

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

Recall that this was obtained from minimizing

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

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

- 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 $\Longrightarrow (\mathbf{X}^\top \mathbf{X})^{-1}$ does not exist.

Motivation for ridge regression

- Model possess too many features
- Go beyond the linear model, even an infinite-dimensional model

Motivation for ridge regression

- Model possess too many features
- Go beyond the linear model, even an infinite-dimensional model

◎ Stabilize and robustify the solution.

New objective function for ridge regression

- Recap of linear regression: We average the square of the errors over all training samples. This defines the objective or loss function

$$\mathrm{Loss}(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \left(f_{\mathbf{w},b}(\mathbf{x}_i) - y_i \right)^2 .$$

New objective function for ridge regression

- Recap of linear regression: We average the square of the errors over all training samples. This defines the objective or loss function

$$\mathrm{Loss}(\mathbf{w}, b) = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\mathbf{w},b}(\mathbf{x}_i) - y_i\right)^2 .$$

- **Ridge regression:** For a fixed $\lambda \geq 0$, consider

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

  Note that $w_0 = b$, the offset or bias.

New objective function for ridge regression

- **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

- Recall that we wish to solve

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

Solution for ridge regression

- Recall that we wish to solve

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

- Expanding the objective, we obtain

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

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

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

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

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

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

Legitimacy: $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}(\forall\lambda > 0)$ is always invertible

Proposition 4.1

The vector space consisting of only the zero vector has dimension 0.

Legitimacy: $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}(\forall\lambda > 0)$ is always invertible

Proposition 4.1

The vector space consisting of only the zero vector has dimension 0.

Proof. Apply the definition of dimension.

Legitimacy: $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}(\forall\lambda > 0)$ is always invertible

Proposition 4.1

The vector space consisting of only the zero vector has dimension 0.

Proof. Apply the definition of dimension.

Definition 4.2 (Definite matrix)

Let $\mathbf{A}$ denote a square matrix in $\mathbb{R}^{n\times n}$. $\mathbf{A}$ is said to be **positive-definite** if

$$\mathbf{x}^\top\mathbf{A}\mathbf{x} > 0 \text{ for all } \mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}.$$

$\mathbf{A}$ is said to be **negative-definite** if

$$\mathbf{x}^\top\mathbf{A}\mathbf{x} < 0 \text{ for all } \mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}.$$

Legitimacy: $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I} (\forall \lambda > 0)$ is always invertible

Proposition 4.3

If $A \in \mathbb{R}^{n\times n}$ is positive-definite or negative-definite, then A is invertible.

Legitimacy: $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I} (\forall \lambda > 0)$ is always invertible

Proposition 4.3

If $A \in \mathbb{R}^{n \times n}$ is positive-definite or negative-definite, then A is invertible.

Proof. (I) If A is positive-definite, $\mathbf{x}^\top \mathbf{A}\mathbf{x} > 0$ for all $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ implies that

$$\mathcal{N}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = 0\} = \{0\}. \tag{4.1}$$

Legitimacy: $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}(\forall\lambda > 0)$ is always invertible

Proposition 4.3

If $A \in \mathbb{R}^{n\times n}$ is positive-definite or negative-definite, then A is invertible.

Proof. (I) If A is positive-definite, $\mathbf{x}^\top\mathbf{A}\mathbf{x} > 0$ for all $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ implies that

$$\mathcal{N}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = 0\} = \{0\}. \tag{4.1}$$

Hence, $\dim(\mathcal{N}(\mathbf{A})) = 0$

Legitimacy: $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I} (\forall \lambda > 0)$ is always invertible

Proposition 4.3

If $A \in \mathbb{R}^{n \times n}$ is positive-definite or negative-definite, then A is invertible.

Proof. (I) If A is positive-definite, $\mathbf{x}^\top \mathbf{A}\mathbf{x} > 0$ for all $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ implies that

$$\mathcal{N}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = 0\} = \{0\}. \tag{4.1}$$

Hence, $\dim(\mathcal{N}(\mathbf{A})) = 0$ and $\operatorname{rank}(\mathbf{A}) = \dim(\mathcal{R}(\mathbf{A})) = d - \dim(\mathcal{N}(\mathbf{A})) = d$.

Legitimacy: $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I} (\forall \lambda > 0)$ is always invertible

Proposition 4.3

If $A \in \mathbb{R}^{n \times n}$ is positive-definite or negative-definite, then A is invertible.

Proof. (I) If A is positive-definite, $\mathbf{x}^\top \mathbf{A}\mathbf{x} > 0$ for all $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ implies that

$$\mathcal{N}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = 0\} = \{0\}. \tag{4.1}$$

Hence, $\dim(\mathcal{N}(\mathbf{A})) = 0$ and $\operatorname{rank}(\mathbf{A}) = \dim(\mathcal{R}(\mathbf{A})) = d - \dim(\mathcal{N}(\mathbf{A})) = d$. Therefore, A is invertible.

(II) Case where A is negative-definite can be similarly proven.

Legitimacy: $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I} (\forall\lambda > 0)$ is always invertible

Proof. $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I} \in \mathbb{R}^{(d+1)\times(d+1)}$ is a square matrix.

Legitimacy: $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}(\forall\lambda > 0)$ is always invertible

Proof. $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I} \in \mathbb{R}^{(d+1)\times(d+1)}$ is a square matrix. For all $\mathbf{z} \in \mathbb{R}^(d+1) \setminus \{\mathbf{0}\}$,

$$\mathbf{z}^\top(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})\mathbf{z} = \mathbf{z}^\top(\mathbf{X}^\top\mathbf{X})\mathbf{z} + \mathbf{z}^\top(\lambda\mathbf{I})\mathbf{z} = (\mathbf{X}\mathbf{z})^\top(\mathbf{X}\mathbf{z}) + \lambda\mathbf{z}^\top\mathbf{z} > 0. \tag{4.2}$$

Legitimacy: $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I} (\forall\lambda > 0)$ is always invertible

Proof. $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I} \in \mathbb{R}^{(d+1)\times(d+1)}$ is a square matrix. For all $\mathbf{z} \in \mathbb{R}^(d+1) \setminus \{\mathbf{0}\}$,

$$\mathbf{z}^\top(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})\mathbf{z} = \mathbf{z}^\top(\mathbf{X}^\top\mathbf{X})\mathbf{z} + \mathbf{z}^\top(\lambda\mathbf{I})\mathbf{z} = (\mathbf{X}\mathbf{z})^\top(\mathbf{X}\mathbf{z}) + \lambda\mathbf{z}^\top\mathbf{z} > 0. \tag{4.2}$$

$\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}$ is positive-definite and hence invertible.

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 $(d+1) \times (d+1)$.

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 is the problem with inverting the $(d+1) \times (d+1)$ matrix $\mathbf{X}^\top \mathbf{X} + \lambda_{d+1} \mathbf{I}$?

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 is 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.]

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 is 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 $\lambda > 0$,

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

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

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

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

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

Ridge regression in dual form [exercise]

Ridge regression in dual form [exercise]

Note that $\mathbf{X} \in \mathbb{R}^{m\times(d+1)}$. Starting from $\mathbf{X}^\top\left(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}_m\right)^{-1}\mathbf{y}$, we have

$$
\begin{aligned}
&\mathbf{X}^\top\left(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}_m\right)^{-1}\mathbf{y} \\
&= \lambda^{-1}\mathbf{X}^\top\left(\mathbf{I}_m + \lambda^{-1}\mathbf{X}\mathbf{X}^\top\right)^{-1}\mathbf{y} \\
&= \lambda^{-1}\mathbf{X}^\top\left[\mathbf{I}_m - \lambda^{-1}\mathbf{X}\left(\mathbf{I}_{d+1} + \lambda^{-1}\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\right]\mathbf{y} \qquad (4.3) \\
&= \lambda^{-1}\left(\mathbf{X}^\top\mathbf{y} - \mathbf{X}^\top\mathbf{X}\left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_{d+1}\right)^{-1}\mathbf{X}^\top\mathbf{y}\right) \\
&= \lambda^{-1}\left(\mathbf{I}_{d+1} - \mathbf{X}^\top\mathbf{X}\left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_{d+1}\right)^{-1}\right)\mathbf{X}^\top\mathbf{y} \\
&= \lambda^{-1}\left[\mathbf{I}_{d+1} - \left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_{d+1}\right)\left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_{d+1}\right)^{-1} + \lambda\mathbf{I}_{d+1}\left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_{d+1}\right)^{-1}\right]\mathbf{X}^\top\mathbf{y} \\
&= \left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_{d+1}\right)^{-1}\mathbf{X}^\top\mathbf{y}
\end{aligned}
$$

where (4.3) follows from the Woodbury matrix identity with $\mathbf{U} \equiv \lambda^{-1}\mathbf{X}$ and $\mathbf{V} \equiv \mathbf{X}^\top$.

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

- 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`

Take-away

Take-away

1. Least squares and linear regression
2. Linear classification
3. Polynomial regression
4. Ridge regression

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

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

MLE and linear regression

- Assume $y_i = \mathbf{w}^\top \mathbf{x}_i + b + e_i$ for each data point i and error $e_i \sim \mathcal{N}\left(0, \sigma^2\right)$.
- Likelihood function for the entire dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ is

$$L\left(\mathbf{W}, \sigma^2 \mid \{y_i, \mathbf{x}_i\}\right) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(y_i - \mathbf{W}^\top \mathbf{x}_i\right)^2}{2\sigma^2}\right)$$

- Maximizing the Log-Likelihood: take the derivative of the log-likelihood function with respect to $\mathbf{W}$ and set it equal to zero:

$$\frac{\partial}{\partial \mathbf{W}} \log L\left(\mathbf{W}, \sigma^2 \mid \{y_i, \mathbf{x}_i\}\right) = \frac{1}{\sigma^2} \sum_{i=1}^{n} \left(y_i - \mathbf{W}^\top \mathbf{x}_i\right) \mathbf{x}_i = 0$$

- If $\mathbf{X}$ has full column rank, $\mathbf{X}^\top \mathbf{X}$ is invertible and

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

is the least squares solution.

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

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

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

Review of linear regression with multiple outputs

- **Learning/Training**: Least Squares Solution

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

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

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

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

Review of linear regression with multiple outputs

- **Learning/Training**: Least Squares Solution

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

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

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

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

◎ Is the matrix $\mathbf{X}^\top \mathbf{X}$ invertible?

Review of polynomial regression

- **Learning/Training**:

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

where

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

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

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

◎ Is the matrix $\mathbf{P}^\top\mathbf{P}$ invertible?

Review of ridge regression (linear form)

- Ridge regression in primal form (when $m > d' = \binom{p+d}{p}$)
  - **Learning/Training**:
$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_{d+1})^{-1} \mathbf{X}^\top \mathbf{y}$$
  - **Prediction/Testing**: Given a new sample $\mathbf{x}_{\text{new}}$
$$\hat{y}_{\text{new}} = \begin{bmatrix} 1 \\ \mathbf{x}_{\text{new}} \end{bmatrix}^\top \mathbf{w}^*$$

  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**:
$$\mathbf{w}^* = \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_m)^{-1} \mathbf{y}$$
  - **Prediction/Testing**: Given a new sample $\mathbf{x}_{\text{new}}$
$$\hat{y}_{\text{new}} = \begin{bmatrix} 1 \\ \mathbf{x}_{\text{new}} \end{bmatrix}^\top \mathbf{w}^*$$

Review of ridge regression (polynomial form)

- 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 $\mathbf{x}_{\mathrm{new}}$
  $$\hat{y}_{\mathrm{new}} = \mathbf{p}_{\mathrm{new}}^\top\mathbf{w}^*$$
  where $\mathbf{p}_{\mathrm{new}}$ is the polynomial vector associated to $\mathbf{x}_{\mathrm{new}}$.

- 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 $\mathbf{x}_{\mathrm{new}}$
  $$\hat{y}_{\mathrm{new}} = \mathbf{p}_{\mathrm{new}}^\top\mathbf{w}^*$$

Review of regression/classification

- 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

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

  - Multi-class prediction

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

Thanks for listening

1. Tell us your question/feedback via the QR code.
2. Lab reminder: 8-9PM today, same classroom.

⊙ Slides credit: some slides are adapted from Vincent Y. F. Tan (NUS).