THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

DSAA2011 Machine Learning

L3 Supervised Learning: Regression and Classification II

Dr. Zixin Zhong

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

February 21, 2025

Syllabus

Week #	Topic	Lecturer
1	Introduction + course Info	Zixin
2-4	**Supervised learning: regression and classification**	Zixin
5	Model evaluation and choice + 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	

Office hours (weekly, starting from 11 Feb)

Time	Venue	Instructor/TA	Email
7-8PM Wed (7-8PM Tue)	Rm E2-301	Weiwen CHEN	wchen948@connect.hkust-gz.edu.cn
		Guanghua LI	gli945@connect.hkust-gz.edu.cn
		Yang LUO	yluo208@connect.hkust-gz.edu.cn
		Chunming MA	cma859@connect.hkust-gz.edu.cn
		Jingyi PAN	jpan305@connect.hkust-gz.edu.cn
		Liangwei WANG	lwang344@connect.hkust-gz.edu.cn
		Yifan ZHANG	yzhang854@connect.hkust-gz.edu.cn
3-4PM Fri	Rm W1-316	Weikai Yang	weikaiyang@hkust-gz.edu.cn
	Rm W1-308	Zixin Zhong	zixinzhong@hkust-gz.edu.cn

♡ **Please show respect and appreciation to our TAs!**

Recap of Lecture 2

- Maximum likelihood estimation (MLE)
- Least squares and linear regression
  - Linear regression with multiple outputs
  - Linear regression and MLE

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.
- **Asymptotic Normality.** MLE is asymptotically normal: $\sqrt{n}(\hat{\theta}_{\mathrm{ML}} - \theta) \xrightarrow{d} \mathcal{N}(0, I(\theta)^{-1})$.
  - $I(\theta)$: 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: $\sqrt{n}(\hat{\theta}_{\mathrm{ML}} - \theta) \xrightarrow{d} \mathcal{N}(0, I(\theta)^{-1})$.
- **Model Sensitivity.** MLE is sensitive to model assumptions, and incorrect assumptions can lead to biased or inconsistent estimates.

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

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

Linear regression with multiple outputs

- Suppose there are h outputs we want to predict (above $h = 1$).
- 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.

Linear regression with multiple outputs

- Suppose there are h outputs we want to predict (above $h = 1$).
- 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.

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 $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$.

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

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

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

- If $\mathbf{X}$ has full column rank, $\mathbf{X}^\top\mathbf{X}$ is invertible and the maximizer $(\hat{\mathbf{w}}, \hat{\sigma}^2)$ is:

$$\hat{\mathbf{w}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} \quad (\text{least squares solution}),$$

$$\hat{\sigma}^2 = \frac{1}{m}\sum_{i=1}^{m}(y_i - \begin{bmatrix}1 & x_i^\top\end{bmatrix} \cdot \hat{\mathbf{w}})^2 = \frac{1}{m}(\mathbf{X}\hat{\mathbf{w}} - \mathbf{y})^\top(\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}).$$

⊙ **MLE of distribution of error:** $e_i \sim \mathcal{N}(0, \hat{\sigma}^2) = \mathcal{N}(0, (\mathbf{X}\hat{\mathbf{w}} - \mathbf{y})^\top(\mathbf{X}\hat{\mathbf{w}} - \mathbf{y})/m)$.

MLE and linear regression

- If $\mathbf{X} = \left[\overline{\mathbf{x}}_1^\top \ \overline{\mathbf{x}}_2^\top \ \ldots \ \overline{\mathbf{x}}_n^\top\right]^\top$ has full column rank, $\mathbf{X}^\top\mathbf{X}$ is invertible and

$$\frac{\partial}{\partial \overline{\mathbf{w}}} \log L\left(\overline{\mathbf{w}}, \sigma^2 \mid \{y_i, \mathbf{x}_i\}\right) = \frac{1}{\sigma^2} \sum_{i=1}^{m} \left(y_i - \overline{\mathbf{w}}^\top \overline{\mathbf{x}}_i\right) \overline{\mathbf{x}}_i = \mathbf{0}_{(d+1)\times 1} \tag{0.1}$$

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

MLE and linear regression

- If $\mathbf{X} = \left[\overline{\mathbf{x}}_1^\top \ \overline{\mathbf{x}}_2^\top \ \ldots \ \overline{\mathbf{x}}_n^\top\right]^\top$ has full column rank, $\mathbf{X}^\top\mathbf{X}$ is invertible and

$$\frac{\partial}{\partial \overline{\mathbf{w}}} \log L\left(\overline{\mathbf{w}}, \sigma^2 \mid \{y_i, \mathbf{x}_i\}\right) = \frac{1}{\sigma^2}\sum_{i=1}^{m}\left(y_i - \overline{\mathbf{w}}^\top \overline{\mathbf{x}}_i\right)\overline{\mathbf{x}}_i = \mathbf{0}_{(d+1)\times 1} \tag{0.1}$$

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

Proof. Firstly, (0.1) can be rewritten as $\sum_{i=1}^{m}\left(\overline{\mathbf{w}}^\top \overline{\mathbf{x}}_i\right)\overline{\mathbf{x}}_i = \sum_{i=1}^{m} y_i \overline{\mathbf{x}}_i$.

MLE and linear regression

Proof. Firstly, (0.1) can be rewritten as $\sum_{i=1}^{m}\left(\overline{\mathbf{w}}^{\top}\overline{\mathbf{x}}_i\right)\overline{\mathbf{x}}_i = \sum_{i=1}^{m} y_i\overline{\mathbf{x}}_i$. Then observe that

$$\sum_{i=1}^{m} y_i\overline{\mathbf{x}}_i = \begin{bmatrix}\overline{\mathbf{x}}_1 & \overline{\mathbf{x}}_2 & \dots & \overline{\mathbf{x}}_n\end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \mathbf{X}^{\top}\mathbf{y},$$

MLE and linear regression

Proof. Firstly, (0.1) can be rewritten as $\sum_{i=1}^{m}\left(\overline{\mathbf{w}}^\top \overline{\mathbf{x}}_i\right)\overline{\mathbf{x}}_i = \sum_{i=1}^{m} y_i \overline{\mathbf{x}}_i$. Then observe that

$$\sum_{i=1}^{m} y_i \overline{\mathbf{x}}_i = \begin{bmatrix} \overline{\mathbf{x}}_1 & \overline{\mathbf{x}}_2 & \dots & \overline{\mathbf{x}}_n \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \mathbf{X}^\top \mathbf{y},$$

$$\begin{aligned}\sum_{i=1}^{m}\left(\overline{\mathbf{w}}^\top \overline{\mathbf{x}}_i\right)\overline{\mathbf{x}}_i &= \sum_{i=1}^{m}\left(\overline{\mathbf{x}}_i^\top \overline{\mathbf{w}}\right)\overline{\mathbf{x}}_i = \sum_{i=1}^{m}\overline{\mathbf{x}}_i\left(\overline{\mathbf{x}}_i^\top \overline{\mathbf{w}}\right) = \sum_{i=1}^{m}\left(\overline{\mathbf{x}}_i\overline{\mathbf{x}}_i^\top\right)\overline{\mathbf{w}} = \left(\sum_{i=1}^{m}\overline{\mathbf{x}}_i\overline{\mathbf{x}}_i^\top\right)\overline{\mathbf{w}} \\ &= \begin{bmatrix} \overline{\mathbf{x}}_1 & \overline{\mathbf{x}}_2 & \dots & \overline{\mathbf{x}}_m \end{bmatrix} \cdot \begin{bmatrix} \overline{\mathbf{x}}_1^\top \\ \overline{\mathbf{x}}_2^\top \\ \dots \\ \overline{\mathbf{x}}_m^\top \end{bmatrix} \cdot \overline{\mathbf{w}} = \mathbf{X}^\top \mathbf{X}\overline{\mathbf{w}}.\end{aligned}$$

MLE and linear regression

Proof. Firstly, (0.1) can be rewritten as $\sum_{i=1}^{m}\left(\overline{\mathbf{w}}^{\top}\overline{\mathbf{x}}_i\right)\overline{\mathbf{x}}_i = \sum_{i=1}^{m} y_i\overline{\mathbf{x}}_i$. Then observe that

$$\sum_{i=1}^{m} y_i\overline{\mathbf{x}}_i = \begin{bmatrix}\overline{\mathbf{x}}_1 & \overline{\mathbf{x}}_2 & \dots & \overline{\mathbf{x}}_n\end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \mathbf{X}^{\top}\mathbf{y},$$

$$\sum_{i=1}^{m}\left(\overline{\mathbf{w}}^{\top}\overline{\mathbf{x}}_i\right)\overline{\mathbf{x}}_i = \sum_{i=1}^{m}\left(\overline{\mathbf{x}}_i^{\top}\overline{\mathbf{w}}\right)\overline{\mathbf{x}}_i = \sum_{i=1}^{m}\overline{\mathbf{x}}_i\left(\overline{\mathbf{x}}_i^{\top}\overline{\mathbf{w}}\right) = \sum_{i=1}^{m}\left(\overline{\mathbf{x}}_i\overline{\mathbf{x}}_i^{\top}\right)\overline{\mathbf{w}} = \left(\sum_{i=1}^{m}\overline{\mathbf{x}}_i\overline{\mathbf{x}}_i^{\top}\right)\overline{\mathbf{w}}$$

$$= \begin{bmatrix}\overline{\mathbf{x}}_1 & \overline{\mathbf{x}}_2 & \dots & \overline{\mathbf{x}}_m\end{bmatrix} \cdot \begin{bmatrix} \overline{\mathbf{x}}_1^{\top} \\ \overline{\mathbf{x}}_2^{\top} \\ \dots \\ \overline{\mathbf{x}}_m^{\top} \end{bmatrix} \cdot \overline{\mathbf{w}} = \mathbf{X}^{\top}\mathbf{X}\overline{\mathbf{w}}.$$

Then (0.1) can be further rewritten as $\mathbf{X}^{\top}\mathbf{X}\overline{\mathbf{w}} = \mathbf{X}^{\top}\mathbf{y}$.

MLE and linear regression: why Gaussian noise?

MLE and linear regression: why Gaussian noise?

Theorem 0.1 (Central limit theorem (CLT))

Suppose $X_1, X_2, X_3 \ldots$ *is a sequence of i.i.d. random variables with* $\mathrm{E}[X_i] = \mu$ *and* $\mathrm{Var}[X_i] = \sigma^2 < \infty$. *Then, as* $n \to \infty$, *the distribution of* $\sqrt{n}\left(\bar{X}_n - \mu\right)$ *converges to* $\mathcal{N}\left(0, \sigma^2\right)$:

$$\sqrt{n}\left(\bar{X}_n - \mu\right) \xrightarrow{d} \mathcal{N}\left(0, \sigma^2\right)$$

MLE and linear regression: why Gaussian noise?

Theorem 0.1 (Central limit theorem (CLT))

Suppose $X_1, X_2, X_3 \ldots$ *is a sequence of i.i.d. random variables with* $\mathrm{E}[X_i] = \mu$ *and* $\mathrm{Var}[X_i] = \sigma^2 < \infty$. *Then, as* $n \to \infty$, *the distribution of* $\sqrt{n}\left(\bar{X}_n - \mu\right)$ *converges to* $\mathcal{N}\left(0, \sigma^2\right)$*:*

$$\sqrt{n}\left(\bar{X}_n - \mu\right) \xrightarrow{d} \mathcal{N}\left(0, \sigma^2\right)$$

- We say $X_n \xrightarrow{d} X$ as $n \to \infty$ if

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

MLE and linear regression: why Gaussian noise?

Theorem 0.1 (Central limit theorem (CLT))

Suppose $X_1, X_2, X_3 \ldots$ *is a sequence of i.i.d. random variables with* $\mathrm{E}[X_i] = \mu$ *and* $\mathrm{Var}[X_i] = \sigma^2 < \infty$. *Then, as* $n \to \infty$, *the distribution of* $\sqrt{n}\left(\bar{X}_n - \mu\right)$ *converges to* $\mathcal{N}\left(0, \sigma^2\right)$:

$$\sqrt{n}\left(\bar{X}_n - \mu\right) \xrightarrow{d} \mathcal{N}\left(0, \sigma^2\right)$$

- We say $X_n \xrightarrow{d} X$ as $n \to \infty$ if

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

- In the case $\sigma > 0$, CLT implies that the cumulative distribution functions (cdf) of $\sqrt{n}\left(\bar{X}_n - \mu\right)$ converge pointwise to the cdf of the $\mathcal{N}\left(0, \sigma^2\right)$ distribution:

$$\lim_{n\to\infty} \mathbb{P}\left[\sqrt{n}\left(\bar{X}_n - \mu\right) \le z\right] = \lim_{n\to\infty} \mathbb{P}\left[\frac{\sqrt{n}\left(\bar{X}_n - \mu\right)}{\sigma} \le \frac{z}{\sigma}\right] = \Phi\left(\frac{z}{\sigma}\right)$$

where $\Phi(z)$ is the standard normal cdf evaluated at z.

Outline

1. Linear classification
   - Linear models for binary classification
   - Linear models for multi-class classification
2. Polynomial regression
3. Ridge regression

Outline

Linear models for binary 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 binary 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.

- **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

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

Linear models for binary 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.

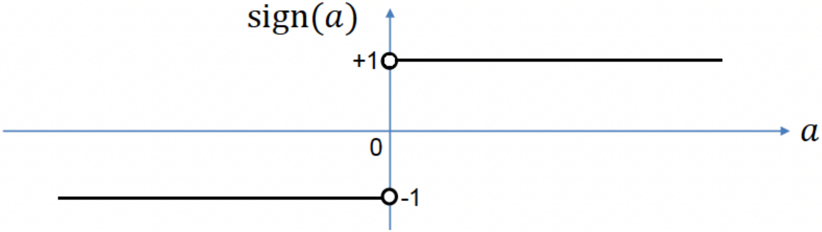
- **Learning/Training:** given a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ (where each $y_i \in \{+1, -1\}$), learn the weights using least squares

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

- **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

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

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

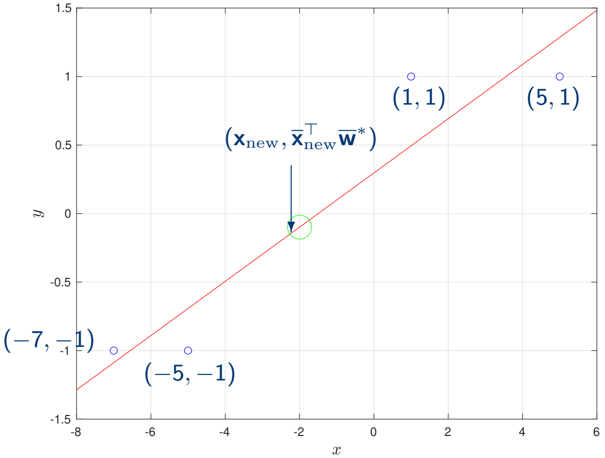
$$\overline{\mathbf{w}}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \begin{bmatrix} 0.2967 \\ 0.1978 \end{bmatrix}.$$

- 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

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

- 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 $\mathbf{x}_{\text{new}}$ is $\text{sign}(\overline{\mathbf{x}}_{\text{new}}^\top \overline{\mathbf{w}}^*) = -1$ as $\overline{\mathbf{x}}_{\text{new}}^\top \overline{\mathbf{w}}^*$ is negative.

Python demo: linear model for binary classification

```
import numpy as np
from numpy.linalg import inv

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

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

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

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

Linear models for multi-class classification

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

Any guess?

Linear models for multi-class classification

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

Linear models for multi-class classification

- Suppose we want to distinguish among 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 $\mathbf{x}_{\mathrm{new}} \in \mathbb{R}^d$, we have

$$\hat{y}^{\mathrm{new,reg}}[:, k] = \begin{bmatrix} 1 \\ \mathbf{x}_{\mathrm{new}} \end{bmatrix}^\top \overline{\mathbf{W}}^*[:, k] \quad \forall k \in \{1, 2, \ldots, C\}$$

- What's the next step?

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 $\mathbf{x}_{\rm new} \in \mathbb{R}^d$, we have

$$\hat{y}^{\rm new,reg}[:, k] = \begin{bmatrix} 1 \\ \mathbf{x}_{\rm new} \end{bmatrix}^\top \overline{\mathbf{W}}^*[:, k] \quad \forall k \in \{1, 2, \ldots, C\}$$

and predict its class as

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

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

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

Column position $k \in \{1, 2, 3\}$ of the largest number: predicted class label.

Python demo: setting up and one-hot encoding

```
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?

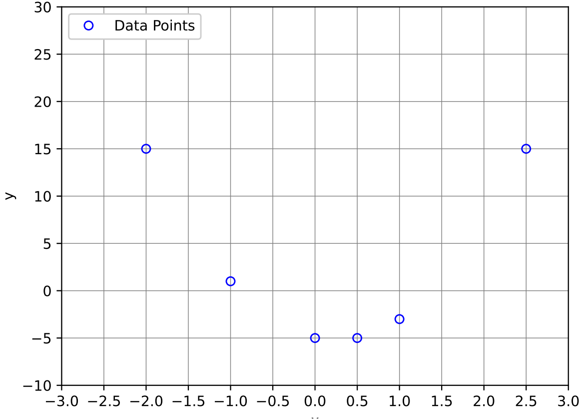
Raises:

LinAlgError

If *a* is not square or inversion fails.

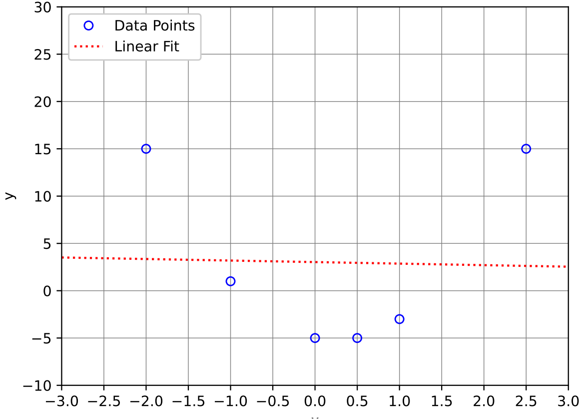
Outline

Motivation for polynomial regression



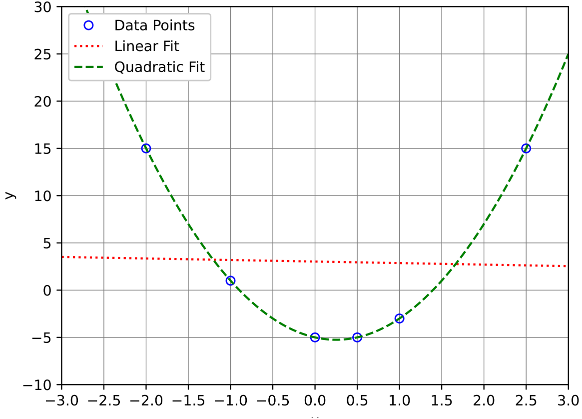
Motivation for polynomial regression

Sometimes affine functions do not do a good job!



Motivation for polynomial regression

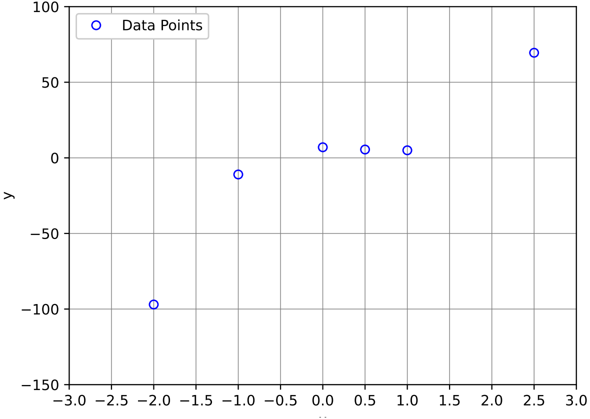
Sometimes affine functions do not do a good job!



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

Motivation for Polynomial Regression

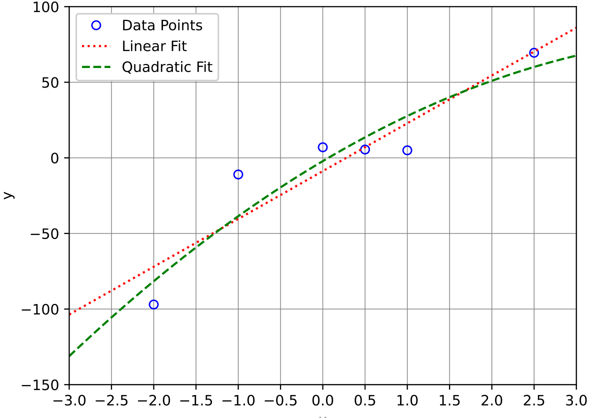
Sometimes affine functions do not do a good job!



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

Motivation for Polynomial Regression

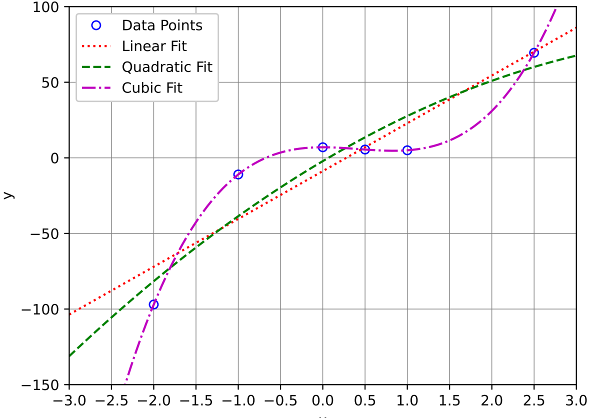
Sometimes affine functions do not do a good job!



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

Motivation for Polynomial Regression

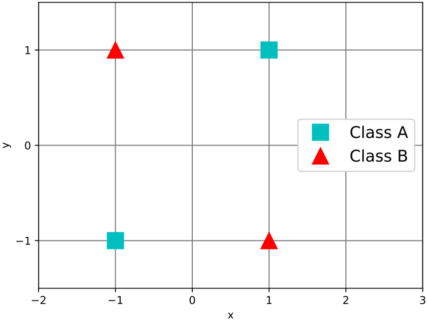
Sometimes affine functions do not do a good job!



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

Motivation for polynomial regression

XOR dataset in $d = 2$ dimensions.



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

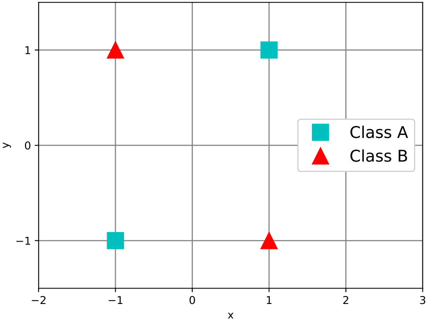
$$\mathbf{x}_2 = \begin{bmatrix} -1 & +1 \end{bmatrix}^\top$$

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

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

Motivation for polynomial regression

XOR dataset in $d = 2$ dimensions.

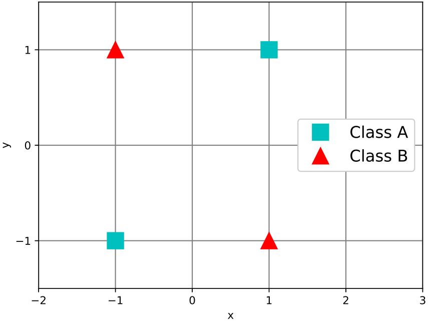


$$\mathbf{x}_1 = \begin{bmatrix} +1 & +1 \end{bmatrix}^\top$$
$$\mathbf{x}_2 = \begin{bmatrix} -1 & +1 \end{bmatrix}^\top$$
$$\mathbf{x}_3 = \begin{bmatrix} +1 & -1 \end{bmatrix}^\top$$
$$\mathbf{x}_4 = \begin{bmatrix} -1 & -1 \end{bmatrix}^\top$$

- XOR (exclusive OR) logical operation
  - fundamental binary operation in Boolean logic
  - output: 1 when the inputs are different, and 0 when the inputs are the same

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 (a product of constants and variables). 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 (a product of constants and variables). 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,

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

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?

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

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

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

Polynomial regression

- Generalized Linear Discriminant Function

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

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 $m \times \binom{d+p}{p}$.

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 $m \times \binom{d+1}{1} = m \times (d+1)$.

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

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

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

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 $\lambda = 0$)

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

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

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

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

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

The XOR example revisited

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

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

- Its predicted label is

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

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

Python demo for XOR: training/learning

```python
import numpy as np
from numpy.linalg import inv
from numpy.linalg import matrix_rank
from sklearn.preprocessing import PolynomialFeatures
X = np.array([[1, 1], [-1, 1], [1, -1], [-1, -1]])
y = np.array([[1], [-1], [-1], [1]])
## Generate polynomial features
order = 2
poly = PolynomialFeatures(order)
print(poly)
P = poly.fit_transform(X)
print("matrix P")
print(P)
```

Python demo for XOR: prediction/testing

```python
print("Under-determined system")
print(matrix_rank(P))
PY = np.vstack((P.T, y.T))
print(matrix_rank(PY.T))

## dual solution m < d (without ridge)
w_dual = P.T @ inv(P @ P.T) @ y
print("Unique constrained solution, no ridge")
print(w_dual)
#print(np.around(w_dual,3))
```

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:

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

- 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

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

Outline

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). 变量多样本小，用岭回归
- 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".

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".
- Question: what is the coincidence?

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

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

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

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

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

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

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

Why?

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

Proposition 3.1

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 3.1

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 3.1

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

Proof. Apply the definition of dimension.

Definition 3.2 (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 3.3

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 3.3

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

$$\mathcal{N}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = 0\} = \{\mathbf{0}\}. \tag{3.1}$$

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

Proposition 3.3

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

$$\mathcal{N}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = 0\} = \{\mathbf{0}\}. \tag{3.1}$$

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

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

Proposition 3.3

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

$$\mathcal{N}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = 0\} = \{\mathbf{0}\}. \tag{3.1}$$

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

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

Proposition 3.3

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

$$\mathcal{N}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = 0\} = \{\mathbf{0}\}. \tag{3.1}$$

Hence, $\dim(\mathcal{N}(\mathbf{A})) = 0$ and $\mathrm{rank}(\mathbf{A}) = \dim(\mathcal{R}(\mathbf{A})) = n - \dim(\mathcal{N}(\mathbf{A})) = n$. 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.
- To show $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I} \in \mathbb{R}^{(d+1)\times(d+1)}$ is invertible, it is sufficient to show that $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I} \in \mathbb{R}^{(d+1)\times(d+1)}$ is positive-definite or negative definite.

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.
- To show $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I} \in \mathbb{R}^{(d+1)\times(d+1)}$ is invertible, it is sufficient to show that $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I} \in \mathbb{R}^{(d+1)\times(d+1)}$ is positive-definite or negative definite.
- For all $\mathbf{z} \in \mathbb{R}^{(d+1)}\backslash\{\mathbf{0}\}$,

$$\mathbf{z}^\top(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})\mathbf{z} = \mathbf{z}^\top(\mathbf{X}^\top\mathbf{X})\mathbf{z} + \mathbf{z}^\top(\lambda\mathbf{I})\mathbf{z} = (\mathbf{X}\mathbf{z})^\top(\mathbf{X}\mathbf{z}) + \lambda\mathbf{z}^\top\mathbf{z} > 0. \tag{3.2}$$

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.
- To show $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I} \in \mathbb{R}^{(d+1)\times(d+1)}$ is invertible, it is sufficient to show that $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I} \in \mathbb{R}^{(d+1)\times(d+1)}$ is positive-definite or negative definite.
- For all $\mathbf{z} \in \mathbb{R}^{(d+1)}\backslash\{\mathbf{0}\}$,

$$\mathbf{z}^\top(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})\mathbf{z} = \mathbf{z}^\top(\mathbf{X}^\top\mathbf{X})\mathbf{z} + \mathbf{z}^\top(\lambda\mathbf{I})\mathbf{z} = (\mathbf{X}\mathbf{z})^\top(\mathbf{X}\mathbf{z}) + \lambda\mathbf{z}^\top\mathbf{z} > 0. \tag{3.2}$$

- Hence, $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}$ 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

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

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

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

Ridge regression in primal form

- The solution is known as the

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

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 $(d+1) \times (d+1)$.
- Q: What is the problem with inverting the $(d+1) \times (d+1)$ matrix $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_{d+1}$?

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 $(d+1) \times (d+1)$.
- Q: What is the problem with inverting the $(d+1) \times (d+1)$ matrix $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_{d+1}$?
- $d > m$ 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 $(d+1) \times (d+1)$.
- Q: What is the problem with inverting the $(d+1) \times (d+1)$ matrix $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_{d+1}$?
- $d > m$ 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

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

Ridge regression in dual form

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

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

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

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

- **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]

Apply the Woodbury formula

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

to show

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

d和m谁更小选谁

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

$$\begin{aligned}
&\mathbf{X}^\top\left(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_m\right)^{-1}\mathbf{y} \\
&= \lambda^{-1}\mathbf{X}^\top\left(\mathbf{I}_m + \lambda^{-1}\mathbf{X}\mathbf{X}^\top\right)^{-1}\mathbf{y} \\
&= \lambda^{-1}\mathbf{X}^\top\left[\mathbf{I}_m - \lambda^{-1}\mathbf{X}\left(\mathbf{I}_{d+1} + \lambda^{-1}\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\right]\mathbf{y} \qquad (3.3) \\
&= \lambda^{-1}\left(\mathbf{X}^\top\mathbf{y} - \mathbf{X}^\top\mathbf{X}\left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_{d+1}\right)^{-1}\mathbf{X}^\top\mathbf{y}\right) \\
&= \lambda^{-1}\left(\mathbf{I}_{d+1} - \mathbf{X}^\top\mathbf{X}\left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_{d+1}\right)^{-1}\right)\mathbf{X}^\top\mathbf{y} \\
&= \lambda^{-1}\left[\mathbf{I}_{d+1} - \left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_{d+1}\right)\left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_{d+1}\right)^{-1} + \lambda\mathbf{I}_{d+1}\left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_{d+1}\right)^{-1}\right]\mathbf{X}^\top\mathbf{y} \\
&= \left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_{d+1}\right)^{-1}\mathbf{X}^\top\mathbf{y}
\end{aligned}$$

where (3.3) follows from the Woodbury matrix identity with $\mathbf{U} \equiv \lambda^{-1}\mathbf{X}$ and $\mathbf{V} \equiv \mathbf{X}^\top$.

Python demo: ridge regression

	Previous Scores	Hours Studied	Extracurricular Activities_bool	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	99	7	1	9	1	91.0
1	82	4	0	4	2	65.0
2	51	8	1	7	2	45.0
3	52	5	1	5	2	36.0
4	75	7	0	8	5	66.0
5	78	3	0	9	6	61.0
6	73	7	1	5	6	63.0
7	45	8	1	4	6	42.0
8	77	5	0	8	2	61.0
9	89	4	0	4	0	69.0
10	91	8	0	4	5	84.0
11	79	8	0	6	2	73.0
12	47	3	0	9	2	27.0
13	47	6	0	4	2	33.0
14	79	5	0	7	8	68.0

Python demo: ridge regression

```
X = df_bool[['Previous Scores',
 'Hours Studied',
 'Extracurricular Activities_bool',
 'Sleep Hours',
 'Sample Question Papers Practiced']].to_numpy(copy=True)
y = df_bool[['Performance Index']].to_numpy(copy=True)
```

```
# split the data into training and test samples
X_train, X_test, y_train, y_test = sklearn.model_selection.train_test_split(
    X, y, test_size=0.3)
clf = Ridge(alpha=1.0).fit(X_train, y_train)
```

Python demo: ridge regression

```
for i in range(10):
    X_true = X_test[i,:].reshape(1, -1)
    y_true = y_test[i]
    y_pred = clf.predict(X_true)

    print('%d-th new sample:' % (i+1))
    print('True y: %.3f' % y_true[0])
    print('Predicted y: %.3f' % y_pred[0])
    print('=========')
```

```
1-th new sample:
True y: 36.000
Predicted y: 33.970
=========
2-th new sample:
True y: 26.000
Predicted y: 25.063
=========
```

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

Ridge regression (polynomial form)

- 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`, `pandas`

Thanks for listening

1. Tell us your question/feedback via QR code/Email/Teams.
2. Lab reminder: 4:30-5:30PM Thur, same classroom.

⊙ Slides credit: some slides are adapted from (alphabetical order)
Dan Klein and Pieter Abbeel (UC Berkeley), Haiyun He (Cornell) and Tommi S. Jaakkola (MIT).