IML Tutorial 2

Linear Regression

26.02.2020
Xianyao Zhang (CGL/DRS)
xianyao.zhang@inf.ethz.ch

Correction of HW 1 on Moodle

- Question 14
- https://piazza.com/class/k6i4ygvjdai2re?cid=40

Today’s Tutorial

- A recap on recent lectures regarding regression
- More in-depth demos based on Prof. Krause’s demos
    - Also a bit about python usage
- Please only ask questions about this tutorial
    - Unless you think it is relevant enough and I can definitely answer it :)

Linear Regression

- Model: $\hat{y} = w_1 x_1 + w_2 x_2 + \ldots + w_{d-1} x_{d-1} + w_0$

Linear Regression

- Model: $\hat{y} = w_1 x_1 + w_2 x_2 + \ldots + w_{d-1} x_{d-1} + w_0$
- **OR**: $\hat{y} = \mathbf{w}^T \mathbf{x}$, $\mathbf{w} = [w_0, w_1, w_2, \ldots, w_{d-1}]^T$, $\mathbf{x} = [1, x_1, x_2, \ldots, x_{d-1}]^T$

Linear Regression

- Model: $\hat{y} = w_1 x_1 + w_2 x_2 + \ldots + w_{d-1} x_{d-1} + w_0$
- **OR**: $\hat{y} = \mathbf{w}^T \mathbf{x}$, $\mathbf{w} = [w_0, w_1, w_2, \ldots, w_{d-1}]^T, \mathbf{x} = [1, x_1, x_2, \ldots, x_{d-1}]^T$
- Data distribution: $(\mathbf{x}_i, y_i) \sim P_{(\mathbf{x},y)}$
  - e.g. $y = \mathbf{u}^T \mathbf{x} + \epsilon$, $\epsilon \sim N(0,1)$; e.g. $y \sim N(\|\mathbf{x}\|_2, \sigma^2)$

Linear Regression

- Model: $\hat{y} = w_1 x_1 + w_2 x_2 + \ldots + w_{d-1} x_{d-1} + w_0$
- **OR**: $\hat{y} = \mathbf{w}^T \mathbf{x}$, $\mathbf{w} = [w_0, w_1, w_2, \ldots, w_{d-1}]^T, \mathbf{x} = [1, x_1, x_2, \ldots, x_{d-1}]^T$
- Data distribution: $(\mathbf{x}_i, y_i) \sim P_{(\mathbf{x}, y)}$
  - e.g. $y = \mathbf{u}^T \mathbf{x} + \epsilon$, $\epsilon \sim N(0,1)$; e.g. $y \sim N(\|\mathbf{x}\|_2, \sigma^2)$
- **True risk** of model $\mathbf{w}$: $R(\mathbf{w}) = \mathbb{E}_{P_{(x,y)}}[(y - \mathbf{w}^T \mathbf{x})^2]$

Linear Regression

- Model: $\hat{y} = w_1 x_1 + w_2 x_2 + \ldots + w_{d-1} x_{d-1} + w_0$
- **OR**: $\hat{y} = \mathbf{w}^T \mathbf{x}$, $\mathbf{w} = [w_0, w_1, w_2, \ldots, w_{d-1}]^T, \mathbf{x} = [1, x_1, x_2, \ldots, x_{d-1}]^T$
- Data distribution: $(\mathbf{x}_i, y_i) \sim P_{(\mathbf{x}, y)}$
  - e.g. $y = \mathbf{u}^T \mathbf{x} + \epsilon$, $\epsilon \sim N(0,1)$; e.g. $y \sim N(\|\mathbf{x}\|_2, \sigma^2)$
- **True risk** of model $\mathbf{w}$: $R(\mathbf{w}) = \mathbb{E}_{P_{(x,y)}}[(y - \mathbf{w}^T \mathbf{x})^2]$
- Data is usually infinite. How to **estimate** the true risk?

Monte Carlo Estimation

- Given a function $f(\cdot)$ and a distribution $p(\cdot)$ in a domain Ω, estimate $\mathbb{E}_{X\sim p}[f(X)]$

Monte Carlo Estimation

- Given a function $f(\cdot)$ and a distribution $p(\cdot)$ in a domain Ω, estimate $\mathbb{E}_{X\sim p}[f(X)]$

- $\mathbb{E}_{X\sim p}[f(X)] = \int_{\Omega} f(x)p(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)})$

    - $x^{(i)}$ are i.i.d. samples from distribution p

Monte Carlo Estimation

- Given a function $f(\cdot)$ and a distribution $p(\cdot)$ in a domain Ω, estimate $\mathbb{E}_{X\sim p}[f(X)]$

- $\mathbb{E}_{X\sim p}[f(X)] = \int_\Omega f(x)p(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)})$

  - $x^{(i)}$ are i.i.d. samples from distribution p

  - This estimate is **unbiased**: $\mathbb{E}_{x^{(i)}\sim p}[\frac{1}{N}\sum_{i=1}^{N} f(x^{(i)})] = \int_\Omega f(x)p(x)\,dx$

Monte Carlo Estimation

- In general, to estimate integral $\int_{\Omega} f(x)\, dx$

- Use samples from distribution $q(\,\cdot\,)$

  - $$\int_{\Omega} f(x)\, dx \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(x^{(i)})}{q(x^{(i)})}$$

  - $x^{(i)} \sim q$ instead of p

Monte Carlo Estimation

- In general, to estimate integral $\int_{\Omega} f(x)\, dx$

- Use samples from distribution $q(\,\cdot\,)$

  - $\int_{\Omega} f(x)\, dx \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(x^{(i)})}{q(x^{(i)})}$

  - $x^{(i)} \sim q$ instead of p

  - Unbiased if $q(x)$ is non-zero wherever $f(x)$ is non-zero

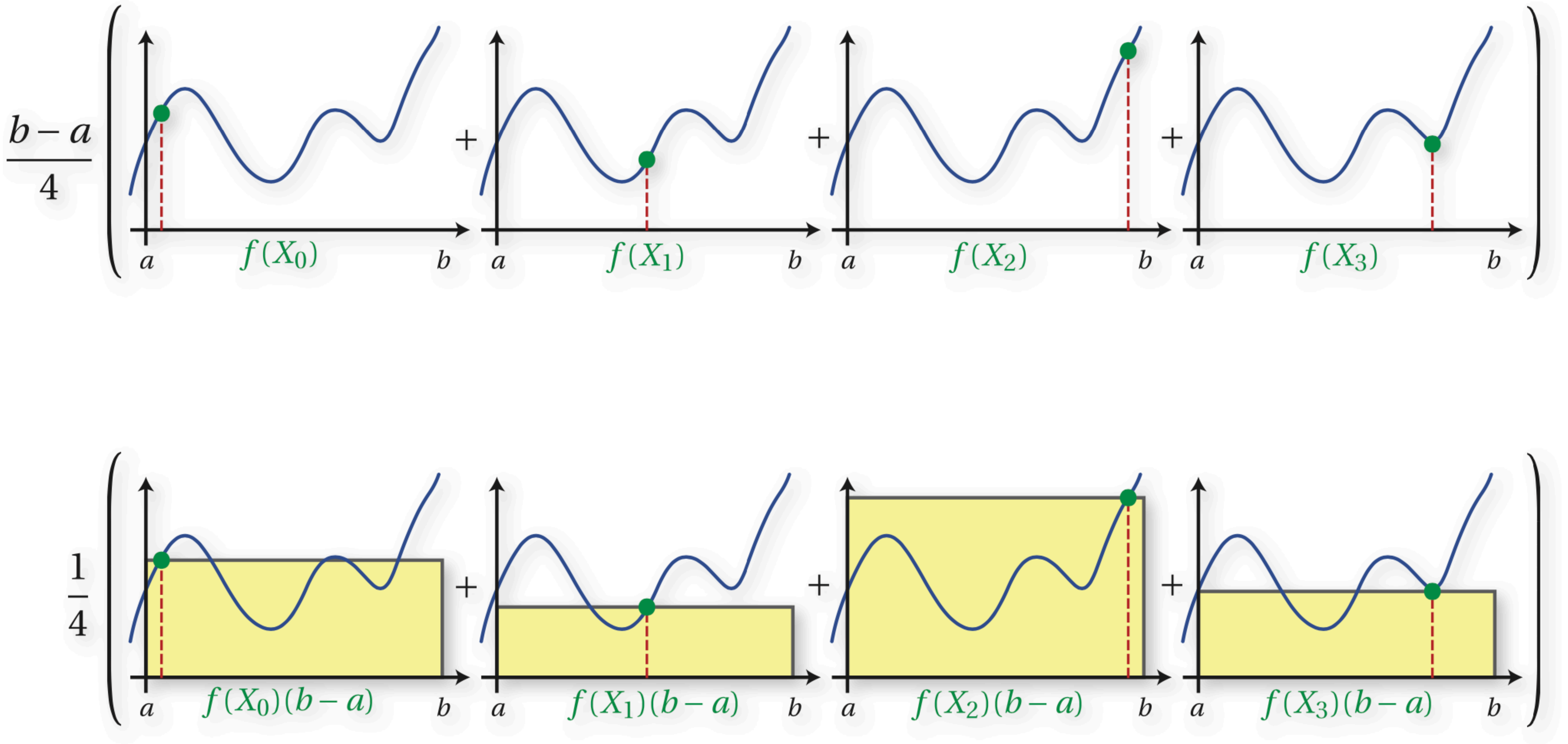


Image credit: Wojciech Jarosz

Monte Carlo Estimation

- $\mathbb{E}_{X\sim p}[f(X)] = \int_\Omega f(x)p(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)})$

- Can also use another distribution $q(\,\cdot\,)$

    - $\int_\Omega f(x)p(x)\,dx = \int_\Omega f(x)\frac{p(x)}{q(x)}q(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)})\frac{p(x^{(i)})}{q(x^{(i)})}$

    - $x^{(i)} \sim q$ instead of p

    - Unbiased if $q(x)$ is non-zero wherever $f(x)p(x)$ is non-zero

Linear Regression

- Hence we can use a finite dataset to estimate true risk $R(\mathbf{w})$
  - $R(\mathbf{w}) = \mathbb{E}_{P_{(\mathbf{x},y)}}[(y - \mathbf{w}^T\mathbf{x})^2]$

Linear Regression

- Hence we can use a finite dataset to estimate true risk $R(\mathbf{w})$
  - $R(\mathbf{w}) = \mathbb{E}_{P_{(\mathbf{x},y)}}[(y - \mathbf{w}^T\mathbf{x})^2]$
  - Dataset: $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, $D \sim P_D$, **i.i.d.** data examples $(\mathbf{x}_i, y_i) \sim P(\mathbf{x}, y)$
  - **Empirical risk** of model $\mathbf{w}$ on D: $\hat{R}_D(\mathbf{w}) = \dfrac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2$

Linear Regression

- Hence we can use a finite dataset to estimate true risk $R(\mathbf{w})$
  - $R(\mathbf{w}) = \mathbb{E}_{P_{(\mathbf{x},y)}}[(y - \mathbf{w}^T\mathbf{x})^2]$
- Dataset: $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, $D \sim P_D$, **i.i.d.** data examples $(\mathbf{x}_i, y_i) \sim P(\mathbf{x}, y)$
- **Empirical risk** of model $\mathbf{w}$ on D: $\hat{R}_D(\mathbf{w}) = \dfrac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2$
- Unbiased estimate of $R(\mathbf{w})$ if we **only use it to evaluate** $\mathbf{w}$
  - But we want to find a good model $\mathbf{w}$ with D (training)!

Closed-form solution

- $\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{i=1}^{N} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \implies \hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$
- $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T \in \mathbb{R}^{N\times d}$, $\mathbf{y} = [y_1, y_2, \ldots, y_N]^T \in \mathbb{R}^N$
  - Reformulate: $\mathbf{y} = \mathbf{X}\mathbf{w}$, usually over-constrained ($N \gg d$)
  - $\implies$ Least squares!

Gradient Descent

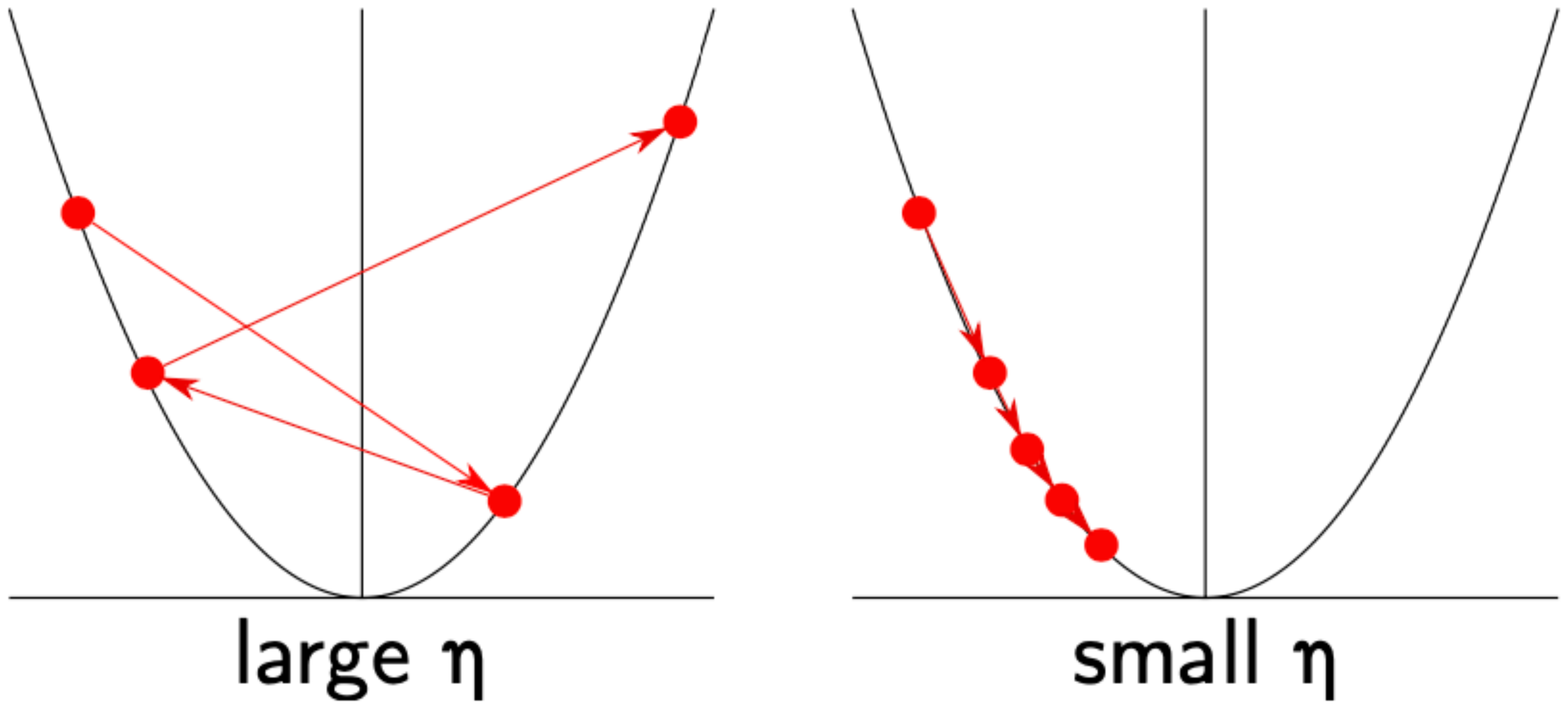
- $\mathbf{w}_0 \in \mathbb{R}^d$: initialization
- $\mathbf{w}_t = \mathbf{w}_{t-1} - \eta_t \nabla \hat{R}(\mathbf{w}_t)$: update at step $t = 1,2,3,\ldots$

Gradient Descent

- $\mathbf{w}_0 \in \mathbb{R}^d$: initialization
- $\mathbf{w}_t = \mathbf{w}_{t-1} - \eta_t \nabla \hat{R}(\mathbf{w}_t)$: update at step $t = 1,2,3,\ldots$
- Convex function: convergence guaranteed for small η_t

Gradient Descent

- $\mathbf{w}_0 \in \mathbb{R}^d$: initialization
- $\mathbf{w}_t = \mathbf{w}_{t-1} - \eta_t \nabla \hat{R}(\mathbf{w}_t)$: update at step $t = 1,2,3,\ldots$
- Convex function: convergence guaranteed for small η_t



Non-linear Features

- $y = \mathbf{w}^T\mathbf{x} \rightarrow y = \mathbf{v}^T\phi(\mathbf{x}), \mathbf{w} \in \mathbb{R}^m, \mathbf{v} \in \mathbb{R}^n$

Non-linear Features

- $y = \mathbf{w}^T \mathbf{x} \rightarrow y = \mathbf{v}^T \phi(\mathbf{x})$, $\mathbf{w} \in \mathbb{R}^m$, $\mathbf{v} \in \mathbb{R}^n$
- $\phi(\mathbf{x})$ is nonlinear: $\mathbb{R}^m \rightarrow \mathbb{R}^n$
    - e.g. $\mathbf{x} = [x_1, x_2, x_3]^T, \phi(\mathbf{x}) = [1, x_1, x_1^2, x_2^2 x_3, \ln 5x_3, e^{x_2 - x_1}]^T$

Non-linear Features

- $y = \mathbf{w}^T\mathbf{x} \rightarrow y = \mathbf{v}^T\phi(\mathbf{x}), \mathbf{w} \in \mathbb{R}^m, \mathbf{v} \in \mathbb{R}^n$
- $\phi(\mathbf{x})$ is nonlinear: $\mathbb{R}^m \rightarrow \mathbb{R}^n$
  - e.g. $\mathbf{x} = [x_1, x_2, x_3]^T, \phi(\mathbf{x}) = [1, x_1, x_1^2, x_2^2 x_3, \ln 5x_3, e^{x_2 - x_1}]^T$
- Can lead to better models if a good $\phi(\cdot)$ is selected

Non-linear Features

- $y = \mathbf{w}^T\mathbf{x} \rightarrow y = \mathbf{v}^T\phi(\mathbf{x}), \mathbf{w} \in \mathbb{R}^m, \mathbf{v} \in \mathbb{R}^n$
- $\phi(\mathbf{x})$ is nonlinear: $\mathbb{R}^m \rightarrow \mathbb{R}^n$
  - e.g. $\mathbf{x} = [x_1, x_2, x_3]^T, \phi(\mathbf{x}) = [1, x_1, x_1^2, x_2^2 x_3, \ln 5x_3, e^{x_2 - x_1}]^T$
- Can lead to better models if a good $\phi(\cdot)$ is selected
  - Worse models when picking a bad one :/
  - Also, tricky to pick the “just right” ones

Which Models Are Better?

- **The models $\mathbf{w}$ with lower true risk $R(\mathbf{w})$**

Which Models Are Better?

- **The models $\mathbf{w}$ with lower true risk $R(\mathbf{w})$**
- Under-fitting: not enough capacity
- Over-fitting: too much capacity —> fitting the noise!

Which Models Are Better?

- **The models $\mathbf{w}$ with lower true risk $R(\mathbf{w})$**
- Under-fitting: not enough capacity
- Over-fitting: too much capacity —> fitting the noise!
- Good model: neither under-fitting nor over-fitting

Training/Testing Split

- Empirical risk $\hat{R}_D(\hat{\mathbf{w}}_D)$ usually underestimate true risk $R(\hat{\mathbf{w}}_D)$
  - $\mathbb{E}_D[\hat{R}_D(\hat{\mathbf{w}}_D)] \leq \mathbb{E}_D[R(\hat{\mathbf{w}}_D)]$

What if we evaluate performance on training data?

$$\hat{\mathbf{w}}_D = \operatorname*{argmin}_{\mathbf{w}} \hat{R}_D(\mathbf{w}) \qquad \mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} R(\mathbf{w})$$

- In general, it holds that $\mathbb{E}_D\left[\hat{R}_D(\hat{\mathbf{w}}_D)\right] \underset{\sim}{<} \mathbb{E}_D\left[R(\hat{\mathbf{w}}_D)\right]$

$$\mathbb{E}_D[\hat{R}_D(\hat{w}_D)] \overset{\text{ERM}}{=} \mathbb{E}_D[\min_w \hat{R}_D(w)] \overset{\text{Jensen's}}{\leq} \min_w \mathbb{E}_D[\hat{R}_D(w)]$$

$$\overset{\text{Def.}}{=} \min_w \mathbb{E}_D\left[\frac{1}{|D|}\sum_{i=1}^{|D|}(y_i - w^T x_i)^2\right] \overset{\text{i.i.d.}}{=} \min_w \frac{1}{|D|}\sum_{i=1}^{|D|} \underbrace{\mathbb{E}_{(x_i,y_i)\sim P}(y_i - w^T x_i)^2}_{R(w)}$$

$$= \min_w R(w) \leq \mathbb{E}_D[R(\hat{w}_D)] \qquad \square$$

- **Thus, we obtain an overly optimistic estimate!**

Training/Testing Split

- Empirical risk $\hat{R}_D(\hat{\mathbf{w}}_D)$ usually underestimate true risk $R(\hat{\mathbf{w}}_D)$
  - $\mathbb{E}_D[\hat{R}_D(\hat{\mathbf{w}}_D)] \leq \mathbb{E}_D[R(\hat{\mathbf{w}}_D)]$
  - "Too optimistic" about the model
    - OR: the model only performs well on training data

Training/Testing Split

- Empirical risk $\hat{R}_D(\hat{\mathbf{w}}_D)$ usually underestimate true risk $R(\hat{\mathbf{w}}_D)$
  - $\mathbb{E}_D[\hat{R}_D(\hat{\mathbf{w}}_D)] \leq \mathbb{E}_D[R(\hat{\mathbf{w}}_D)]$
  - "Too optimistic" about the model
    - OR: the model only performs well on training data
- Unbiasedly estimate the true risk: **random** test set
- $\mathbb{E}_{D_{test}}[\hat{R}_{D_{test}}(\hat{\mathbf{w}}_{D_{train}})] = R(\hat{\mathbf{w}}_{D_{train}})$

Validation and Testing Sets?

- If we use only the training/testing split, we can overfit the testing set
    - $\hat{R}_{D_{test}}(\hat{\mathbf{w}}_{D_{train}}) \neq R(\hat{\mathbf{w}}_{D_{train}})$

Validation and Testing Sets?

- If we use only the training/testing split, we can overfit the testing set
    - $\hat{R}_{D_{test}}(\hat{\mathbf{w}}_{D_{train}}) \neq R(\hat{\mathbf{w}}_{D_{train}})$
- Do not select the model based on test set

Validation and Testing Sets?

- If we use only the training/testing split, we can overfit the testing set
  - $\hat{R}_{D_{test}}(\hat{\mathbf{w}}_{D_{train}}) \neq R(\hat{\mathbf{w}}_{D_{train}})$
- Do not select the model based on test set
- **Validation set** = reserve part of training set for model selection
  - Actually the “test set” before is a validation set
- Cross-validation = avoid bias of the validation set selection

Cross-validation

- Demo: k-fold CV for model selection

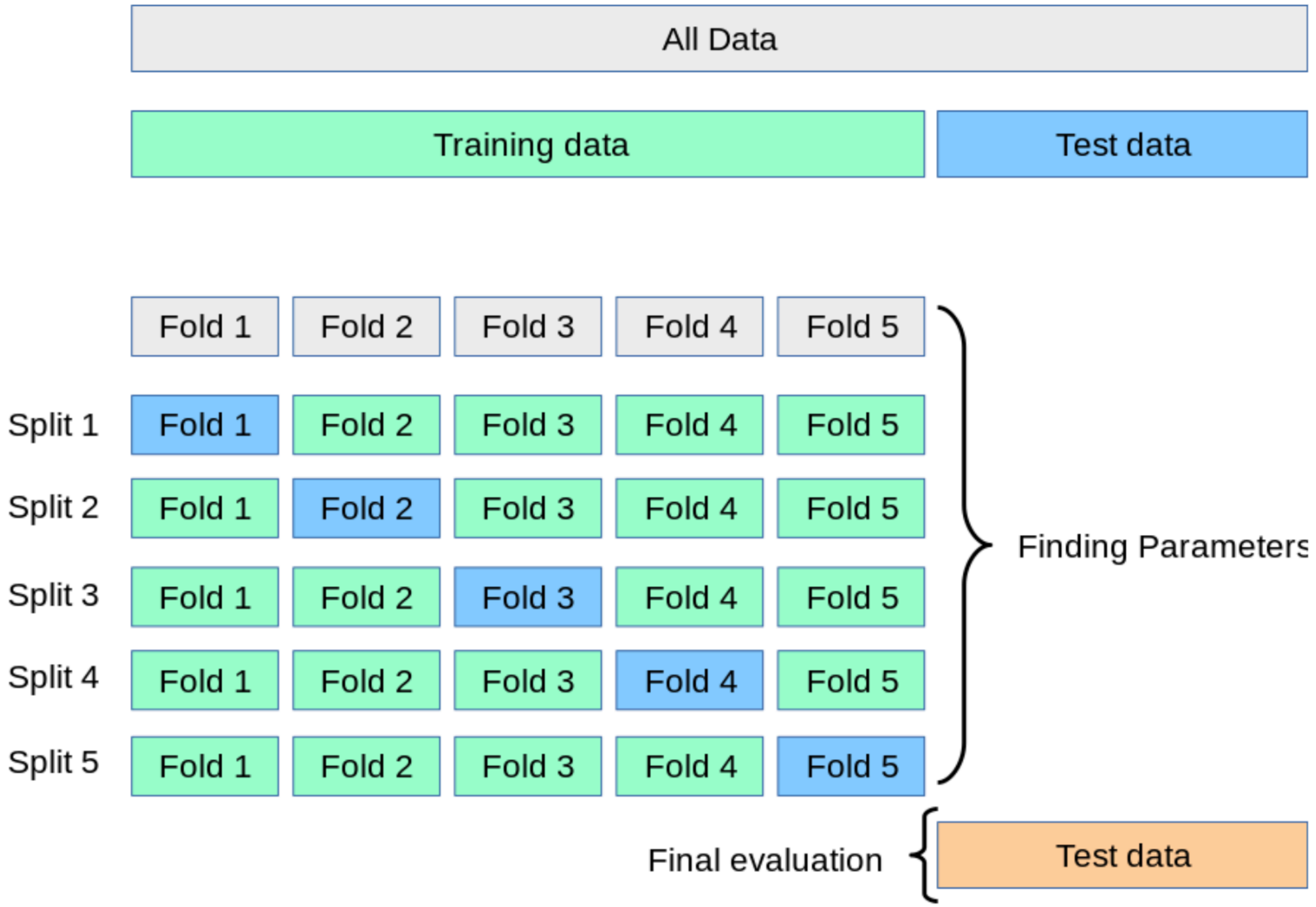


Image credit: sklearn

Regularization

- “Our models cannot be that complex, those large weights can only come from noise”
  - $\implies$ Penalize large weights in the loss functions

Regularization

- $\min_{\mathbf{w}} \hat{R}_D(\mathbf{w}) + \lambda C(\mathbf{w})$

- Linear regression: $\hat{R}_D(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x})^2$

Regularization

- $\min_{\mathbf{w}} \hat{R}_D(\mathbf{w}) + \lambda C(\mathbf{w})$

- Linear regression: $\hat{R}_D(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \mathbf{w}^T \mathbf{x})^2$

  - Ridge (L_2): $C(\mathbf{w}) = \|\mathbf{w}\|_2^2 = \sum_{k=1}^{d} w_k^2$, has closed form solution

Regularization

- $\min_{\mathbf{w}} \hat{R}_D(\mathbf{w}) + \lambda C(\mathbf{w})$

- Linear regression: $\hat{R}_D(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \mathbf{w}^T \mathbf{x})^2$

    - Ridge (L_2): $C(\mathbf{w}) = \|\mathbf{w}\|_2^2 = \sum_{k=1}^{d} w_k^2$, has closed form solution

    - Lasso (L_1): $C(\mathbf{w}) = \|\mathbf{w}\|_1 = \sum_{k=1}^{d} |w_k|$, doesn't have closed form solution

Lasso Leads to Sparsity

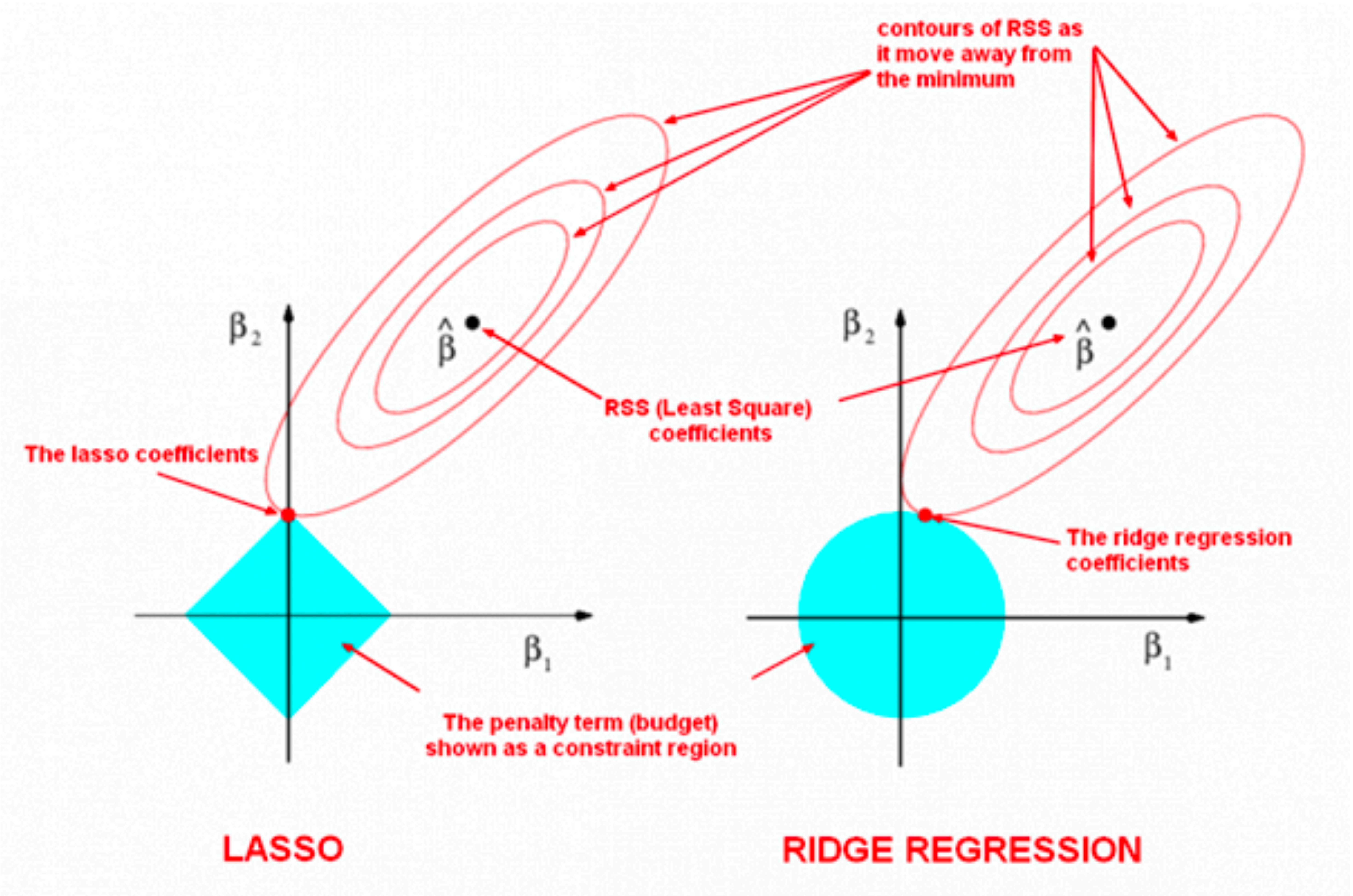


Image credit: link

Standardization

- The “small-weight” idea only applies when the data is standardized

Standardization

- The "small-weight" idea only applies when the data is standardized
- e.g. x_1 is income (10^4), x_2 is altitude (10^3), x_3 is height (10^0)

Standardization

- The “small-weight” idea only applies when the data is standardized
- e.g. x_1 is income (10^4), x_2 is altitude (10^3), x_3 is height (10^0)
- Originally $w_1 = 0.1, w_2 = 2, w_3 = 2000$
- Penalize $\|\mathbf{w}\|_2^2$ and get $w_1 = w_2 = w_3 = 1$
- x_3 will be useless!

Standardization

- The “small-weight” idea only applies when the data is standardized
- e.g. x_1 is income (10^4), x_2 is altitude (10^3), x_3 is height (10^0)
- Penalize $\|\mathbf{w}\|_2^2$ and get $w_1 = w_2 = w_3 = 1$
- x_3 will be useless!
- Standardize when using regularization: $\tilde{x}_{ij} = \dfrac{x_{ij} - \hat{\mu}_j}{\hat{\sigma}_j}$

End of Presentation
Beginning of Q&A

IML Tutorial 2
Linear Regression

26.02.2020

Xianyao Zhang (CGL/DRS)
xianyao.zhang@inf.ethz.ch