Supervised Learning
-- Regression

Hsu-Chao Lai

National Cheng Kung University

About Me

- Hsu-Chao Lai 賴旭昭
- AI Scientist at NetDB Lab
- Ph.D. of NYCU CS
- Specialized in
  - Recommender systems
  - Real-time bidding
  - AI stock trading
- hclai@netdb.csie.ncku.edu.tw

Roadmap

Raw Data →

Feature Engineering
- Normalization
- Data Cleaning
- Dimension Reduction
- Embedding/ Encoding
- Augmentation
- Interaction

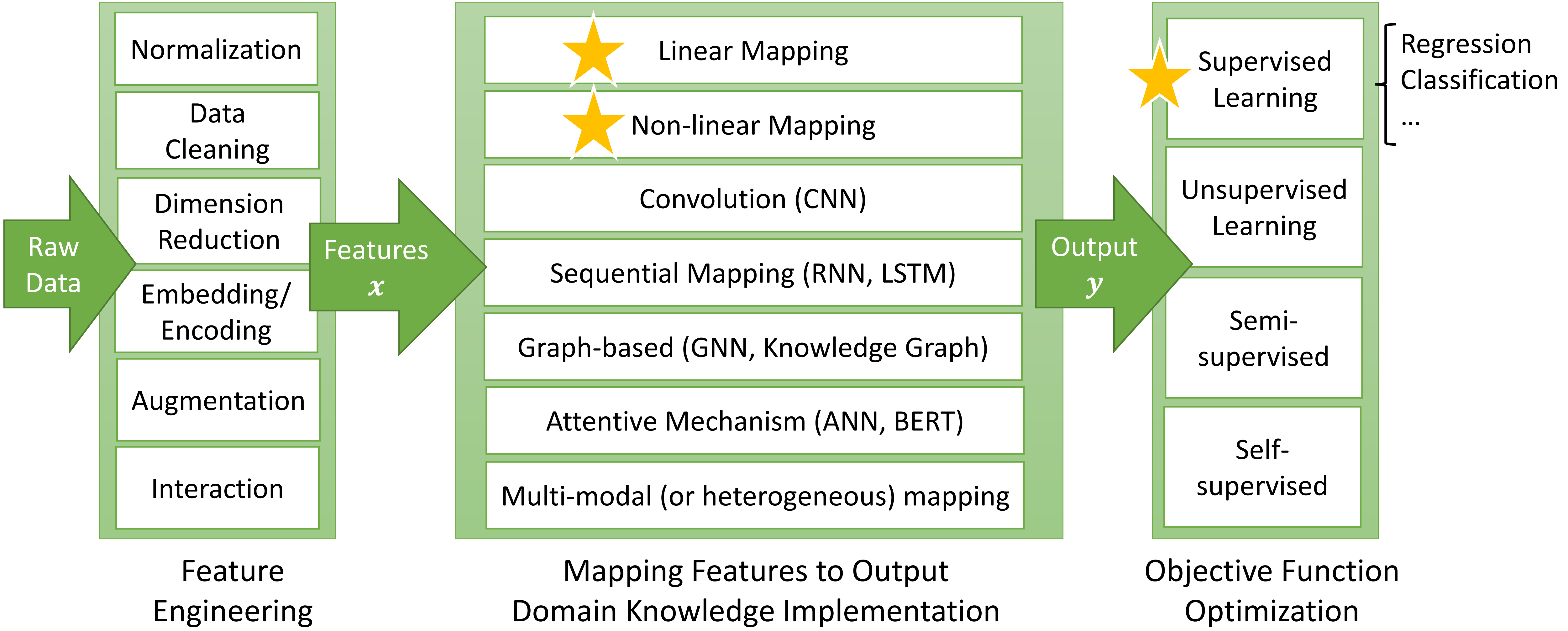
→ *Features x* →

Mapping Features to Output
Domain Knowledge Implementation
- Linear Mapping
- Non-linear Mapping
- Convolution (CNN)
- Sequential Mapping (RNN, LSTM)
- Graph-based (GNN, Knowledge Graph)
- Attentive Mechanism (ANN, BERT)
- Multi-modal (or heterogeneous) mapping

→ *Output y* →

Objective Function Optimization
- Supervised Learning
- Unsupervised Learning
- Semi-supervised
- Self-supervised

Roadmap



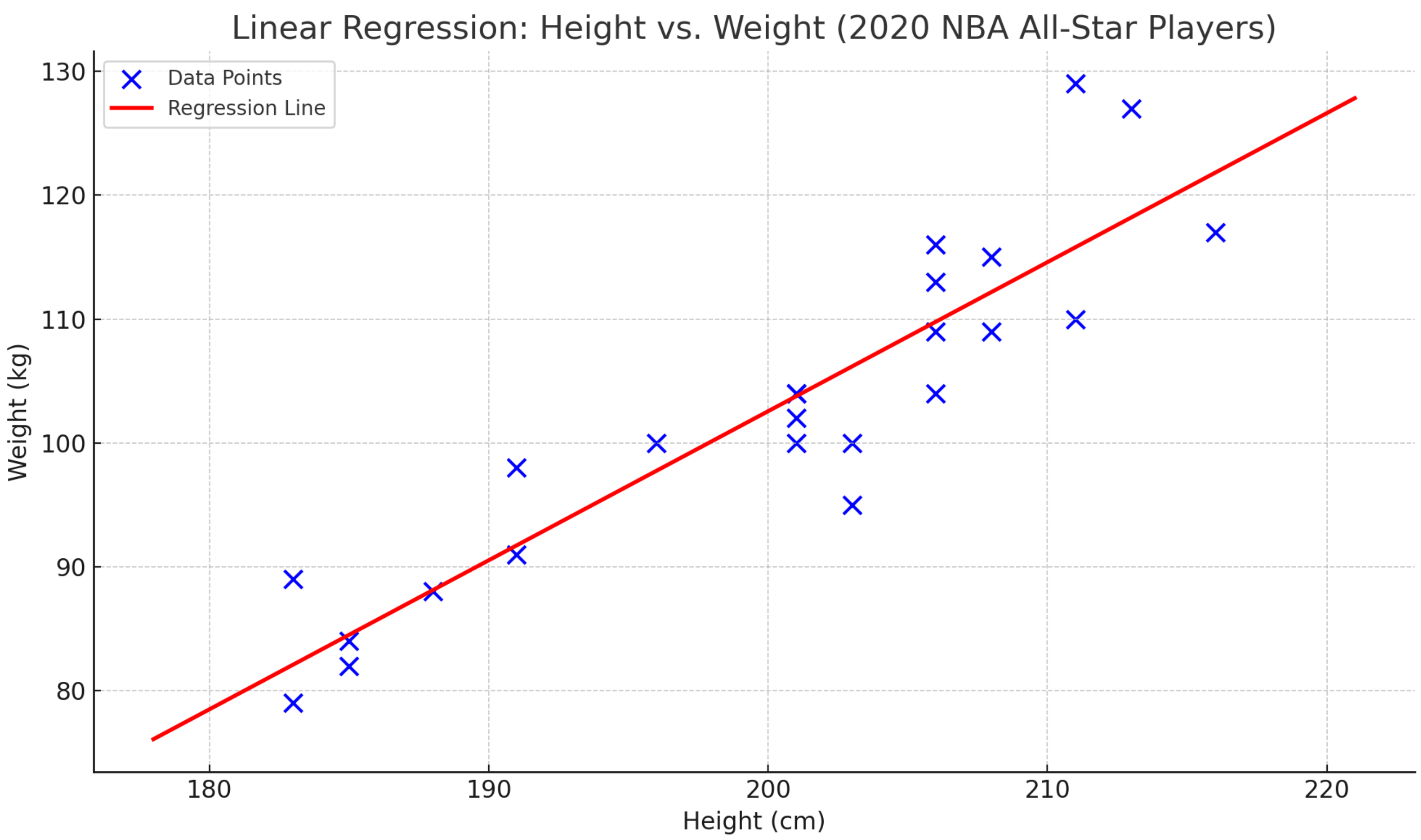
Raw Data
Normalization
Data Cleaning
Dimension Reduction
Embedding/ Encoding
Augmentation
Interaction
Features x
Linear Mapping
Non-linear Mapping
Convolution (CNN)
Sequential Mapping (RNN, LSTM)
Graph-based (GNN, Knowledge Graph)
Attentive Mechanism (ANN, BERT)
Multi-modal (or heterogeneous) mapping
Output y
Supervised Learning
Regression
Classification
…
Unsupervised Learning
Semi-supervised
Self-supervised
Feature Engineering
Mapping Features to Output
Domain Knowledge Implementation
Objective Function Optimization

Outline

- Ordinary Least Square
  - Closed Form
  - Gradient Descent
- Logistic Regression
- Overfitting
- Regularization

Example: From Heights to Weights

- Weight = f(height)



Linear Prediction

- For the i-th item/data
- $\hat{y}_i = \sum_{j=1}^{d} x_{i,j} w_j + b$

  - Weights every feature
  - The whole weighted sum maps/projects features to the predictions

- $\hat{y}_i$ is the linear prediction (weight)
- $\boldsymbol{x}_i$ is the feature vector (height and something else)
- j counts the feature dimension from 1 to d

- $\underline{w}$ and $\underline{b}$ are learnable parameters

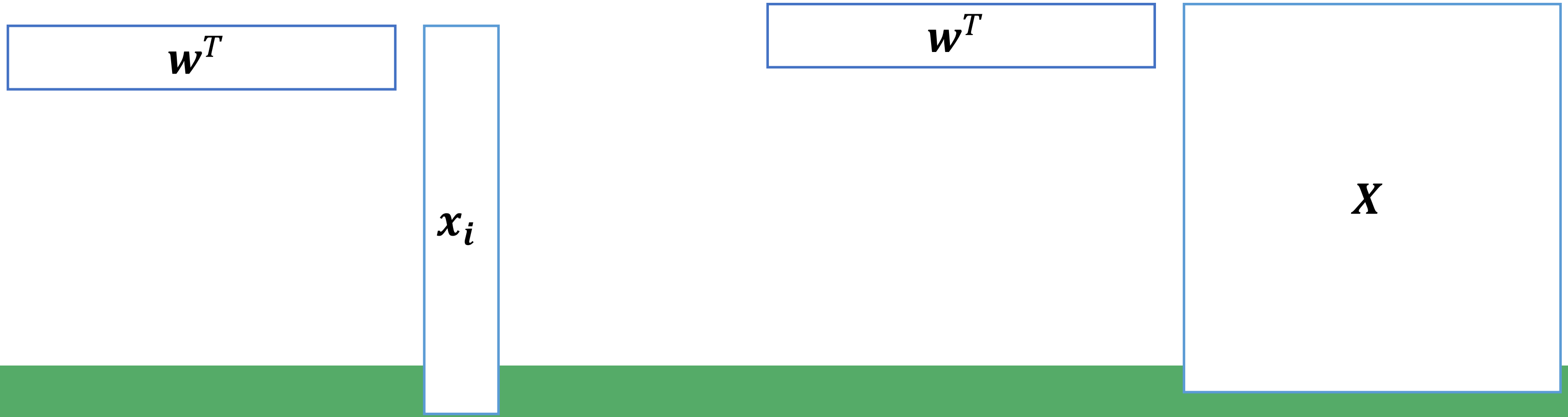
Weights

Bias ~ N(0,σ^2)

Linear Regression (cont'd)

- $\hat{y}_i = \sum_{j=1}^{d} x_{i,j} w_j + b$
- Vector form: $\hat{y}_i = \boldsymbol{w}^T \boldsymbol{x}_i + b$
- For all instances: $\widehat{\boldsymbol{Y}} = \boldsymbol{w}^T \boldsymbol{X} + \boldsymbol{b}$

Given $\boldsymbol{X}$, how to learn $\boldsymbol{w}$ and b to perfectly predict $\boldsymbol{Y}$?



Ordinary Least Square (OLS)

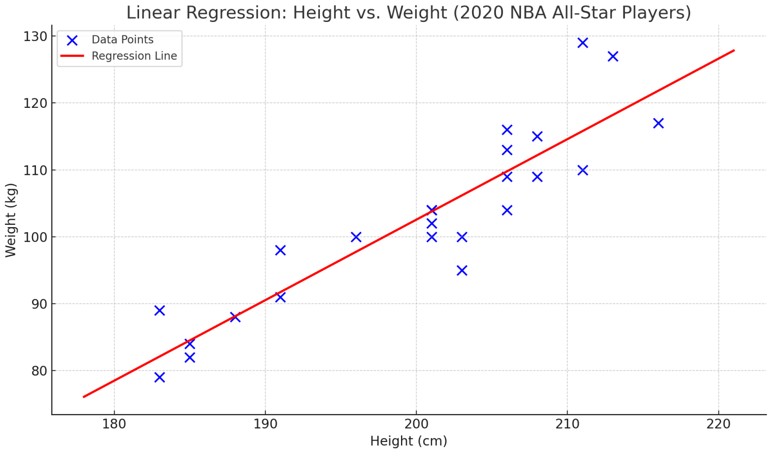
- Need to quantify how bad (or good) the prediction is
- Objective function, loss function, goodness-of-fit, etc

- For OLS, we use *square loss function*:
- $L = \sum_{\forall i} l(y_i, \hat{y}_i) = \sum_{\forall i} (\underline{y_i - \hat{y}_i})^2$
  - Errors between **empirical** values and predictions
  - Square loss additionally punishes extremely bad predictions

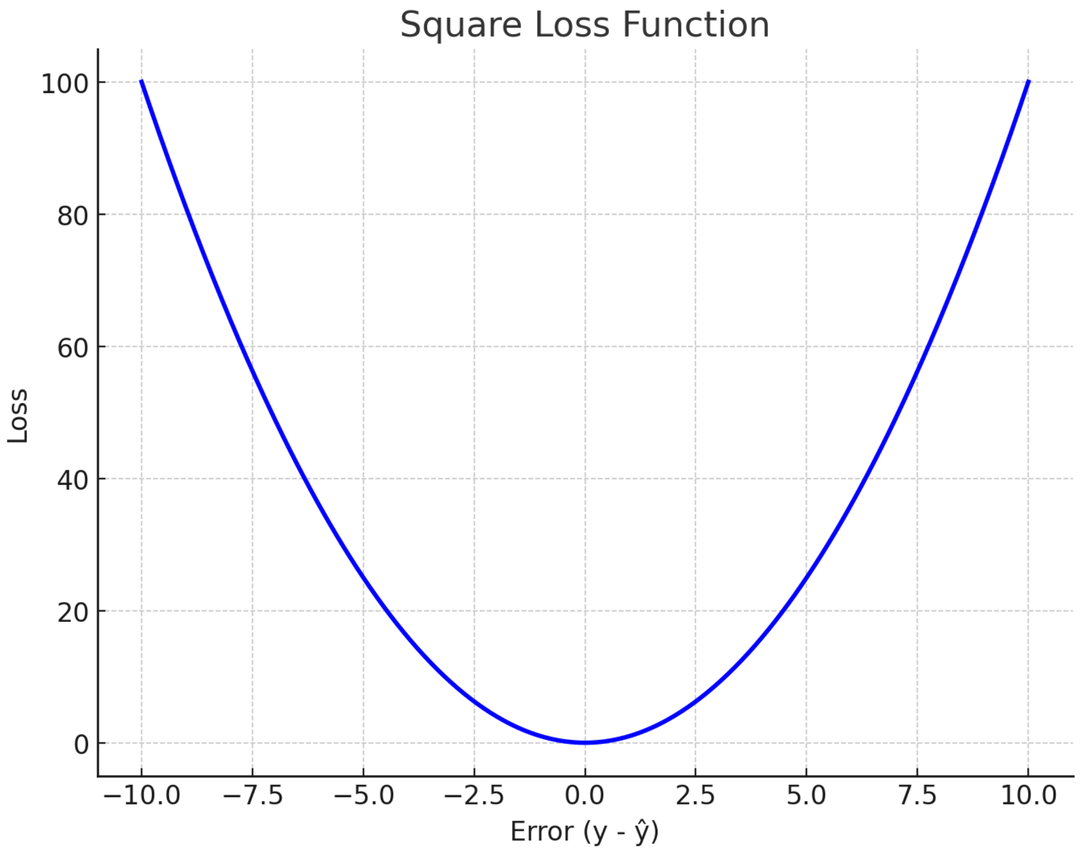
Square Loss Function

- $L = \sum_{\forall i} l(y_i, \widehat{y}_i) = \sum_{\forall i} (y_i - \widehat{y}_i)^2$

- Minimize the overall loss:
- $\min L = \min \sum_{\forall i} (y_i - \widehat{y}_i)^2$



- How?



Closed Form Solution

- For simplicity, rewrite as: $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{w}$
- Ideally, $\boldsymbol{w} = \boldsymbol{X}^{-1}\boldsymbol{Y}$

- But $\boldsymbol{X}$ could be invertible, we can play some trick to avoid this

$$\boldsymbol{X}^T\boldsymbol{Y} = \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w}$$
$$\boldsymbol{w} = \underline{(\boldsymbol{X}^T\boldsymbol{X})}^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

Semi-definite square matrix, more likely to be invertible

- Where is $\boldsymbol{b}$? [hint: its expectation is zero]
- [Math Optional] what if $\boldsymbol{X}^T\boldsymbol{X}$ is not invertible? See Moore-Penrose Pseudoinverse

Individual Instance Aspect

- Rewrite: $L = \frac{1}{2}\sum_{\forall i}(y_i - \hat{y_i})^2 = \frac{1}{2}\sum_{\forall i}(y_i - (\mathbf{w}^T x_i + b))^2$

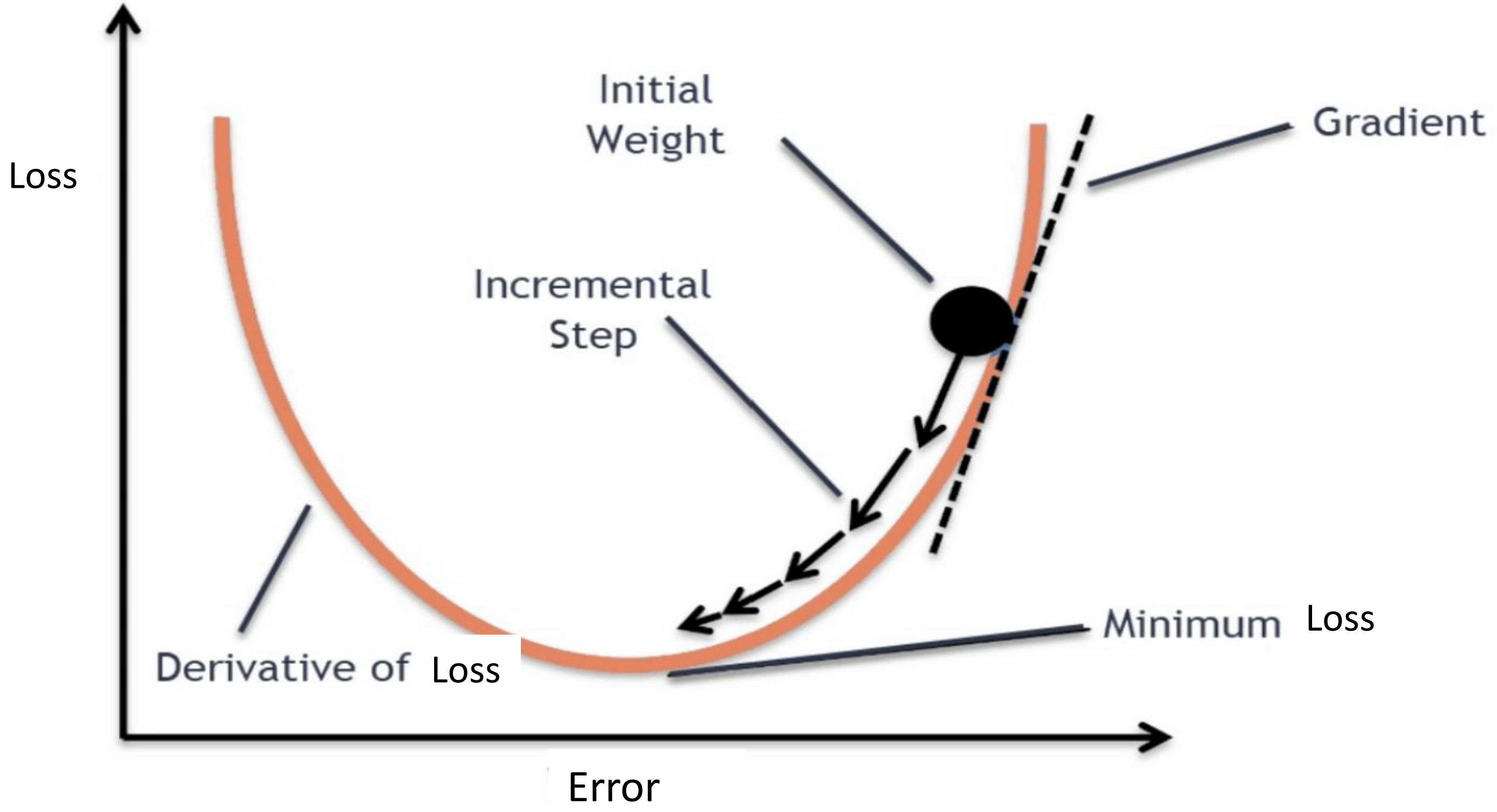
- Find *optimization direction* with **gradient** $\frac{\partial L}{\partial \mathbf{w}}$
- $\frac{\partial L}{\partial \mathbf{w}} = -x_i\left(y_i - (\mathbf{w}^T x_i + b)\right)$, for each instance x_i

- Gradient Descent: $\mathbf{w} = \mathbf{w} - \underline{\eta} \cdot \sum_{\forall i} \frac{\partial L}{\partial \mathbf{w}}$
    - Learning rate <- hyperparameter
    - Weight of gradients

Gradient Descent

How does closed form solution look like on this figure?

$$L = \frac{1}{2}\sum_{\forall i}(y_i - (\boldsymbol{w}^T x_i + b))^2$$



Step length
-> learning rate

Figure Credit: Jingbo Shan, *UCSD DSC148 lecture note*, 2023

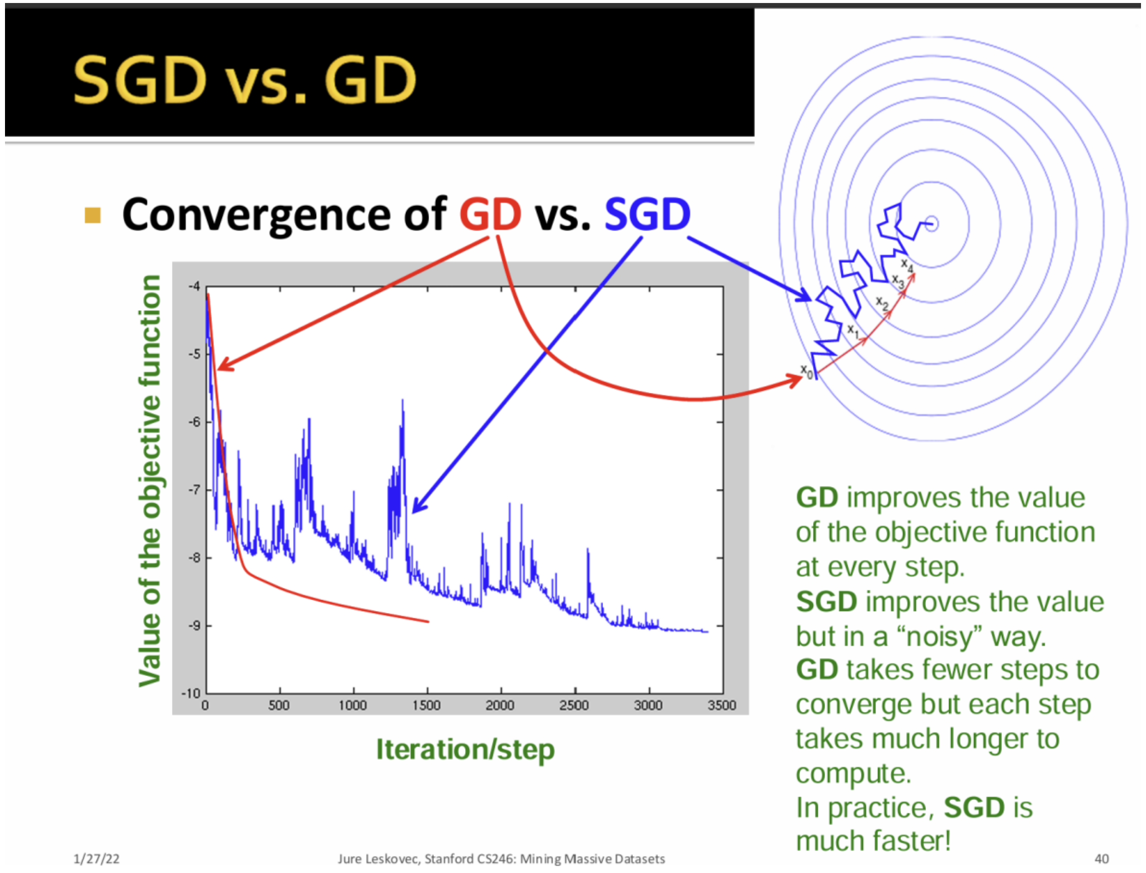
Stochastic Gradient Descent (SGD)

- Matrix operation was **slow for CPU**
  (but fast for GPU)

- **GD**: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \cdot \sum_{\forall i} \frac{\partial L}{\partial \boldsymbol{w}}$
- **SGD**: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \cdot \frac{\partial L}{\partial \boldsymbol{w}}$

- **Evaluate on all predictions** vs.
- **Evaluate on one prediction**

- **Fast convergence**



Stochastic Gradient Descent

1. Initialize $\boldsymbol{w}$ and b with small non-zero values
2. Set learning rate η
3. Loop until converge or meet maximum iteration
4. Shuffle orders of x_i
5. For all i
6. $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \cdot \frac{\partial L}{\partial \boldsymbol{w}}$
7. Adjust η

Initialization
To ensure non-zero gradient

Iteratively optimizing with SGD over every data point

Stochastic Gradient Descent

1. Initialize $\boldsymbol{w}$ and b with small non-zero values
2. Set learning rate η
3. Loop until converge or meet maximum iteration
4. Shuffle orders of x_i
5. For all i
6. $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \cdot \frac{\partial L}{\partial \boldsymbol{w}}$
7. Adjust η

Or we say "For each epoch" in deep learning era

Stochastic Gradient Descent

1. Initialize $\boldsymbol{w}$ and b with small non-zero values
2. Set learning rate η
3. Loop until converge or meet maximum iteration
4. Shuffle orders of x_i
5. For all i
6. $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \cdot \frac{\partial L}{\partial \boldsymbol{w}}$
7. Adjust η

Shuffle data reading orders to avoid sequential errors

Magnitude of Learning Rate

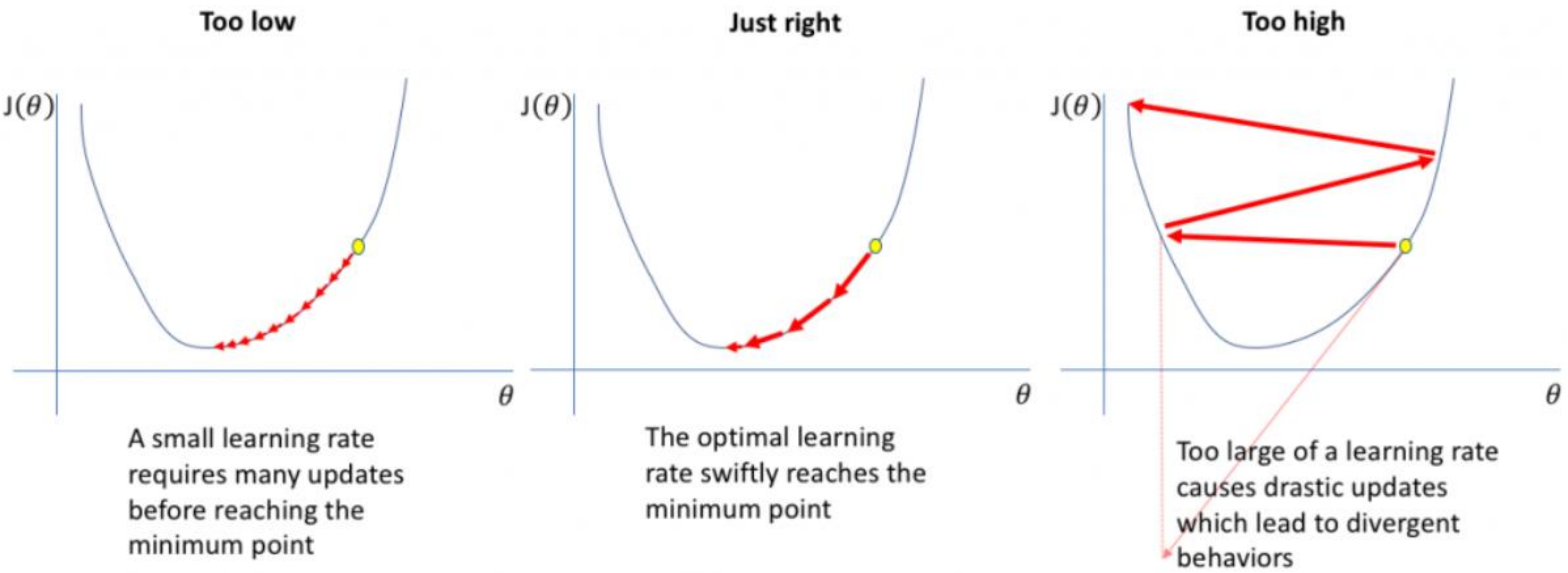


Photo Credit: 模型壓縮及優化 — Learning rate. Learning rate 介紹 | by 22 12 | Medium

Widely-used Learning Rate

- Constant
- Linear decay (over iteration number)
- Exponential decay (over iteration number)
- Adaptive learning rate (based on objective values)
- torch.optim.lr_scheduler

- Pre-defined, not learned

Other Issues

- Feature normalization
  - Z-score
    - Assumes all features follows a normal distribution
    - $x_{i,j} \sim N_j(0,1)$
  - Max-min normalization
    - $\frac{x_{i,j} - min_j}{MAX_j - min_j} \in [0,1]$
  - And more

- Non-linear relationships?
  - $\widehat{\boldsymbol{Y}} = \boldsymbol{w}^T \boldsymbol{X} + \boldsymbol{b}$
  - $\widehat{\boldsymbol{Y}} = \boldsymbol{w}_1^T \boldsymbol{X} + \boldsymbol{w}_2^T \boldsymbol{X}^T \boldsymbol{X} + \boldsymbol{b}$

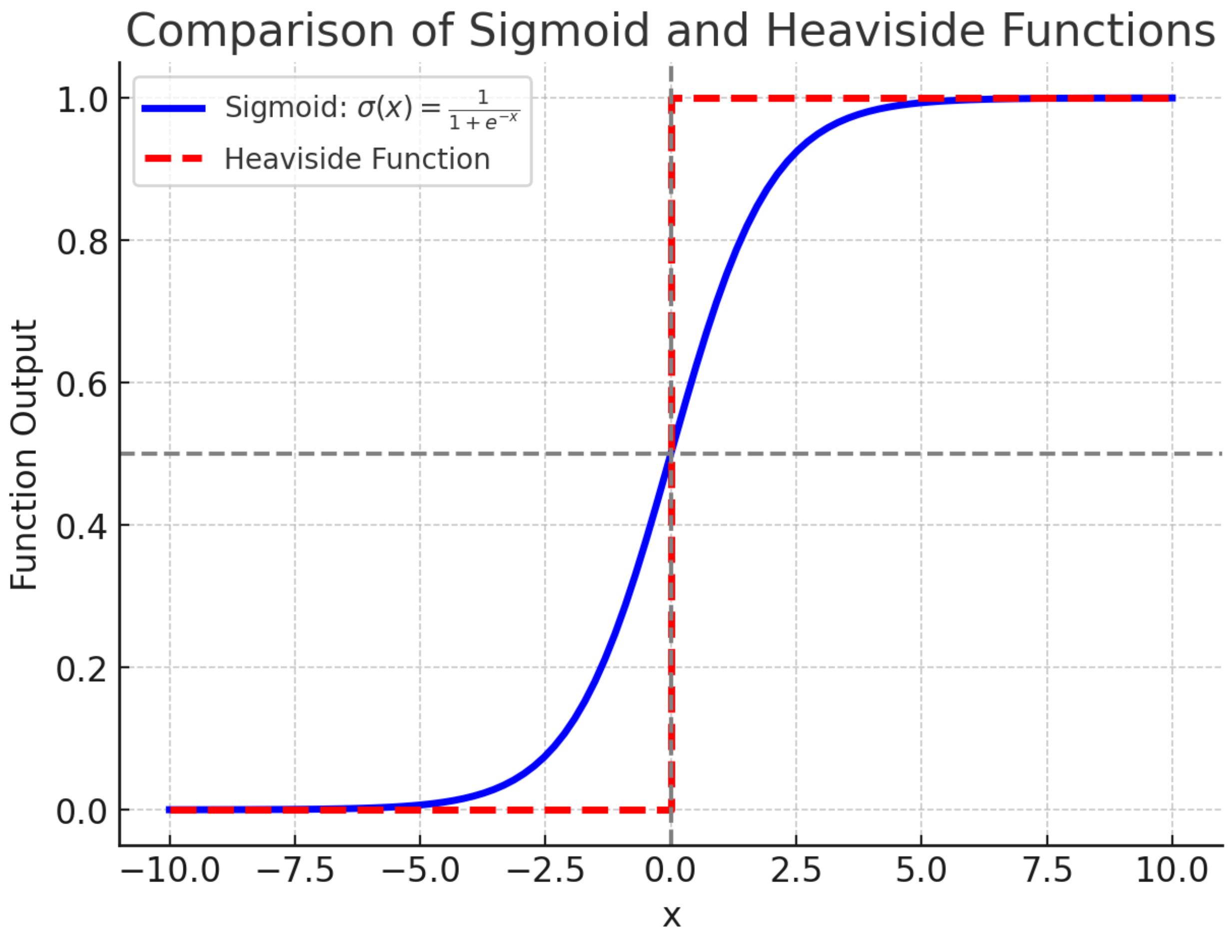
Add high-order terms

Outline

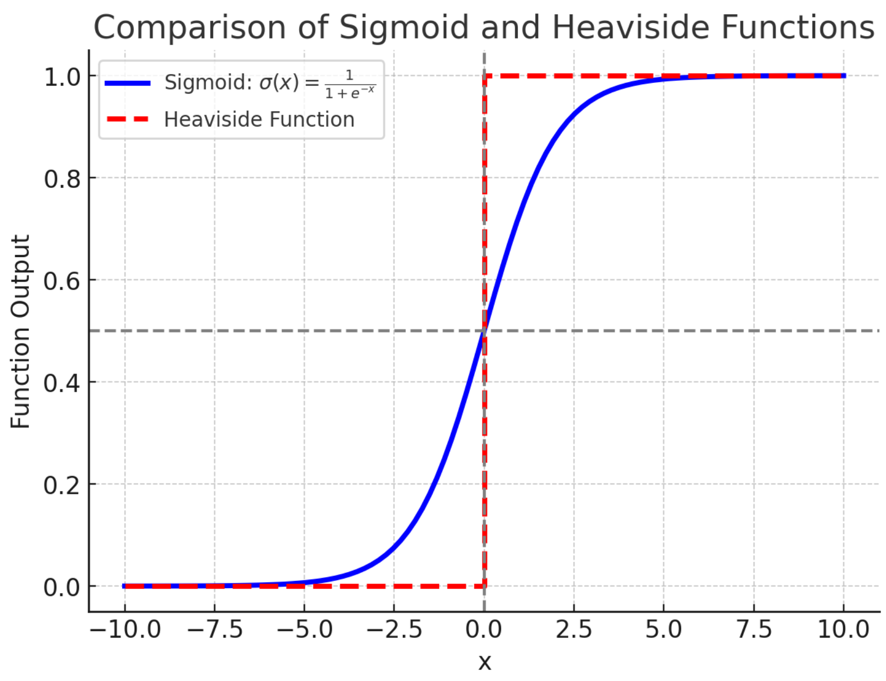
- Ordinary Least Square
- Logistic Regression
    - Sigmoid function
    - Cross-Entropy
- Overfitting
- Regularization

Logistic Regressions for Classification

- Binary labels
  - $y_i \in [0,1]$
  - Is he heavier than 70 kg?
  - Is it a photo of a cat?
- $\hat{y}_i \in \{0,1\}$
- Heaviside function
- Sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$



Logistic Regression



- For the i-th item/data
- $\hat{y}_i = \sigma(\boldsymbol{w}^T \boldsymbol{x_i} + b)$

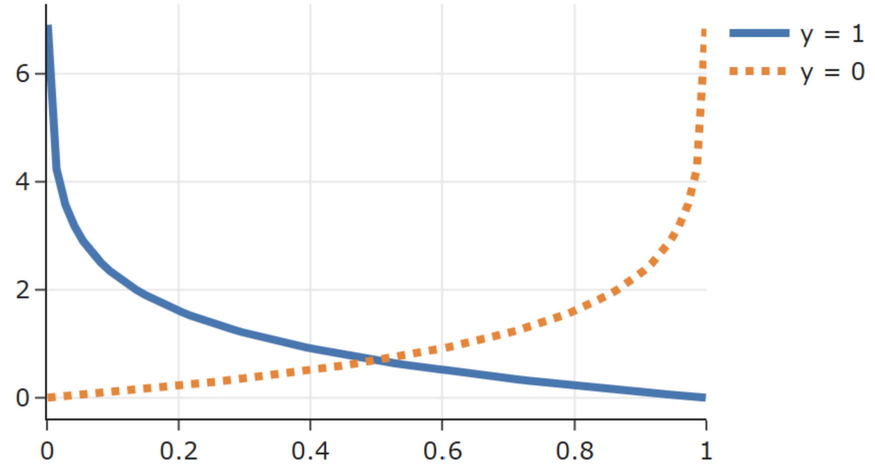
- $\hat{y}_i$ is the linear prediction (weight)
- $\boldsymbol{x}_i$ is the feature vector (height and something else)
- j counts the feature dimension from 1 to d
- σ is the sigmoid function

- $\boldsymbol{w}$ and b are learnable parameters

Objective Function

- Logistic loss function (a.k.a. log loss function)
  - A Negative log-likelihood (NLL) function
  - $L = \sum_{\forall i} l(y_i, \hat{y}_i) = \sum_{\forall i} -[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)]$

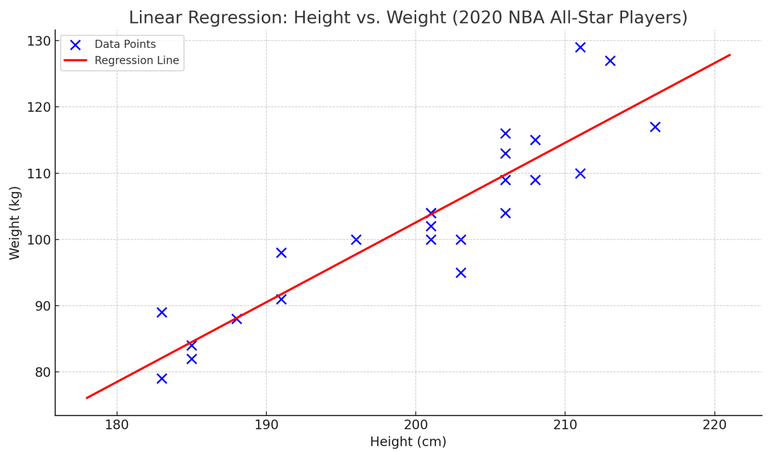
	$\boldsymbol{y_i \log \hat{y}_i}$	$\boldsymbol{(1 - y_i)\, log(1 - \hat{y}_i)}$
Positive case $(\mathbf{y_i = 1})$	$1 \cdot \log \hat{y}_i$	$0 \cdot \log(1 - \hat{y}_i)$
Negative case $(\mathbf{y_i = 0})$	$0 \cdot \log \hat{y}_i$	$1 \cdot \log(1 - \hat{y}_i)$

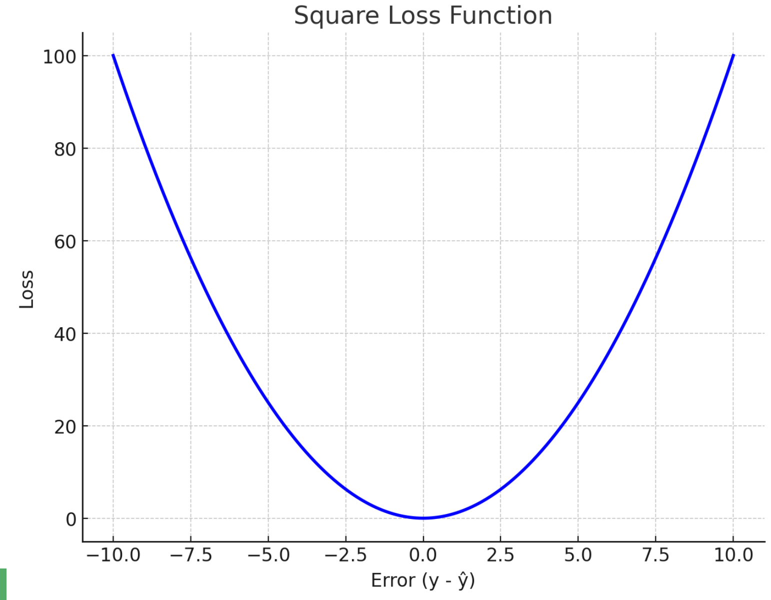


- Gradient $\frac{\partial L}{\partial \boldsymbol{w}} = (\hat{y}_i - y_i)\boldsymbol{x_i}$ [homework; hint: chain rule]

$\Rightarrow$SGD

OLS

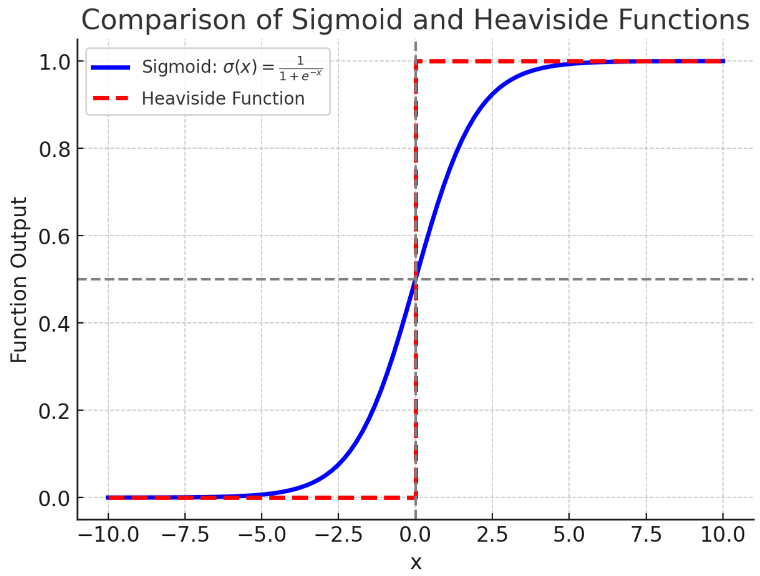


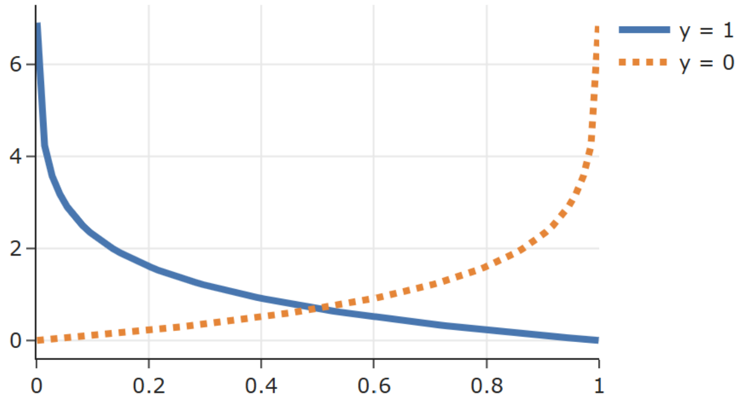


Mapping features to predictions with assumed distributions

Gradients work on **loss functions**

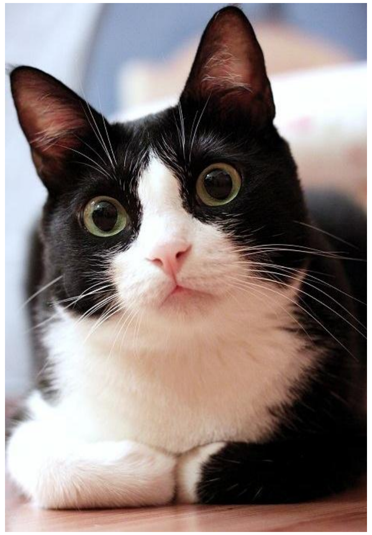
Logistic Regression





Extend to K-class Classification

- Intuition 1
  - Ask K questions: is this instance the k-th class?
  - K binary classifiers
  - One-versus-all
- Intuition 2
  - Ask C_2^K questions: is this instance the k-th or the other specific class?
  - C_2^K binary classifiers
  - One-versus-another
- Both inefficient

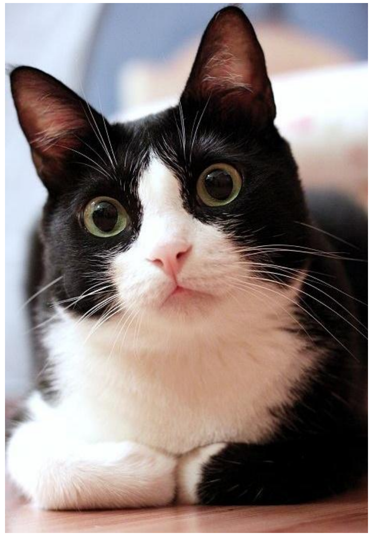


Cat? Dog? Human?

Extend to K-class Classification (cont'd)

- Multi-class Classification
  - $y_{i,k} \in [0,1]$
  - $\widehat{y_{i,k}} \in [0,1]$
  - $k = 1..\mathrm{K}$

- **Cross-Entropy Loss**
  - $L^{CE} = \sum_{\forall i} l(y_i, \widehat{y_i}) = \sum_{\forall i} \sum_{\forall k} (-y_{i,k} \log \widehat{y_{i,k}})$
  - Essential in modern deep models!



1	Cat
0	Dog
0	Human

0	Cat
1	Dog
0	Human

Outline

- Ordinary Least Square
- Logistic Regression
- Overfitting
  - Definition
  - Cross Validation
- Regularization

Overfitting?

- Which model is the best fitting?

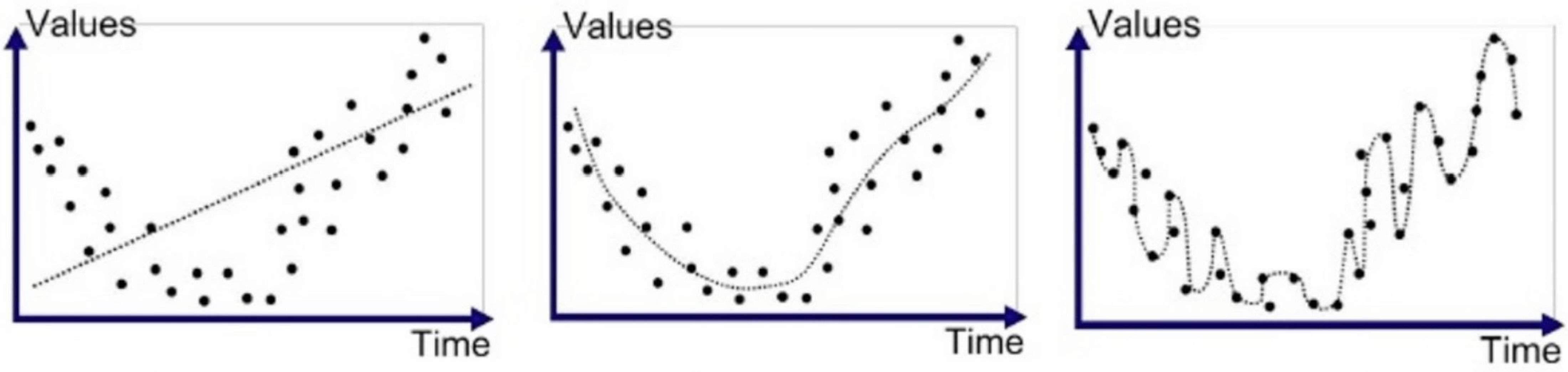
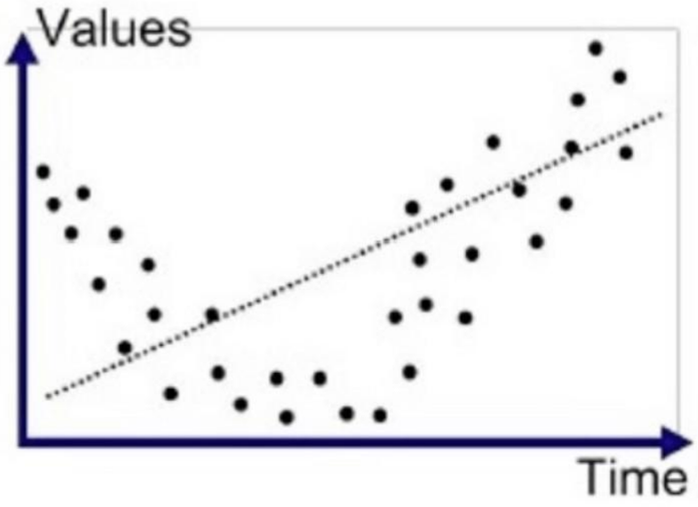


Figure Credit: Jingbo Shan, *UCSD DSC148 lecture note*, 2023

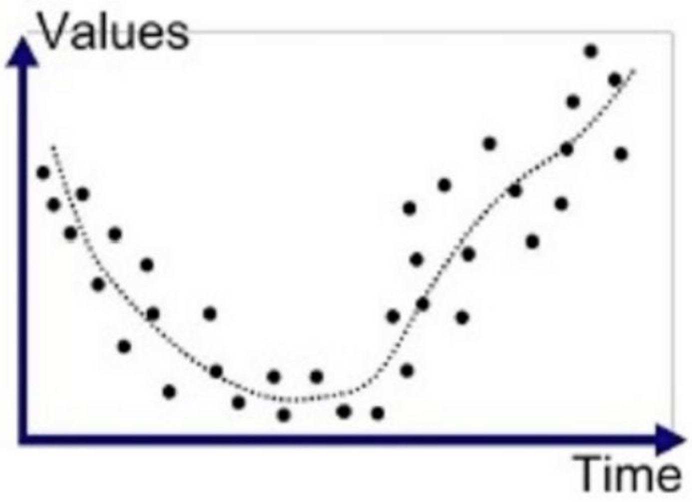
Overfitting?

- Which model is the best fitting?



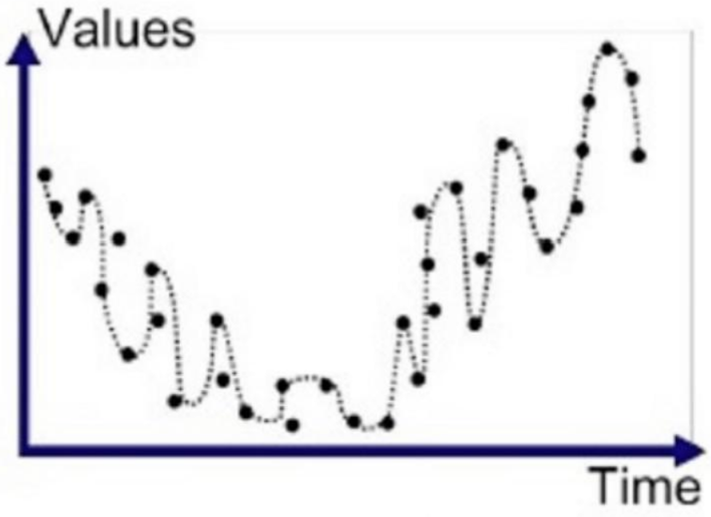
Underfitting

Not catching the trend
Generalized poorly



Good fit

Robust to future predictions



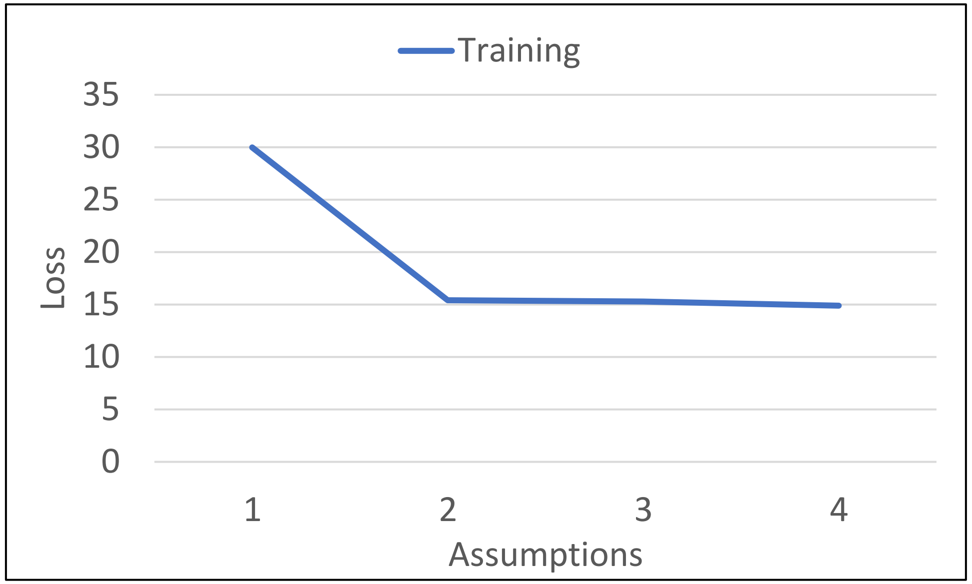
Overfitting

Generalized poorly

Figure Credit: Jingbo Shan, *UCSD DSC148 lecture note*, 2023

Further reading: Variance and Bias

Observation

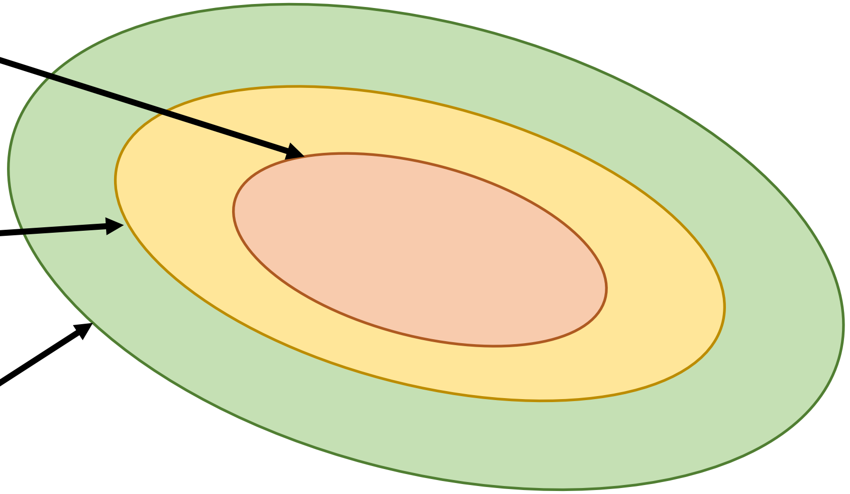


Assumption 1 $\hat{y} = b + w_1 x$

Assumption 2 $\hat{y} = b + w_1 x + w_2 x^2$

Assumption 3 $\hat{y} = b + w_1 x + w_2 x^2 + w_3 x^3$

Assumption 4 $\hat{y} = b + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$



Function Space

Complex models yield lower training loss

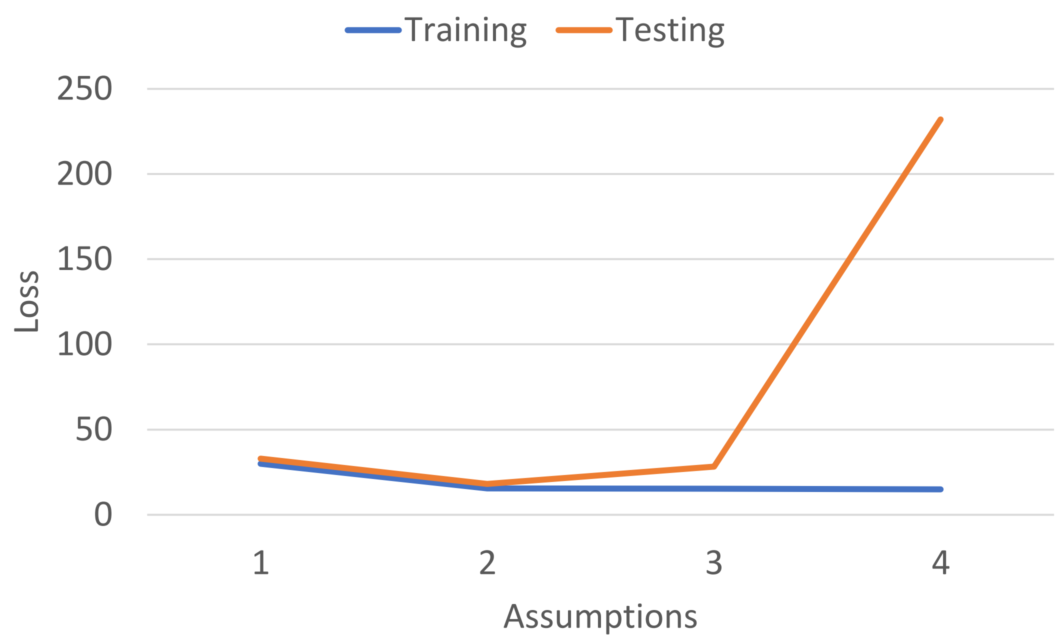
Observation

Assumption 1 $\hat{y} = b + w_1 x$

Assumption 2 $\hat{y} = b + w_1 x + w_2 x^2$

Assumption 3 $\hat{y} = b + w_1 x + w_2 x^2 + w_3 x^3$

Assumption 4 $\hat{y} = b + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$

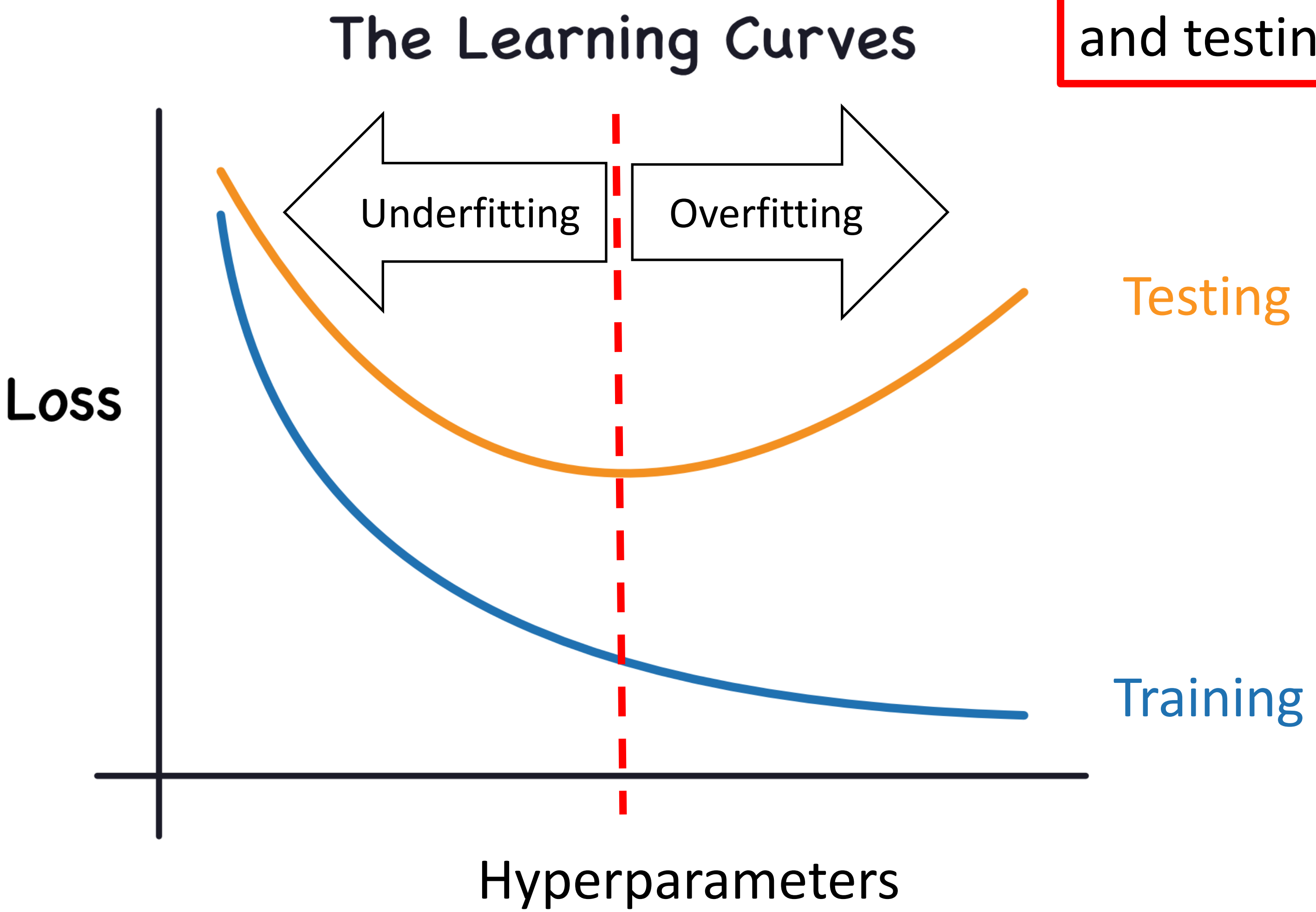


- Complex models yield lower training loss
- But does not always yield better testing loss

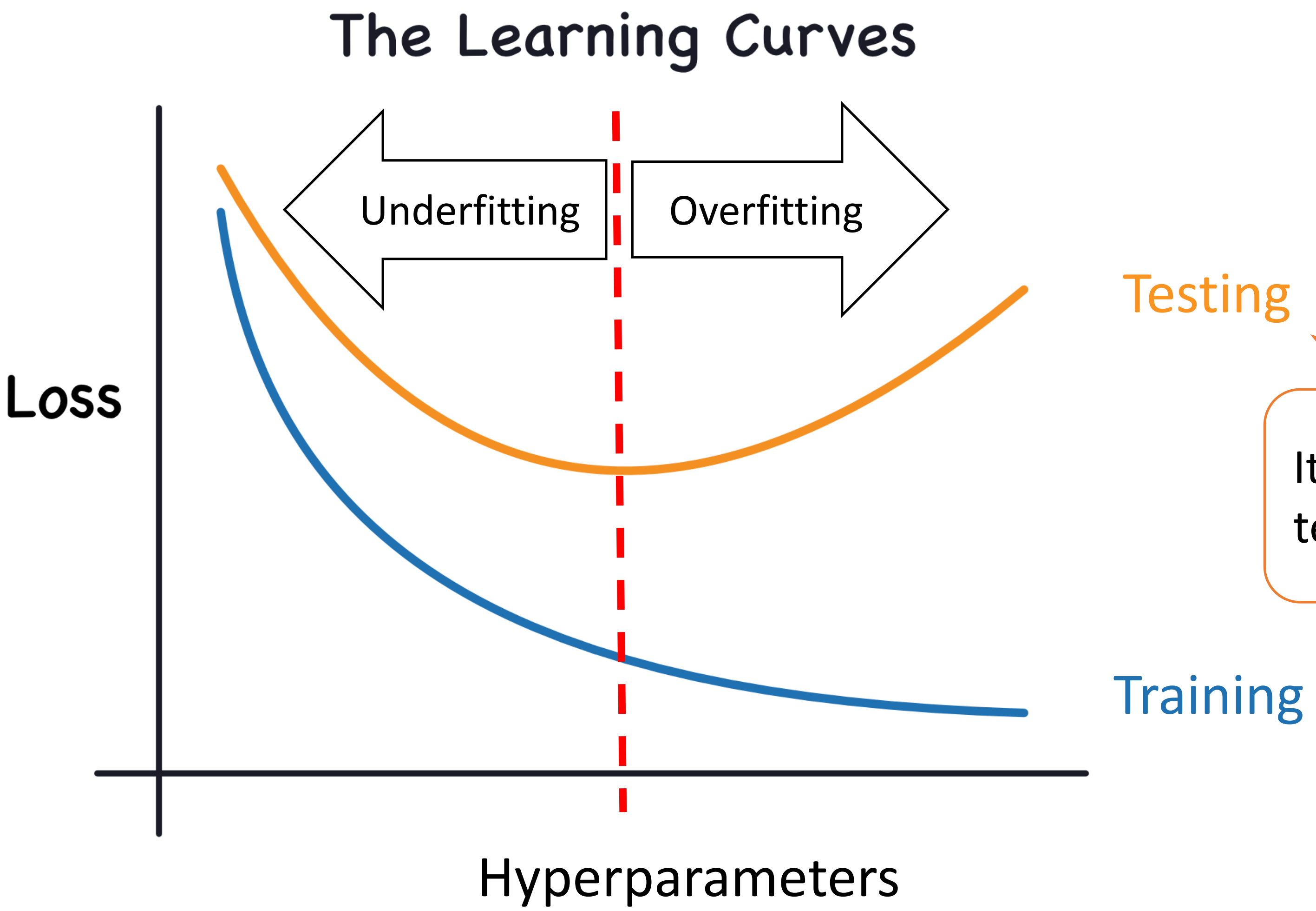
$\Rightarrow$ **Overfitting**

Selecting Hyperparameters

<u>Strategy:</u>
Plot the learning curves and find the sweet spot between training and testing losses



Spying The Future



It is not fair to preview testing data!

Validation Set

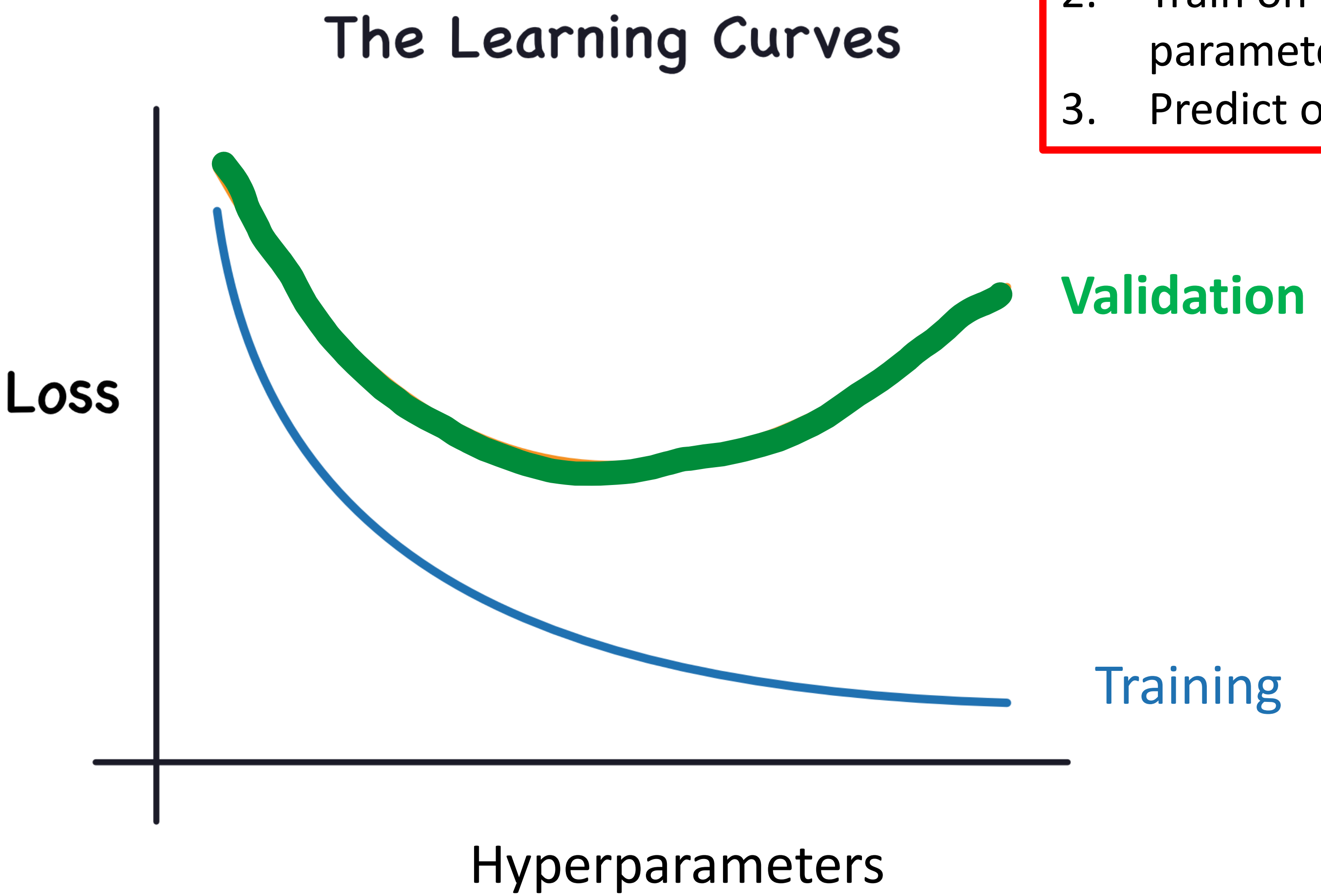
- From train-test split

- To train-validation-test split

Validation data!

Like a pseudotesting set

No More Spying

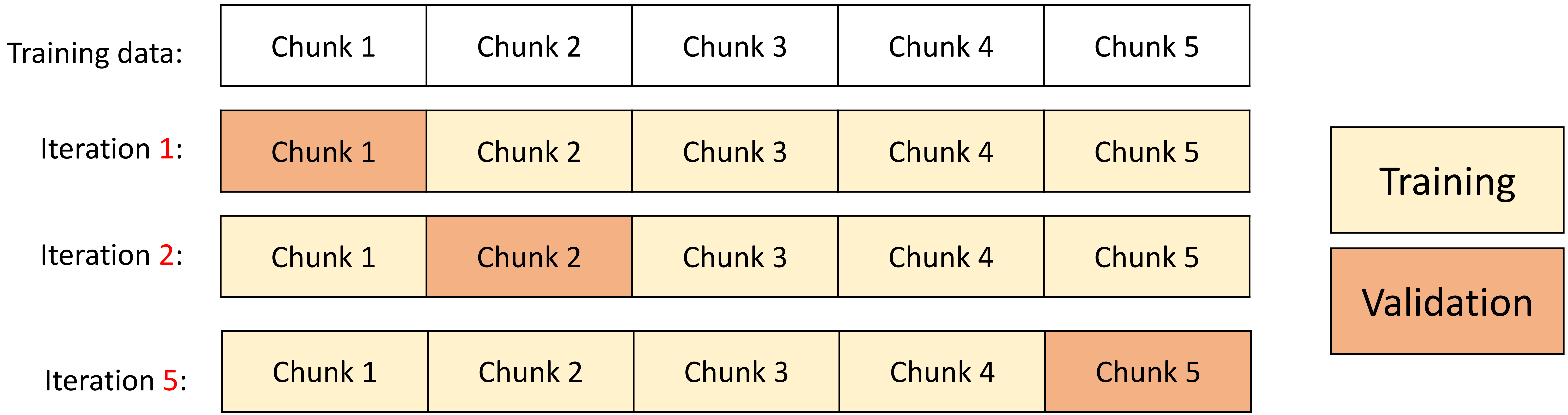


Strategy:

1. Plot the learning curves and find the sweet spot between training and testing losses
2. Train on entire training set with best parameter
3. Predict on testing data

Choosing Validation Set

- Using only one validation set could be biased
- **k-fold cross validation**
    - **k=5:**



Steps of 5-fold CV

With these training skills:
- Avoid spying testing data
- Alleviate overfitting
- Alleviate hyperparameter biases

1. **Shuffle the training set**
2. **Evenly split data into 5 chunks**
3. **For k = 1 to 5**
4. **Set chunk k to validation and the others to training**
5. **Fit on training set**
6. **Evaluate on validation set**

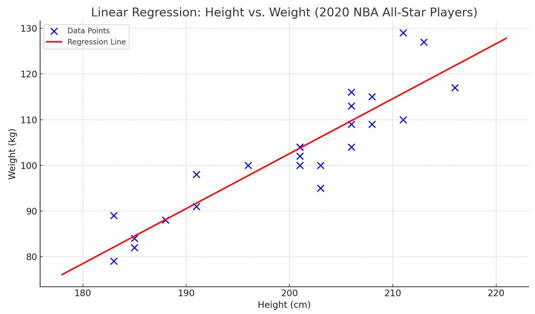
- What you can do with CV:
  - Identify optimal hyperparameters (e.g., the best epoch)
  - Tune hyperparameters based on the five results (including the observation of overfitting)
  - Train the model with the entire training set based on the best-fit parameters
  - Output average performance

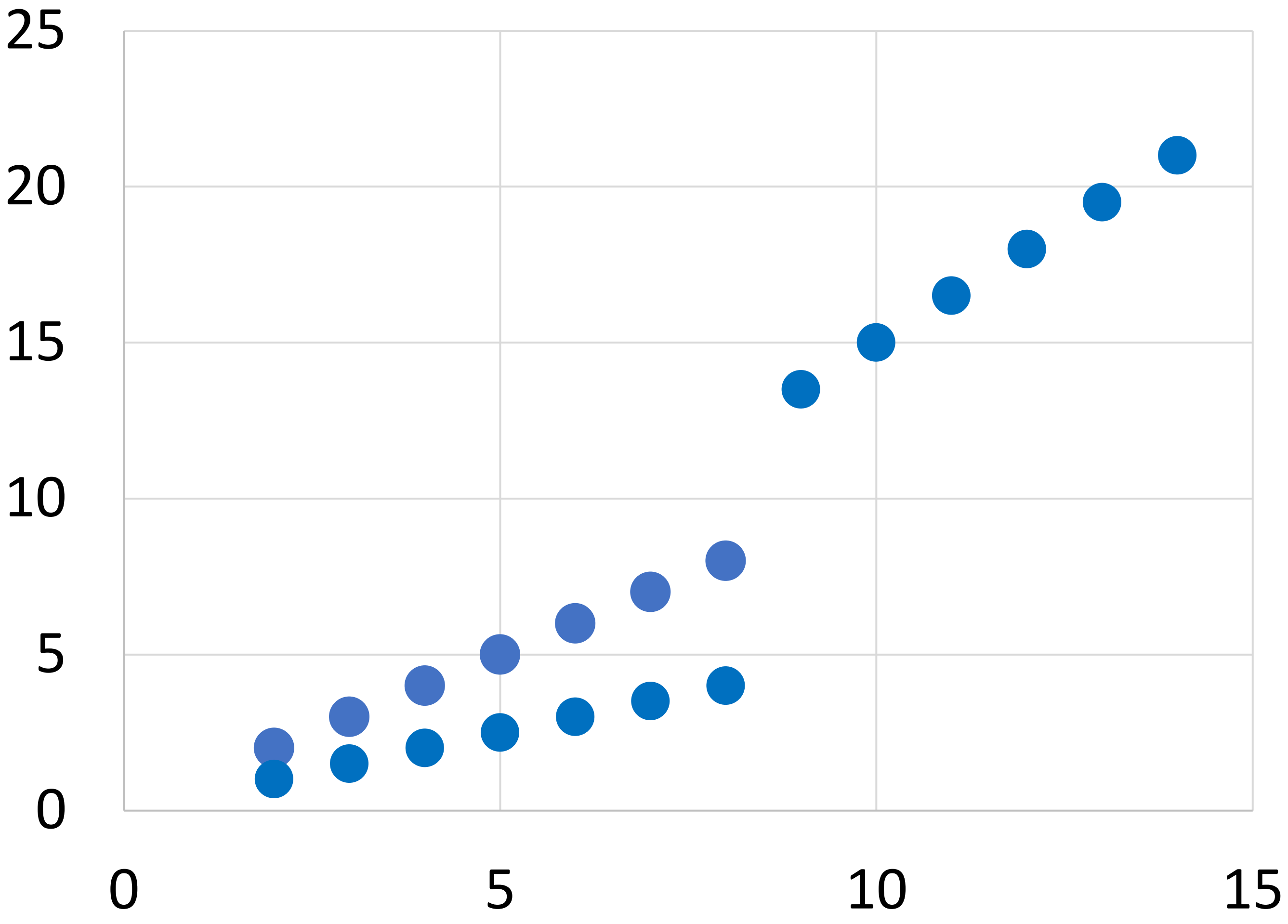
Why people use k=5? What are the trade-offs?

Outline

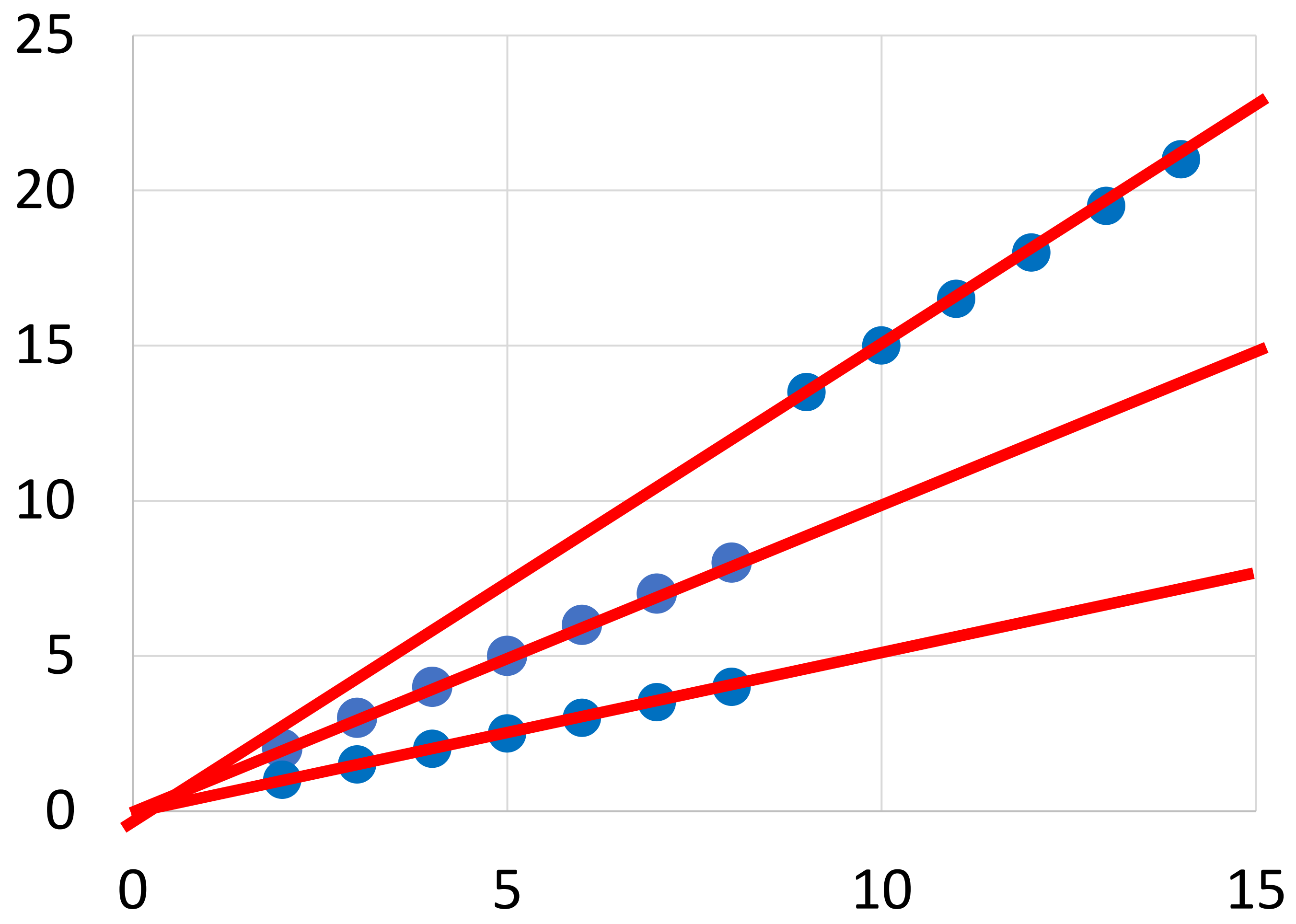
- Ordinary Least Square
- Logistic Regression
- Overfitting
- Regularization
  - L2 Norm – Ridge Regression
  - L1 Norm – Lasso Regression

Hidden Factors





Hidden Factors



Sensitive to Changes

- Recall OLS
  - $L^{OLS} = \sum_{\forall i}(y_i - \hat{y_i})^2 = \sum_{\forall i}(y_i - (\boldsymbol{w}^T x_i + b))^2$

- $\hat{y_i} = (\boldsymbol{w}^T x_i + b)$
- $x'_i \leftarrow x_i + \Delta$
- $\widehat{y'_i} \leftarrow \hat{y_i} + \boldsymbol{w}^T \Delta$

$\boldsymbol{w}$	$\boldsymbol{w}^T \Delta$ ($\Delta = 0.1$)
1	+0.1
10	+1
100	+10
1000	+100

- Very sensitive to tiny changes (or noises) with large $\boldsymbol{w}$

Why do we regularize $\boldsymbol{w}$ but not b? hint: smoothness

Ridge Regression

- Ridge regression adds an L2 norm
  - $L^{RR} = \sum_{\forall i}(y_i - (\boldsymbol{w}^T x_i + b))^2 + \lambda \cdot \|\boldsymbol{w}\|_2^2$

  $\Rightarrow$ Penalize high value of $\boldsymbol{w}$

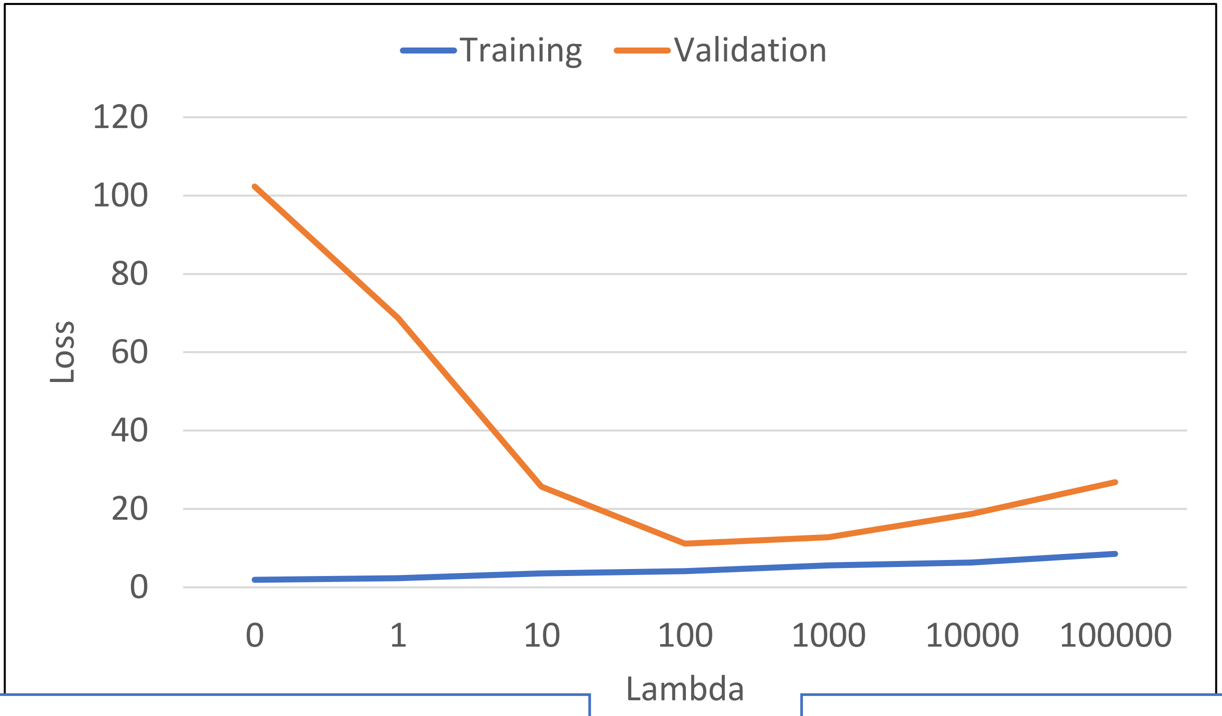
  Hyperparameter weighting the regularization term

- y_i is *insensitive* to small changes Δ with smaller $\boldsymbol{w}$
- **Robust to noises!**

Why do we regularize $\boldsymbol{w}$ but not b? hint: smoothness

How Smooth?

$$L^{RR} = \sum_{\forall i} (y_i - (\boldsymbol{w}^T x_i + b))^2 + \lambda \cdot \|\boldsymbol{w}\|_2^2$$



Merely penalize $\boldsymbol{w}$ => sensitive to noises

Strongly penalize $\boldsymbol{w}$ => almost horizontal line

Lasso Regression

- OLS
  - $L^{OLS} = \sum_{\forall i}(y_i - \hat{y}_i)^2 = \sum_{\forall i}(y_i - (\boldsymbol{w}^T x_i + b))^2$

- Ridge regression adds an L2 norm
  - $L^{RR} = \sum_{\forall i}(y_i - (\boldsymbol{w}^T x_i + b))^2 + \lambda \cdot \|\boldsymbol{w}\|_2^2$

- Lasso regression adds an absolute value, namely L1 norm
  - $L^{LR} = \sum_{\forall i}(y_i - (\boldsymbol{w}^T x_i + b))^2 + \lambda \cdot \|\boldsymbol{w}\|_1$

Effect

- Lasso **zero**-outs the weights of ineffective features
  - ⇒ Feature selection
  - ⇒ Better at high dimensional and sparse datasets

Classifying cat or human

$\boldsymbol{w}$

w1	w2	**w3=0**	w4

$\boldsymbol{x}$

Standing legs
Sound
Gender
Tails

Comparisons

	Ridge	Lasso
Pros	1. Closed form solution exists [optional homework] 2. Robust to noises	1. Zero-outs redundant features 2. Built-in feature selection 3. Robust to sparse data
Cons	1. Not able to zero-out redundant features (close to zero still)	1. Aggressive on correctness but may lose generalizability

Takeaway

- Many functions
  - Linear prediction and sigmoid function
  - OLS and negative log likelihood
- Gradient descent
- Overfitting
  - K-fold cross validation
  - Regularization

References

- Jingbo Shan, *UCSD DSC148 lecture note*, 2023
- ML Lecture 1: Regression Case Study. Hung-Yi Lee
- 模型壓縮及優化 — Learning rate. Learning rate 介紹 | Medium
- GPT-o3-mini (be aware of incorrectness)