CoE202
Fundamentals of Artificial intelligence
<Big Data Analysis and Machine Learning>

Generalization

Prof. Young-Gyu Yoon
School of EE, KAIST

Contents

Recap: Linear classification

For a data set $\mathcal{D} = \{(\vec{x_1}, y_1), (\vec{x_2}, y_2), \cdots, (\vec{x_N}, y_N)\}$

Seeks a function $f(\vec{x}; \theta_0, \vec{\theta}) = \vec{\theta} \cdot \vec{x} + \theta_0$

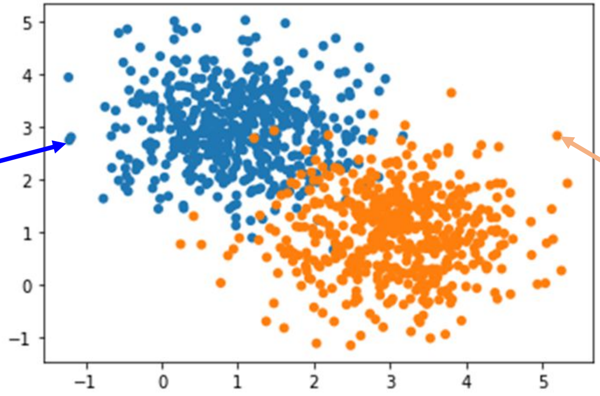
Such that a loss function

$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}(y_i\, log(f(x_i)) + (1-y_i)log(1-f(x_i)))$

is minimized

$\overrightarrow{x_{10}} = \begin{bmatrix} -1.21 \\ 2.85 \end{bmatrix}$

$y_{10} = 1$



$\overrightarrow{x_{98}} = \begin{bmatrix} 5.17 \\ 2.91 \end{bmatrix}$

$y_{98} = 0$

Recap: Softmax function

Standard softmax

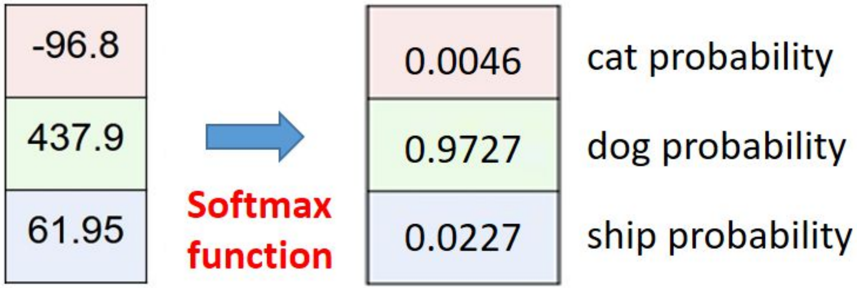
$$\boldsymbol{\sigma}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Softmax with a smoothness parameter

$$\boldsymbol{\sigma}(z_i) = \frac{e^{\beta z_i}}{\sum_{j=1}^{K} e^{\beta z_j}}$$

Vector representation of above

$$\boldsymbol{\sigma}(\vec{z}) = \frac{e^{\beta \vec{z}}}{\sum_{j=1}^{K} e^{\beta z_j}}$$

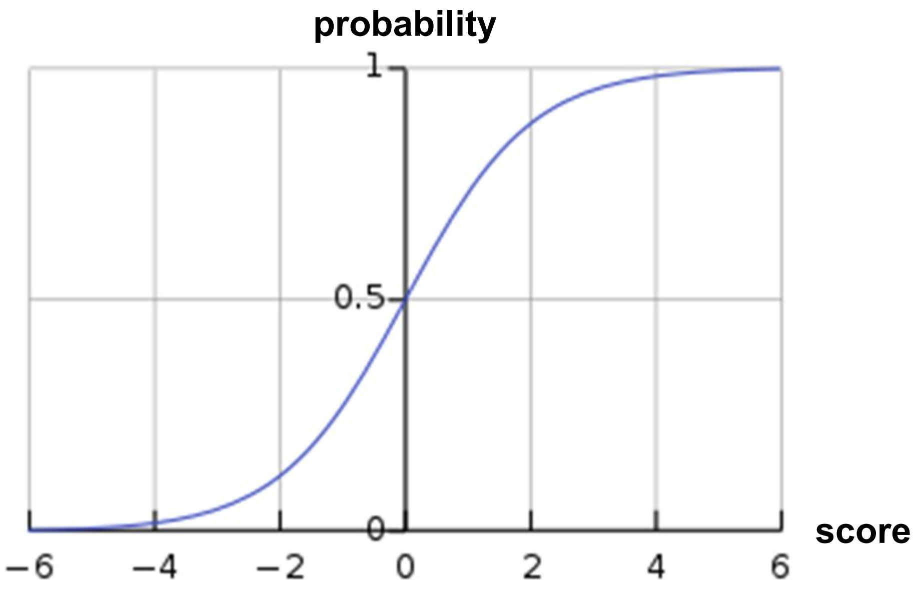


- This softmax function allows us to convert a meaningless set of scores to a set of probabilities (which is a lot more interpretable!)

Recap: sigmoid function

Sigmoid function

$$S(z) = \frac{e^z}{e^z+1} = \frac{1}{1+e^{-z}}$$



- Sigmoid function
  - is monotonic
  - maps all real values to (0,1) → can be used to represent a probability

Recap: Binary cross entropy loss

- **Binary cross entropy (BCE) loss**

$$\mathcal{L} = -(y\, log(f(x)) + (1-y)log(1-f(x)))$$

  - measures the similarity of two probabilities
  - Consider two cases
    - When $y_i=0$, then $f(x_i)=0$ & $y_i=1$, then $f(x_i)=1$
    - When $y_i=0$, then $f(x_i)=1$ & $y_i=1$, then $f(x_i)=0$

- **BCE loss of multiple data samples**

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}(y_i\, log(f(x_i)) + (1-y_i)log(1-f(x_i)))$$

Recap: Cross-entropy loss

- **Cross entropy (CE) loss for K-class classification**

$$\mathcal{L} = -\sum_{j=1}^{K} y_j \, log(f(x)_j)$$

Why is this here?

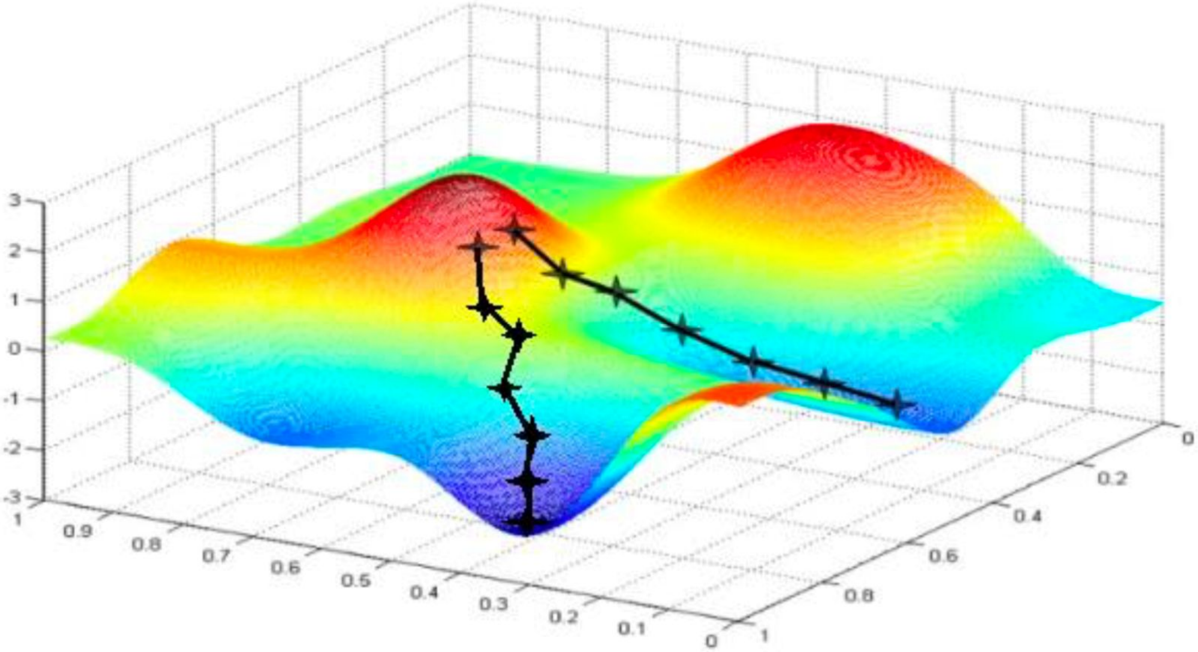
- **When we have multiple data samples, we take an average**

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij} \, log(f(x_i)_j)$$

class index

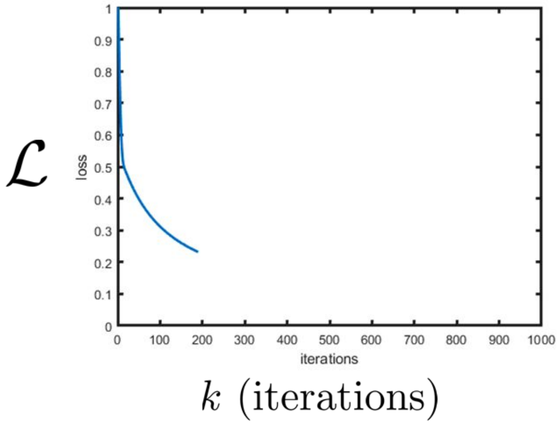
sample index

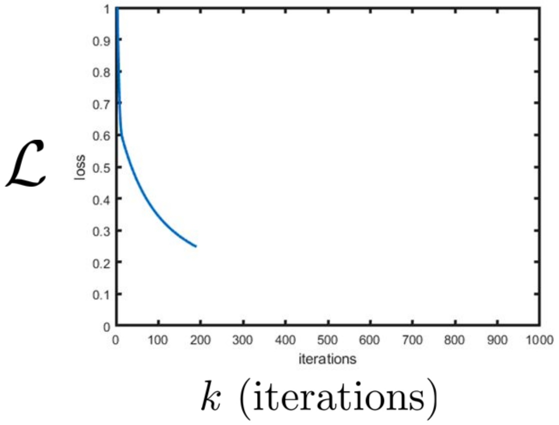
Recap: Gradient Descent for linear classification

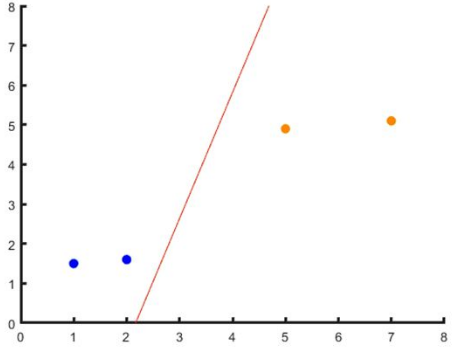


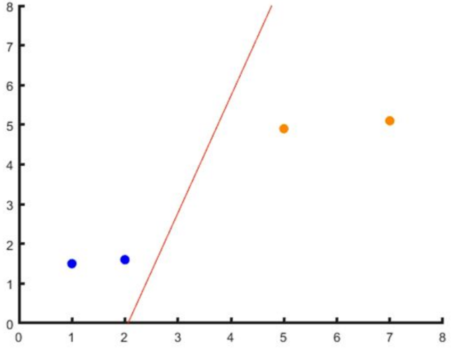
Two trials with different initial parameters

Recap: Gradient Descent for linear classification



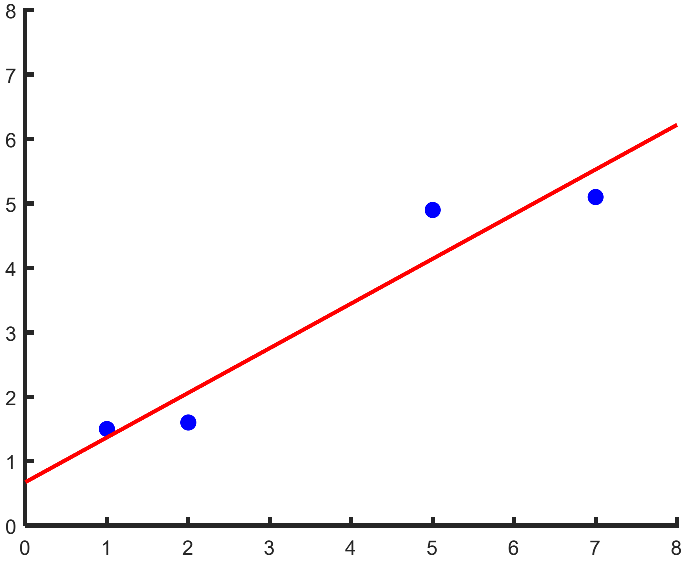




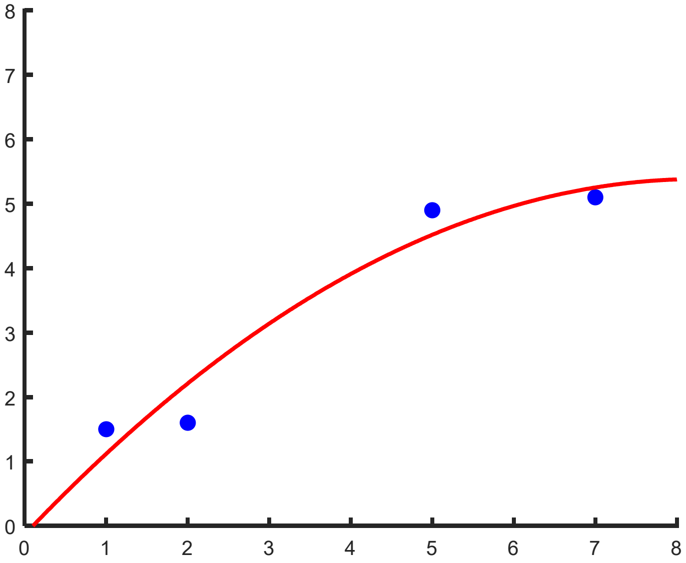


Two trials with different initial parameters

Model selection problem



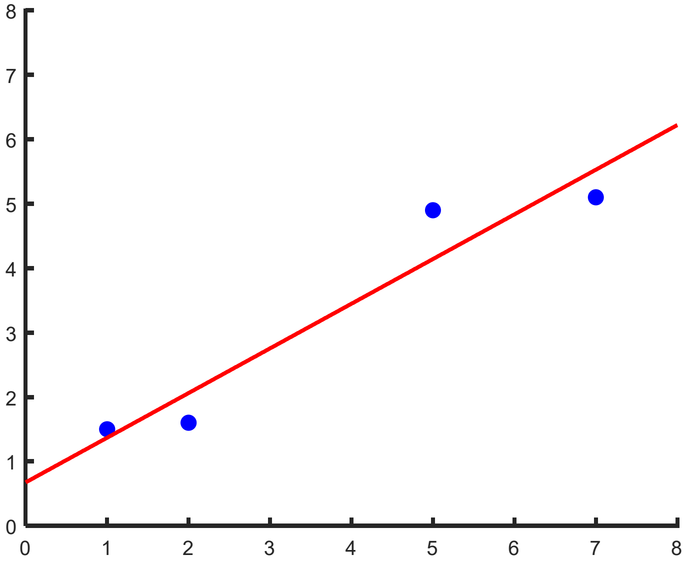
$$f(x) = 0.6934x + 0.6747$$



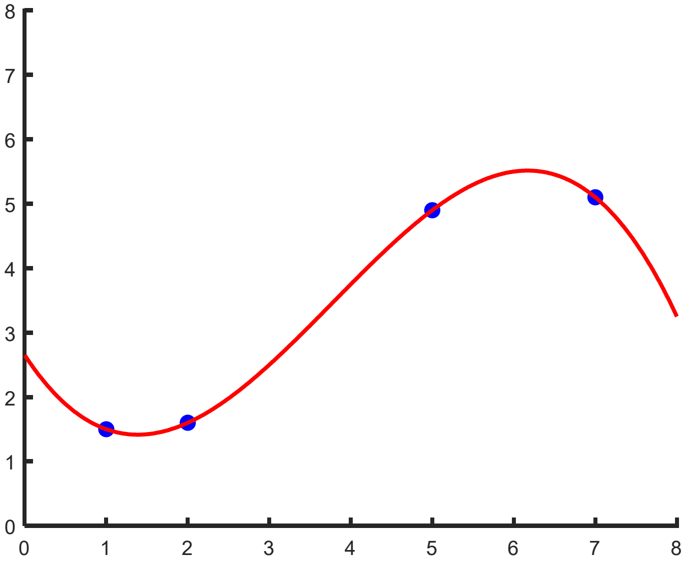
$$f(x) = -0.0805x^2 + 1.3331x - 0.1339$$

Which one is better?

Model selection problem



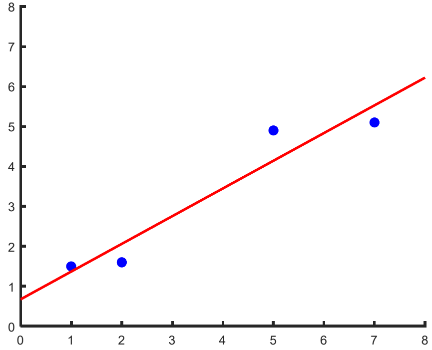
$$f(x) = 0.6934x + 0.6747$$



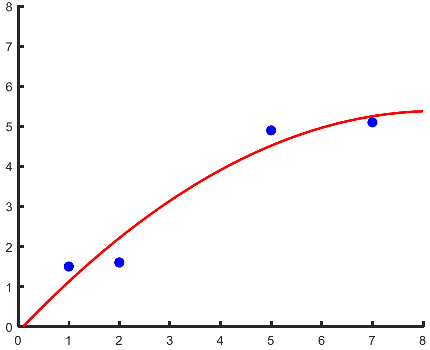
$$f(x) = -0.075x^3 + 0.85x^2 - 1.925x + 2.65$$

Which one is better?

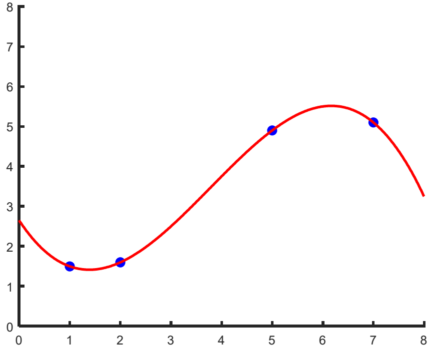
Model selection problem



$f(x) = 0.6934x + 0.6747$



$f(x) = -0.0805x^2 + 1.3331x - 0.1339$



$f(x) = -0.075x^3 + 0.85x^2 - 1.925x + 2.65$

- For a same given data set…
- Higher order polynomial has more degree of freedom
  - 2nd order polynomical can be considered a just special case of 3rd order polynomial
- Higher order polynomial has lower loss value
  - Does it mean it is better?

What is the purpose of machine learning?

- In case of supervised learning, we want our algorithm to learn a function that can represent the relation between the input and the output … and **we are expecting the function to be useful for unseen data**
- In other words, we are assuming that our data has some **underlying structure** (although it may not be apparent) and we are **trying find the structure** (via function approximation)
- Minimizing the loss function is just a mean, but not our real goal

What does this mean?

- If we know something about the underlying structure of the data, we can (and should) exploit that knowledge

- Choosing an appropriate loss function is important

- We need a measure of our algorithm's **performance for unseen data**

Bias-Variance Tradeoff

- Let's assume that our data is generated as follow

$$y = f(x) + \epsilon$$

- *f(x)* : true relation between the input and the output
- ε : noise with zero mean and variance of σ^2

- Here, we want to find $\hat{f}(x)$ that approximates the true function *f(x)* by minimizing the MSE loss

For a data set $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$

Seeks a function $f : X \to Y$

Such that a loss function $\mathcal{L} : X \times Y \to \mathcal{R}$ is minimized

Bias-Variance Tradeoff

- For which ever function $\hat{f}(x)$ we use, the expected error on an unseen sample x is as follow

$$E[(y - \hat{f}(x))^2] = (Bias[\hat{f}(x)])^2 + Var[\hat{f}(x)] + \sigma^2$$

$$Bias[\hat{f}(x)] = E[\hat{f}(x) - y]$$

: “consistent” error (consistently over-estimate or under-estimate)

$$Var[\hat{f}(x)] = E[\hat{f}(x)^2] - E[\hat{f}(x)]^2$$

: fluctuation of $\hat{f}(x)$

$$\sigma^2$$

: irreducible error (even when $\hat{f}(x) = f(x)$)

Proof can be found @ https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff

Bias-Variance Tradeoff

- Bias

$$Bias[\hat{f}(x)] = E[\hat{f}(x) - y]$$

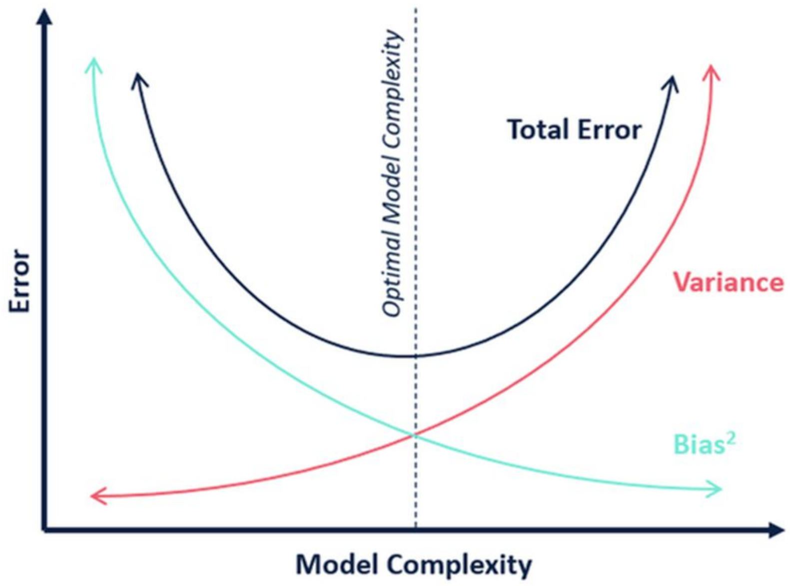
  - The bias error is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to **miss the relevant relations** between features and target outputs (underfitting).

- Variance

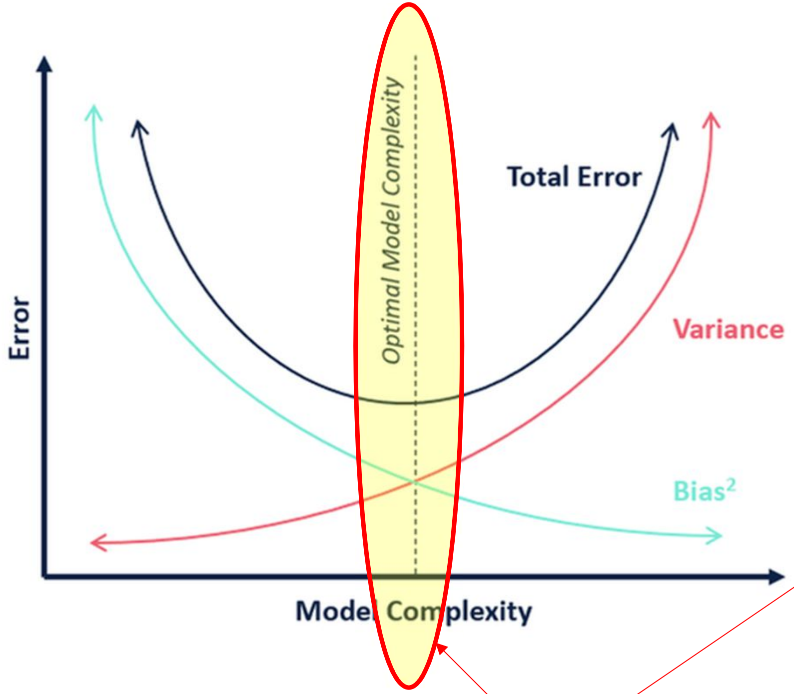
$$Var[\hat{f}(x)] = E[\hat{f}(x)^2] - E[\hat{f}(x)]^2$$

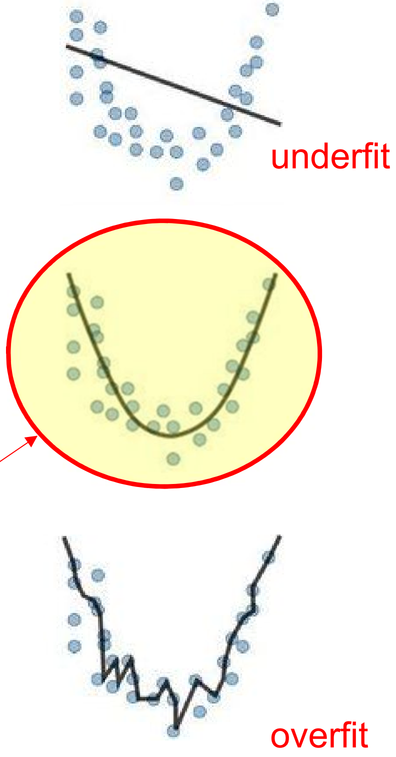
  - The variance is an error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to **model the random noise** in the training data, rather than the intended outputs (overfitting)

Bias-Variance Tradeoff



Bias-Variance Tradeoff

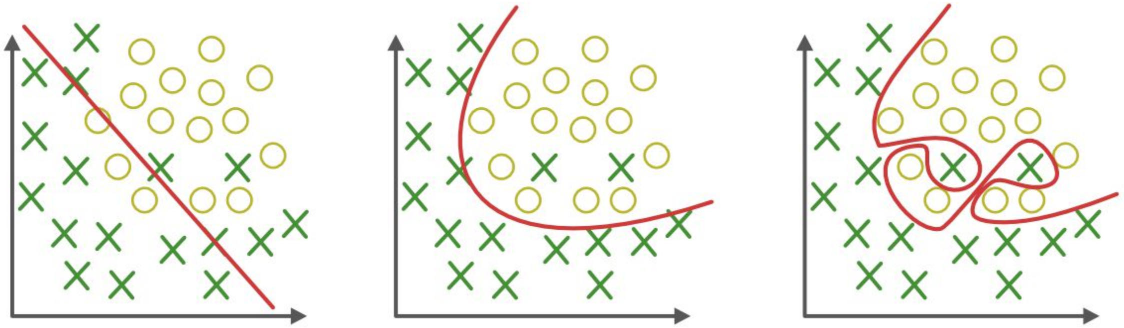




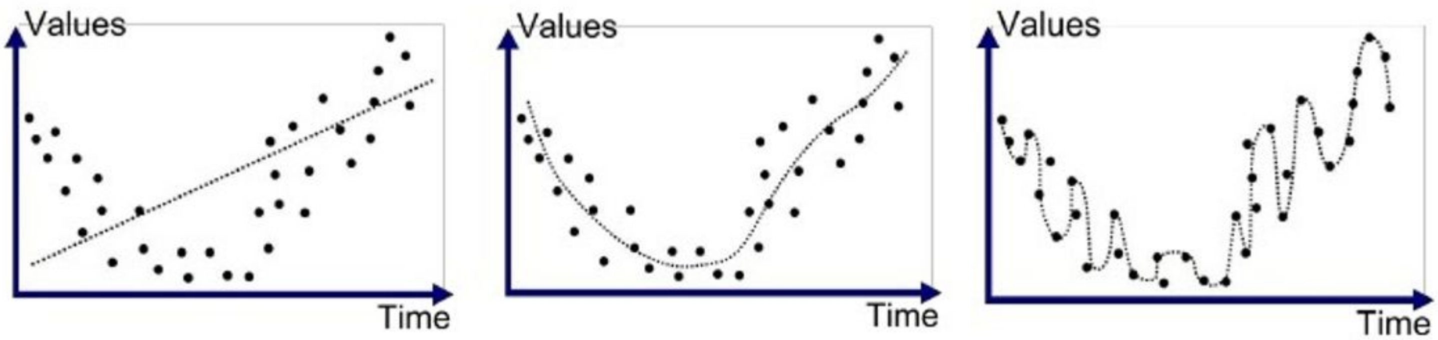
We want to be here

Overfitting

In classification



In regression

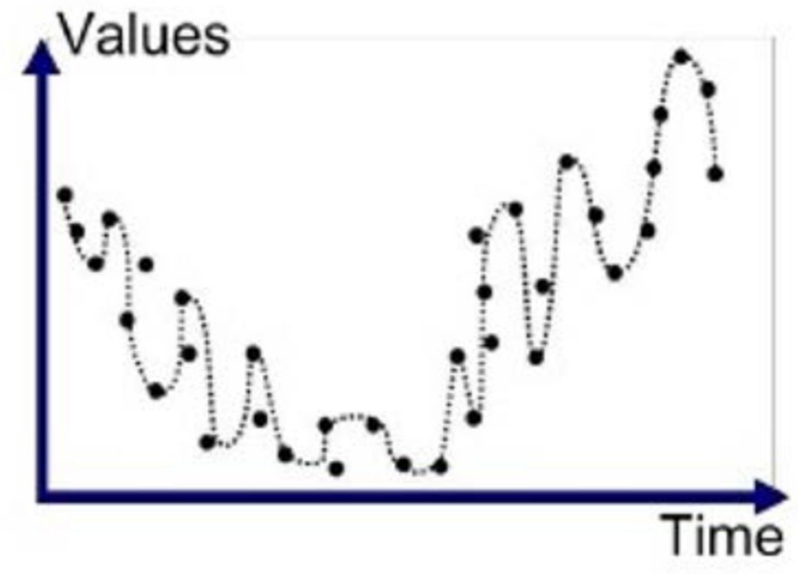


Generalization

- **Generalization** refers to a model's ability to adapt properly to new, previously unseen data, drawn from the same distribution as the one used to create the model.
- If performance on training data ~= performance on unseen data
  - then, we can speculate that the model has learned “real” underlying structure of the data set
  - in such case, we say the model generalizes well
- If performance on training data >> performance on unseen data
  - then, we can speculate that the model has learned something meaningless
  - in such case, we say the model generalizes poorly (or we say the model overfits the data)

How do we know if our model "will" generalize well?

- Q) does figure below truly shows overfitting?



For good generalization …

- From earlier slides, we learned that …
  - Bias
    - The bias error is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to **miss the relevant relations** between features and target outputs (underfitting).
  - Variance
    - The variance is an error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to **model the random noise** in the training data, rather than the intended outputs (overfitting)

- **This tells us we have to balance between bias and variance**
- **to not miss the relevant relation**
- **to not model the random noise**
- **But, how do we know if it's a relevant relation or noise?**

How do we know if our model "will" generalize well?

- **Short answer)** Test our performance on new data set!
  - What we really want to know is our model's prediction performance on unseen new data set…which can be estimated by testing its performance on unseen new data set
  - But then do we have to get truly new data sets?

  - Trick: we split our available data into multiple sets
    - We can regard some of them as "old" and some as "new"
    - We train our model with a part of the data set, and then **validate** our model with the other (exclusive) part of the data set

Validation

- We need a method (or measure) to check how well our model generalizes

- An unbiased evaluation of a **model fit** on the dataset while tuning model hyperparameters
  - The evaluation becomes biased as we tune hyperparameters for better validation

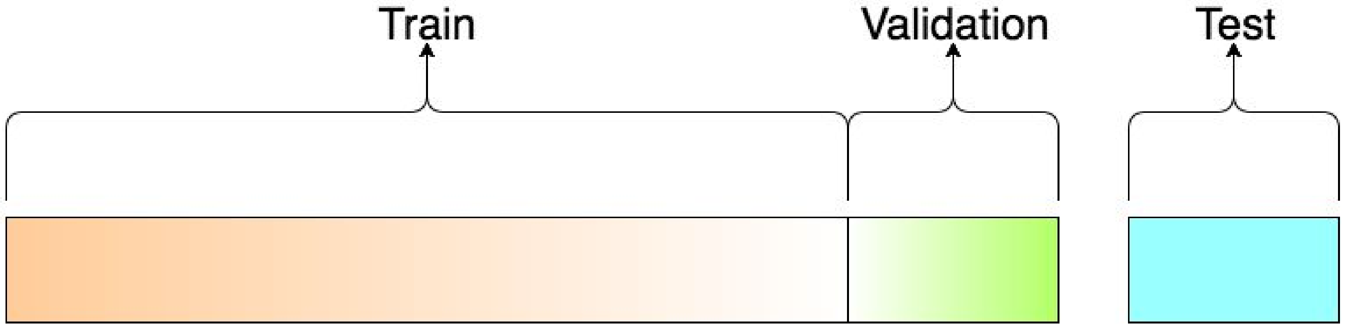
Training, Validation, Test

- How can we check the **performance for unseen data**?
- We (a) split our data, then (b) use only part of it for training and (c) see its performance on the other part of the data

- **How do we split?**
  - **Training data set**: data set that is directly used for parameters updates (i.e., for gradient calculation)
  - **Validation data set**: data set that is indirectly used for training (i.e., for tuning *hyperparameters)
  - **Test data set**: data set that is used for final evaluation of the performance. Should never be used anyhow for training

*hyperparameters: parameters whose values are used to control the learning process
(e.g., learning rate, mini-batch size, number of iterations, model complexity, etc)

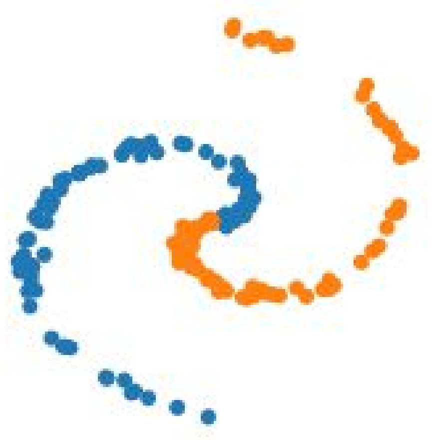
Training, Validation, Test



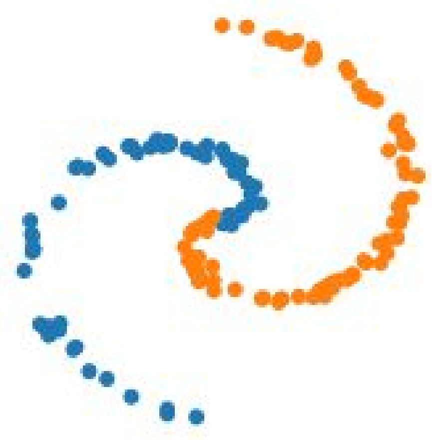
- Depending on the availability and characteristics of the data set, the data can be split differently
  - 80-10-10
  - 50-25-25
  - Other ratios …

Training, Validation, Test

Training data set



Validation data set



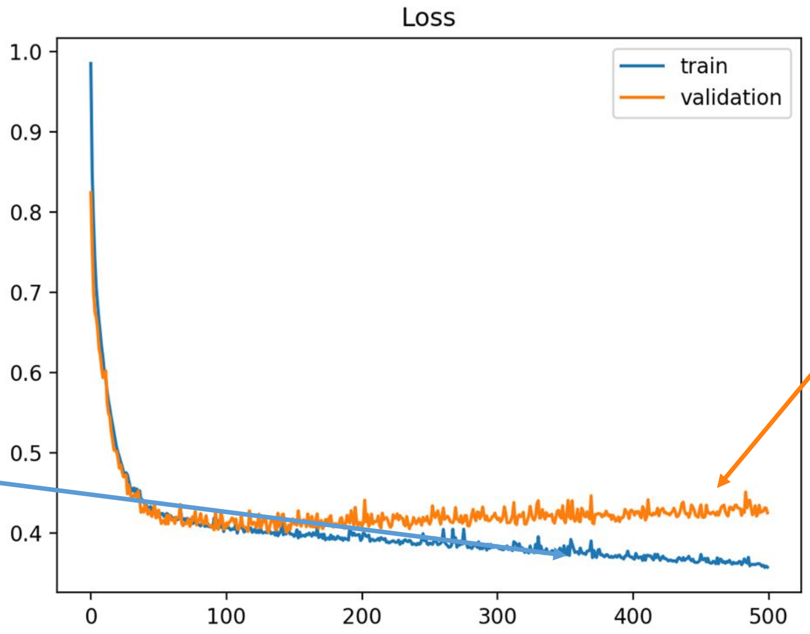
Test data set

- (Randomly) split the entire data set into training, validation and test data sets
- Use training set (for parameter updates) and validation set (for hyperparameter tuning) for training
- Use test set for final performance evaluation

Training & Validation

- Typical learning curve

Our model is getting better and better over time for the training data set…



But from validation, we can see that overfitting is occurring (performance for unseen data is not getting better)

Validation vs. Test

- **How is a test different from validation?**
  - From the earlier slide **…**“The evaluation becomes biased as we tune hyperparameters for better validation”

  - Although validation data set was not directly used for parameter updates, we (human) reflected the results from validation set to tune the hyperparameters
  - This means validation performance does not 100% truly reflect the performance for unseen data set (as our model was optimized for the validation performance)
  - **Test data set** is different from the validation data set in the sense that it <u>never anyhow affected the training procedure</u>

Summary

- Model selection problem
- Bias-variance tradeoff
  - If we decrease one, the other will increase
- Underfitting & Overfitting
  - Model capacity, model complexity
- Generalization & Validation
- Splitting dataset
  - Training, validation, test

References

- Lecture notes
  - CC229 lecture note
    - http://cs229.stanford.edu/notes2020fall/notes2020fall/cs229-notes5.pdf

- Website
  - CS231n course website: https://cs231n.github.io/