Machine Learning

CSE 142

Xin (Eric) Wang

Friday, October 8, 2021

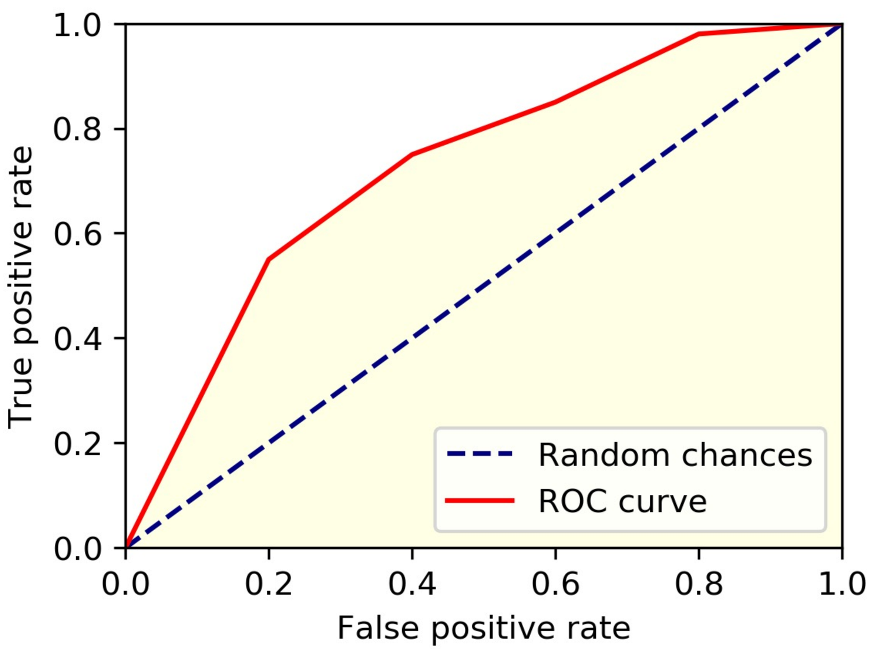
Today

- Classification, Ch. 2 & 3

Classifier design – operating point

Review

- You, as a classifier designer, can often move decision boundaries (modify thresholds) to make the false positive rate as high or as low as you wish
  – A very high threshold (don't let anything through!) results in no false positives – but lots of false negatives
  – A very low threshold (let everything through!) results in no false negatives – but lots of false positives

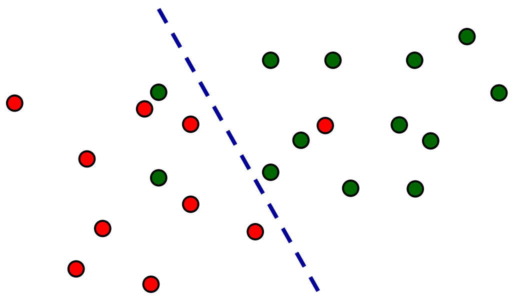


- This doesn't necessarily make the classifier better or worse – it just changes the operating point of the classifier

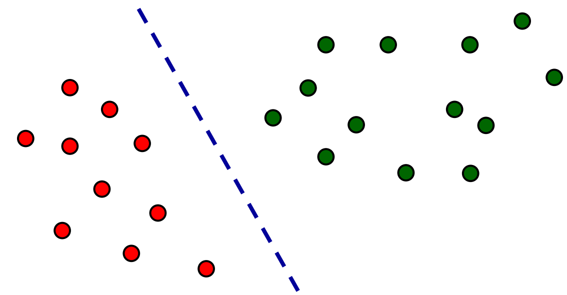
- This is often application-specific:
  - When might false positives be especially undesirable?
  - When might false negatives be especially undesirable?
  - We can encode these preferences in a cost function to compute an optimal threshold, given this information

Classifier design

A or B?

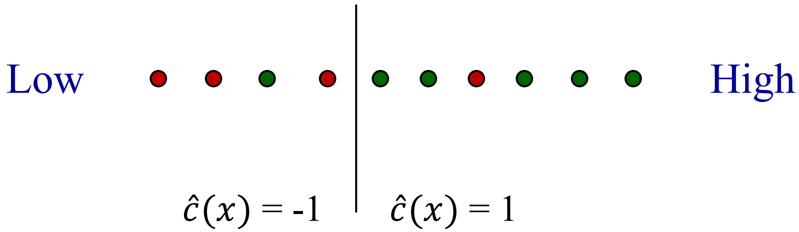


A

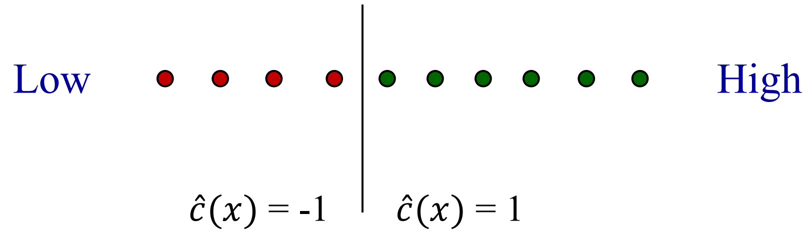


B

A or B?



A



B

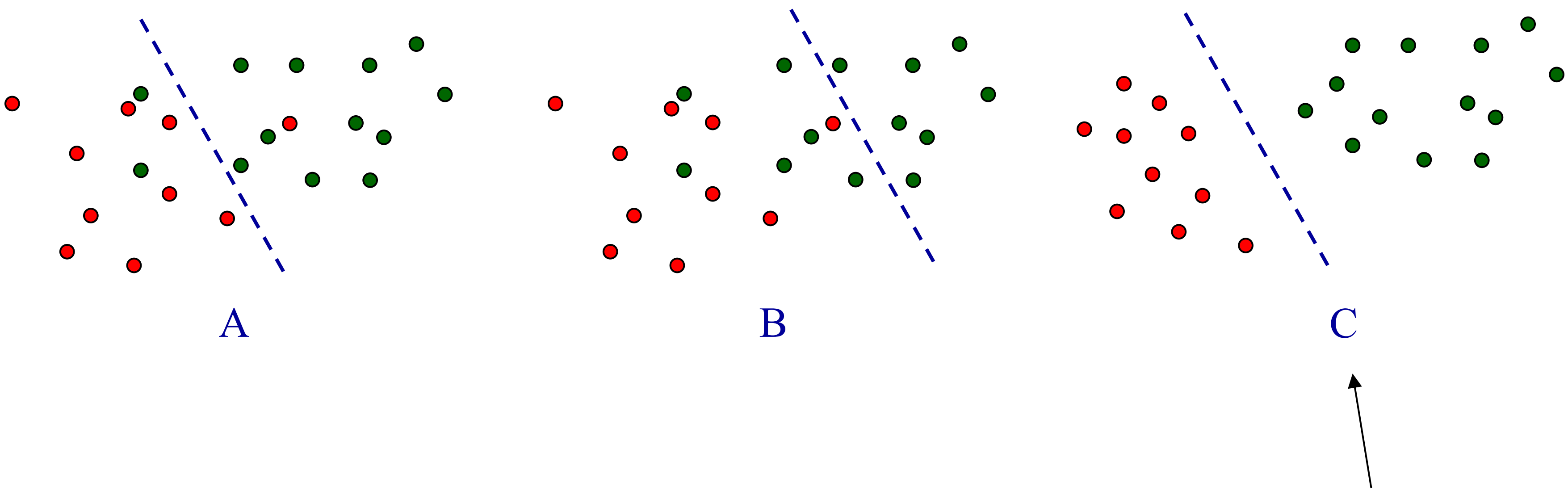
It also depends on how good your features are!

- Either *raw* features or *constructed* features

Feature separation vs. classifier design

Placing the separating boundary = classifier design

Increasing the feature separation = feature construction



We can design a better classifier starting with the features in C!

Typical predictive machine learning scenarios

Task	*Label space*	*Output space*	*Learning problem*
Classification	$\mathscr{L} = \mathscr{C}$	$\mathscr{Y} = \mathscr{C}$	learn an approximation $\hat{c} : \mathscr{X} \to \mathscr{C}$ to the true labelling function c
Scoring and ranking	$\mathscr{L} = \mathscr{C}$	$\mathscr{Y} = \mathbb{R}^{\lvert\mathscr{C}\rvert}$	learn a model that outputs a score vector over classes
Probability estimation	$\mathscr{L} = \mathscr{C}$	$\mathscr{Y} = [0,1]^{\lvert\mathscr{C}\rvert}$	learn a model that outputs a probability vector over classes
Regression	$\mathscr{L} = \mathbb{R}$	$\mathscr{Y} = \mathbb{R}$	learn an approximation $\hat{f} : \mathscr{X} \to \mathbb{R}$ to the true labelling function f

Class probability estimation

A class probability estimator is a scoring classifier that outputs probabilities over the k classes – i.e., a mapping:

$$\hat{p} \; : \; \mathcal{X} \; \rightarrow \; [0,1]^k$$

where

$$\sum_{i=1}^{k} \hat{p}_i \,(x) = 1$$

A key issue here is that we generally do not have access to the true probabilities for training data.

- E.g., an email is either spam or ham – it doesn't have a probability of being spam!
- So how can we train to learn such probabilities?

Empirical probabilities

- In machine learning, we often calculate *empirical probabilities* – i.e., calculate relative frequencies from the available data

N_i instances of the class C_i in the training data S:

$$\text{Relative frequency} = \frac{N_i}{|S|} = \hat{p}_i$$

- But this can be problematic, especially with small amounts of training data
  - Probabilities of 0 and 1 generally should be avoided
- There are various common ways to smooth or correct the relative frequencies to avoid 0 and 1
  - E.g., Laplace correction and m-estimate:

Add a pseudo-count to each class

$$\text{Laplace correction} = \frac{N_i + 1}{|S| + k}$$

Choose number of pseudo-counts m and their class distribution π_i

$$\text{m-estimate} = \frac{N_i + m\pi_i}{|S| + m} \qquad \textstyle\sum_i \pi_i = 1$$

Think of π_i as the prior probabilities, and m the amount to weigh the priors relative to $|S|$

Quiz: empirical probabilities

Training data set S

C_1: 7 instances

C_2: 14

C_3: 0

C_4: 4

Relative frequency $= \frac{N_i}{|S|} = \hat{p}_i$

$\hat{p}_1$:

$\hat{p}_2$:

$\hat{p}_3$:

$\hat{p}_4$:

Laplace correction $= \frac{N_i+1}{|S|+k}$

m-estimate $= \frac{N_i+m\pi_i}{|S|+m}$ $\quad \sum_i \pi_i = 1$

$m = 40 : \{10, 10, 10, 10\}$ $\qquad m = 20 : \{5, 0, 9, 6\}$

Quiz: empirical probabilities

Training data set S

C_1: 7 instances

C_2: 14

C_3: 0

C_4: 4

Relative frequency = $\frac{N_i}{|S|} = \hat{p}_i$

$|S| = 25$

$\hat{p}_1$: 7/25 = 0.28
$\hat{p}_2$: 14/25 = 0.56
$\hat{p}_3$: 0/25 = 0.0
$\hat{p}_4$: 4/25 = 0.16

Laplace correction = $\frac{N_i+1}{|S|+k}$

$\hat{p}_1$: 8/29 = 0.28
$\hat{p}_2$: 15/29 = 0.52
$\hat{p}_3$: 1/29 = 0.03
$\hat{p}_4$: 5/29 = 0.17

m-estimate = $\frac{N_i+m\pi_i}{|S|+m}$ $\quad \Sigma_i \pi_i = 1$

π_i = {0.25, 0.25, 0.25, 0.25}
m = 40 : {10, 10, 10, 10}

$\hat{p}_1$: 17/65 = 0.26
$\hat{p}_2$: 24/65 = 0.37
$\hat{p}_3$: 10/65 = 0.15
$\hat{p}_4$: 14/65 = 0.22

π_i = {0.25, 0, 0.45, 0.3}
m = 20 : {5, 0, 9, 6}

$\hat{p}_1$: 12/45 = 0.27
$\hat{p}_2$: 14/45 = 0.31
$\hat{p}_3$: 9/45 = 0.2
$\hat{p}_4$: 10/45 = 0.22

Multi-class classification – beyond binary!

- Many classification problems involve multiple classes
- Performance can be described with the multi-class contingency table
  - Also known as the **confusion matrix**
  - We can compute accuracy, per-class precision, per-class recall…

	Predicted			
	15	2	3	20
Actual	7	15	8	30
	2	3	45	50
	24	20	56	100

Accuracy = (15+15+45)/100 = 0.75

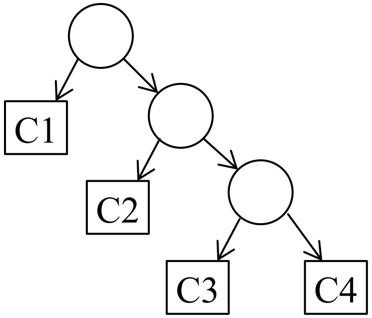
Class 1 precision = 15/24 = 0.63

Class 1 recall = 15/20 = 0.75

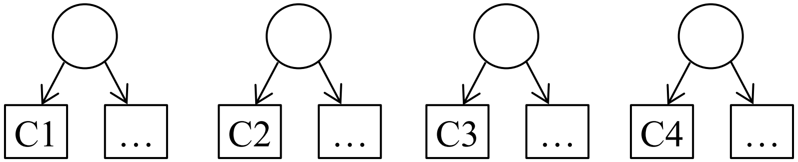
Etc.

K-class classifiers

- How to build a *k*-class classifier?
  - We can combine several binary classifiers, e.g.:
    - One-versus-rest scheme #1 – learn *k*-1 models, apply in sequence
      - C1 vs. { C2, C3, C4 }
      - C2 vs. { C3, C4 }
      - C3 vs. C4

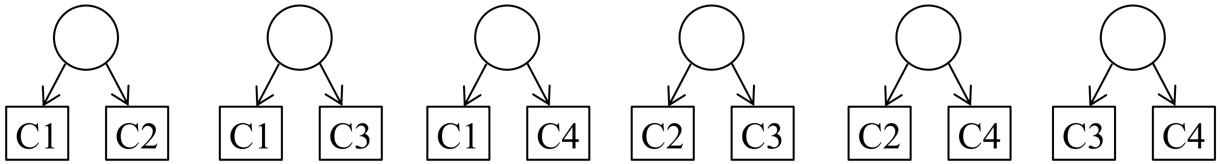


    - One-versus-rest scheme #2 – learn a *one-class* model for each class
      - C1 vs. { C2, C3, C4 }
      - C2 vs. { C1, C3, C4 }
      - C3 vs. { C1, C2, C4 }
      - C4 vs. { C1, C2, C3 }

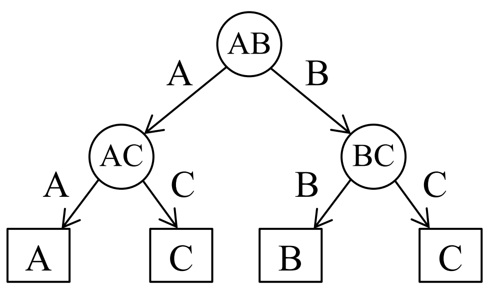


K-class classifiers

- One-versus-one scheme #1 – learn a model for each pair of classes
  - Train $k(k\text{-}1)/2$ binary classifiers, apply them all to x and **vote**

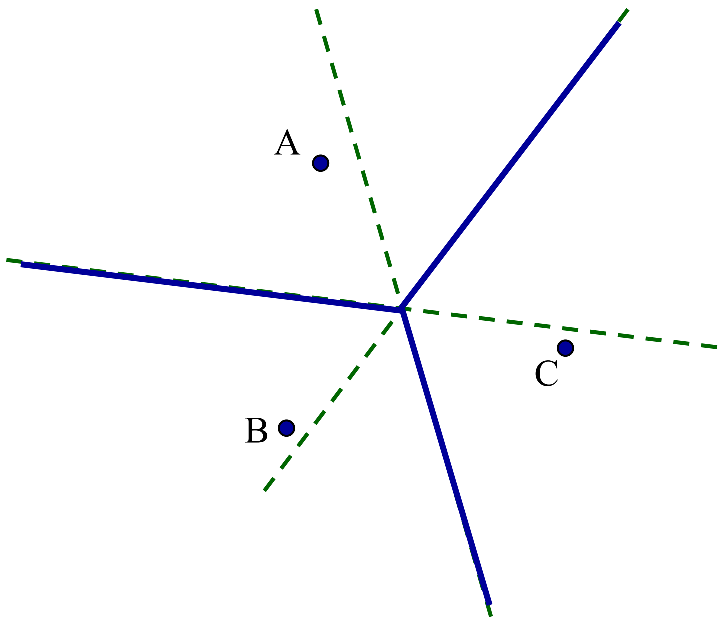


- One-versus-one scheme #2 with a decision tree:



Example: A 3-class linear classifier

Classify instances into three classes {A, B, C} using three linear discriminant functions classifying (A vs. B), (A vs. C), and (B vs. C)



This implements the one-versus-one scheme #2 on the previous slide

Typical predictive machine learning scenarios

Task	*Label space*	*Output space*	*Learning problem*
Classification	$\mathscr{L} = \mathscr{C}$	$\mathscr{Y} = \mathscr{C}$	learn an approximation $\hat{c} : \mathscr{X} \rightarrow \mathscr{C}$ to the true labelling function c
Scoring and ranking	$\mathscr{L} = \mathscr{C}$	$\mathscr{Y} = \mathbb{R}^{\lvert\mathscr{C}\rvert}$	learn a model that outputs a score vector over classes
Probability estimation	$\mathscr{L} = \mathscr{C}$	$\mathscr{Y} = [0,1]^{\lvert\mathscr{C}\rvert}$	learn a model that outputs a probability vector over classes
Regression	$\mathscr{L} = \mathbb{R}$	$\mathscr{Y} = \mathbb{R}$	learn an approximation $\hat{f} : \mathscr{X} \rightarrow \mathbb{R}$ to the true labelling function f

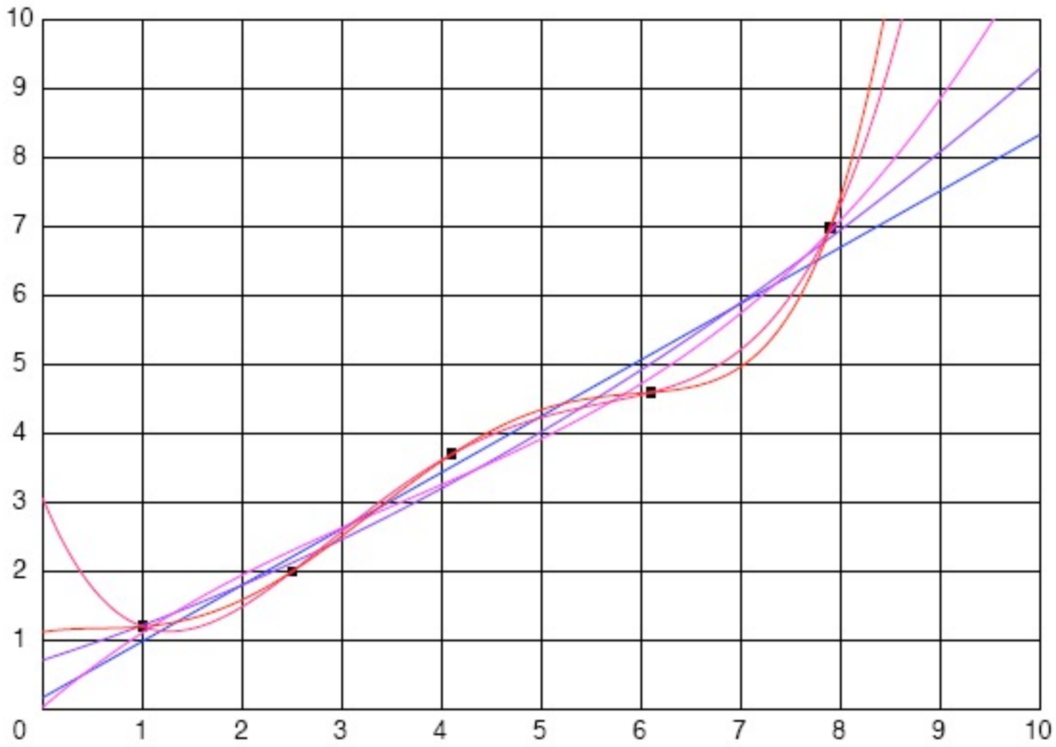
Regression – another predictive ML task

- In the classification tasks we've been discussing, the label space was a discrete set of classes
  - Classification, scoring, ranking, probability estimation
- Regression learns a function (the regressor) that is a mapping $\hat{f} : \mathcal{X} \rightarrow \mathbb{R}$ from examples – $f(x_i)$
  - I.e., the target variable (output) is real-valued
- Assumption: the examples will be noisy, so watch out for overfitting – we want to capture the general trend or shape of the function, not exactly match every data point
  - E.g., if fitting an N-degree polynomial to the training data (thus N+1 parameters to estimate), choose as small N as possible
- The number of data points should be much greater than the number of parameters to be estimated!
  - How much data is needed? This is an open question in ML….

Regression example

Training data

x	$f(x)$
1.0	1.2
2.5	2.0
4.1	3.7
6.1	4.6
7.9	7.0



{1, 2, 3, 4, 5}-degree polynomial functions

The regression function may or may not fit the training data exactly

Regression

- We'll generally estimate a regression function based on some function of the *residual*, the different between the estimate and the label (the true value):

$$r(x) = \underbrace{f(x)}_{\text{True function}} - \underbrace{\hat{f}(x)}_{\text{Regression function}}$$

- That function is (again) the *loss function* L
  - The most common loss function for regression is the squared residual:

$$L(r) = \mathbb{E}\left[\, r^2(x))\right] = \mathbb{E}\left[\left(f(x) - \hat{f}(x)\right)^2\right]$$

Bias-Variance Tradeoff

$$L(r) = \mathbb{E}\left[\left(f(x) - \hat{f}(x)\right)^2\right]$$

$$= \mathbb{E}\left[\left(f(x) - \mathbb{E}[\hat{f}(x)] + \mathbb{E}[\hat{f}(x)] - \hat{f}(x)\right)^2\right]$$

$(a+b)^2 = a^2 + b^2 + 2ab$

$$= \mathbb{E}\left[\left(f(x) - \mathbb{E}[\hat{f}(x)]\right)^2\right] + \mathbb{E}\left[\left(\mathbb{E}[\hat{f}(x)] - \hat{f}(x)\right)^2\right]$$
$$+ \mathbb{E}\left[2\left(f(x) - \mathbb{E}[\hat{f}(x)]\right)\left(\mathbb{E}[\hat{f}(x)] - \hat{f}(x)\right)\right]$$

$f(x)$ and $\mathbb{E}\left[\hat{f}(x)\right]$ constant with respect to the noise

$$= \left(f(x) - \mathbb{E}[\hat{f}(x)]\right)^2 + \mathbb{E}\left[\left(\mathbb{E}[\hat{f}(x)] - \hat{f}(x)\right)^2\right]$$

$= \text{Bias}^2 + \text{Variance}$

difference between the average prediction of our model and the true value

variations of our training data

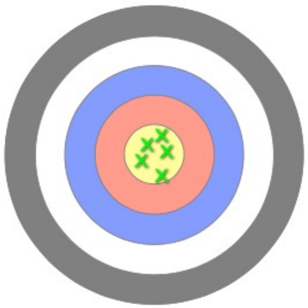
Bias-Variance Tradeoff

$$L(r) = \left(f(x) - \mathbb{E}[\hat{f}(x)]\right)^2 + \mathbb{E}\left[\left(\mathbb{E}[\hat{f}(x)] - \hat{f}(x)\right)^2\right]$$

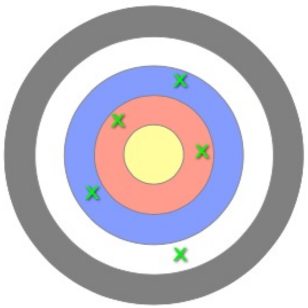
$$= \text{Bias}^2 + \text{Variance}$$

difference between the average prediction of our model and the true value

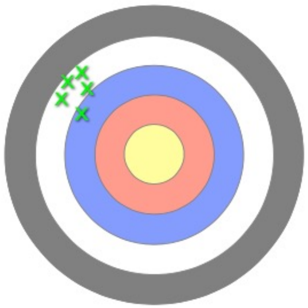
variations of our training data



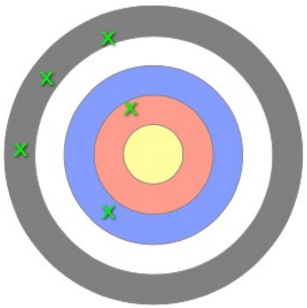
Low bias, low variance



Low bias, high variance



High bias, low variance

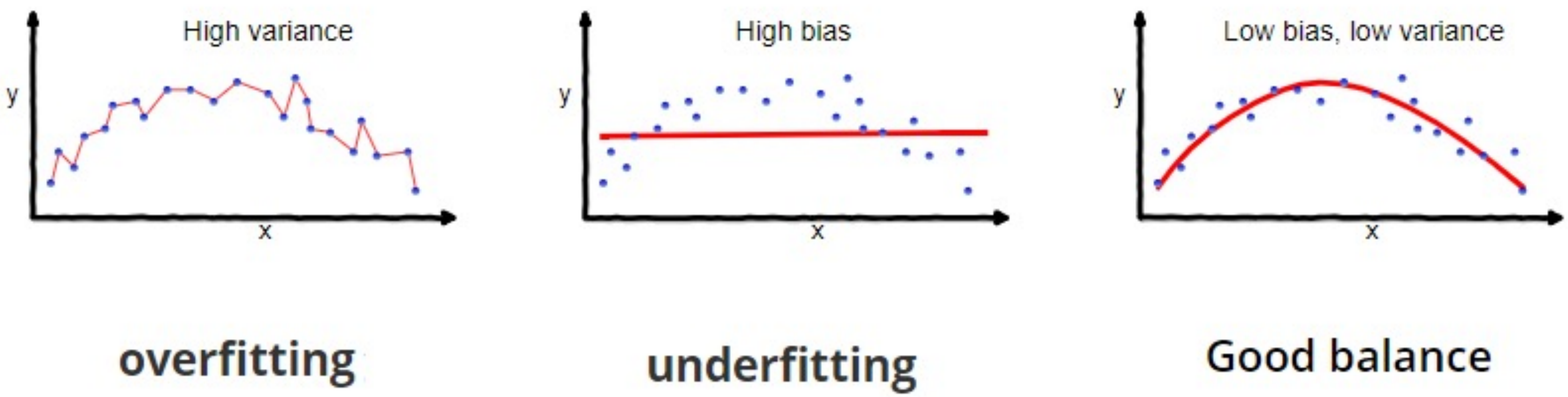


High bias, high variance

Bias-Variance Tradeoff

- Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before – **Overfitting**
    - Such models perform very well on training data but poorly on test data

- Model with high bias pays very little attention to the training data and oversimplifies the model – **Underfitting**
    - High error on training and test data



NEXT

Concept Learning

READ

Chapter 4 in the textbook

Logical Models: tree models and rule models