Medical imaging project - Classifiers

Dr. Veronika Cheplygina

Medical imaging project - timeline

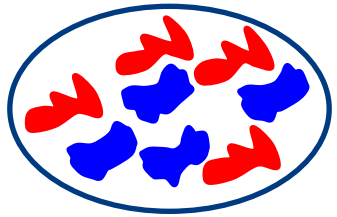
- Tue 6 – Introduction
- Fri 9 – Extract features from images
- Tue 13 – **Classify image to predict diagnosis**
- Fri 16 – Evaluate results
- Tue 20 – TBA / Recent research

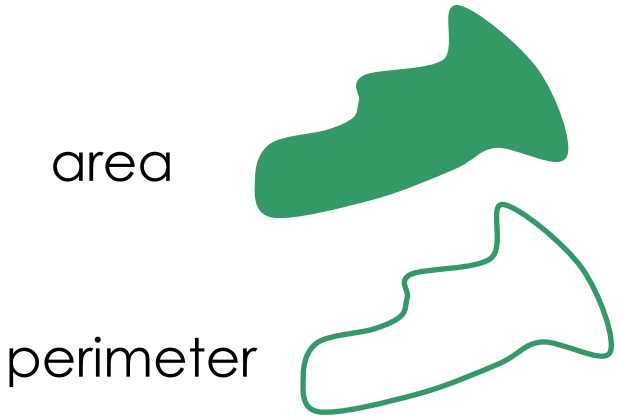
Today

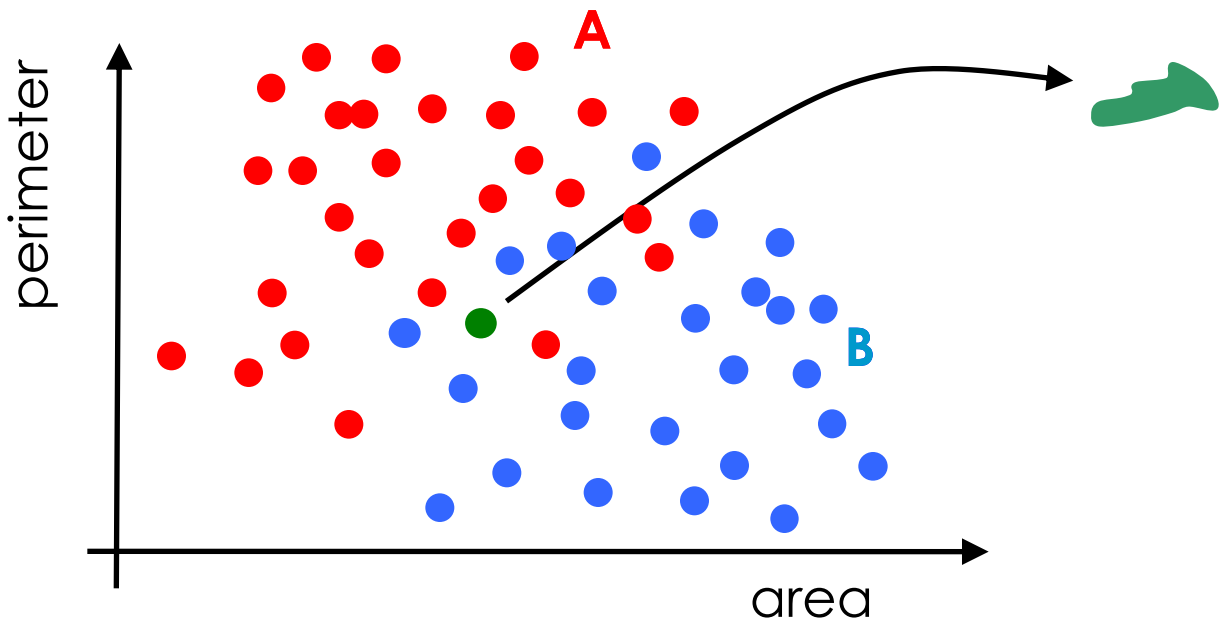
- Nearest mean & nearest neighbor classifiers
  - https://www.youtube.com/watch?v=EK55c9r-n8w
- Classifier properties
- Overfitting

Nearest mean & nearest neighbor

- Use the features of existing data to train a **classifier**

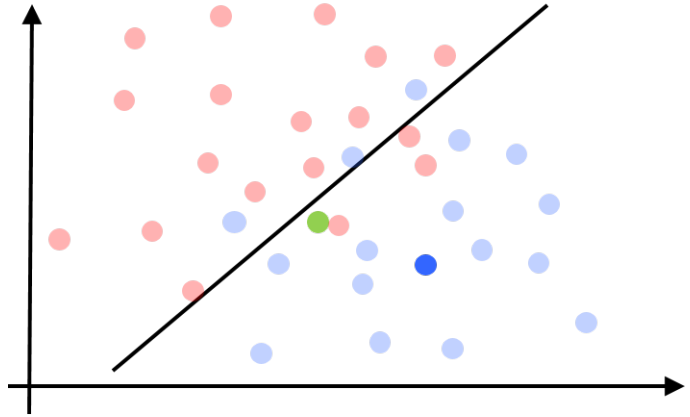






Example classifier – Nearest mean classifier

- Represent each class by its average point
- Measure distances to “average red” and “average blue”



Example with 5 training points using 3 features

Class 1:

[3 4 4]

[5 5 2]

[1 6 3]

Test point:

[5 3 4]

Class 2:

[2 2 4]

[2 4 6]

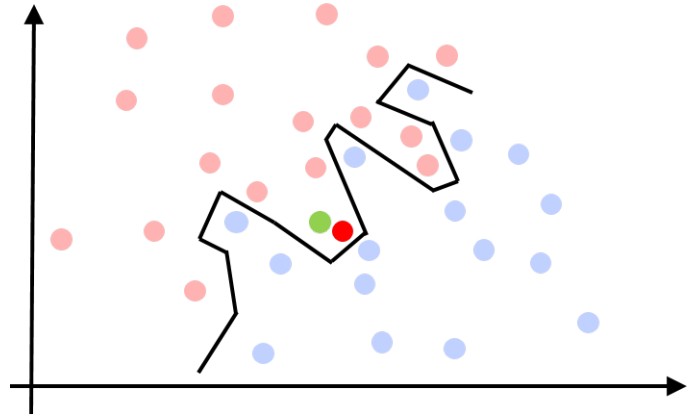
Nearest mean classifier - Steps

- Model each class **by its mean**
  - [3 5 3] for class 1, [2 3 5] for class 2

- For the test point, calculate its distances to the class means
  - $\sqrt{9}$ to class 1, $\sqrt{10}$ to class 2

- Find class with the smallest distance
  - Answer is **class 1**

- **Nearest neighbor classifier**
  - Model each class by all of its training points ("lazy learning")
  - For a test point, calculate its distances to all training point, find class with the smallest distance



- **Example nearest neighbor**

Class 1:

[3 4 4]

[5 5 2]

[1 6 3]

Class 2:

[2 2 4]

[2 4 6]

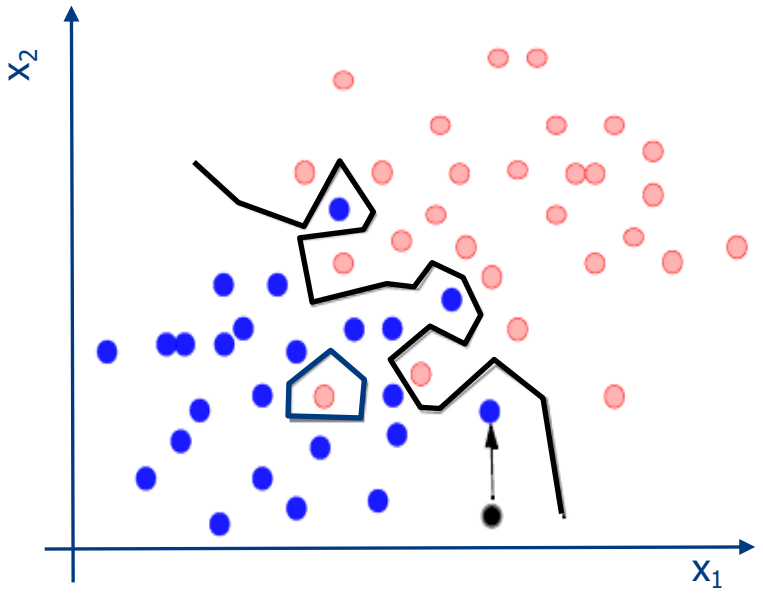
Test point:

[5 3 4]

Class 1

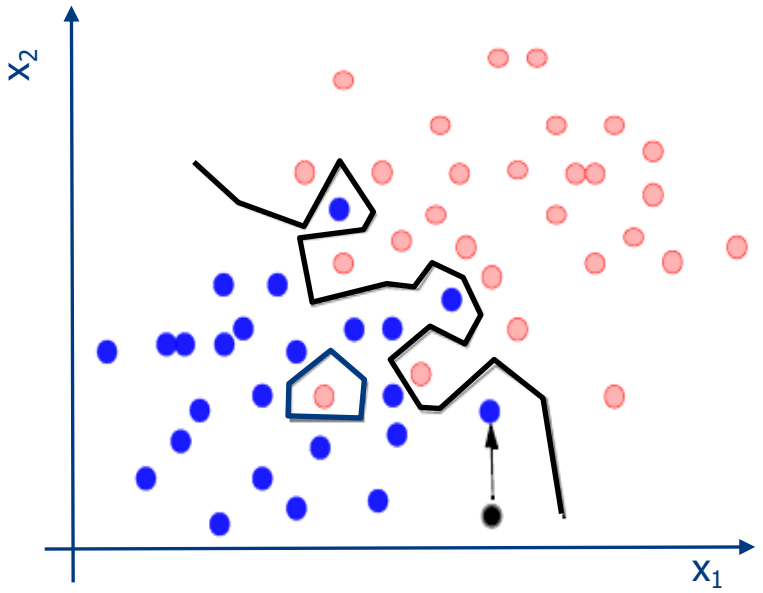
Classifier accuracy / error

- Accuracy = fraction of points that are correctly classified
- Error = 1 – accuracy
- [More metrics in next lecture]



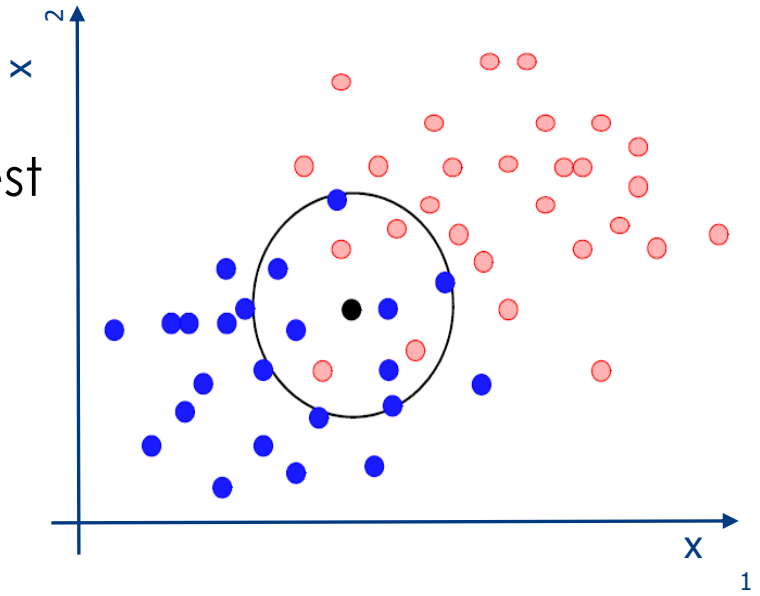
Nearest neighbor classifier

- Local changes in training data have large influence on the boundary
- If test data = training data, error = 0

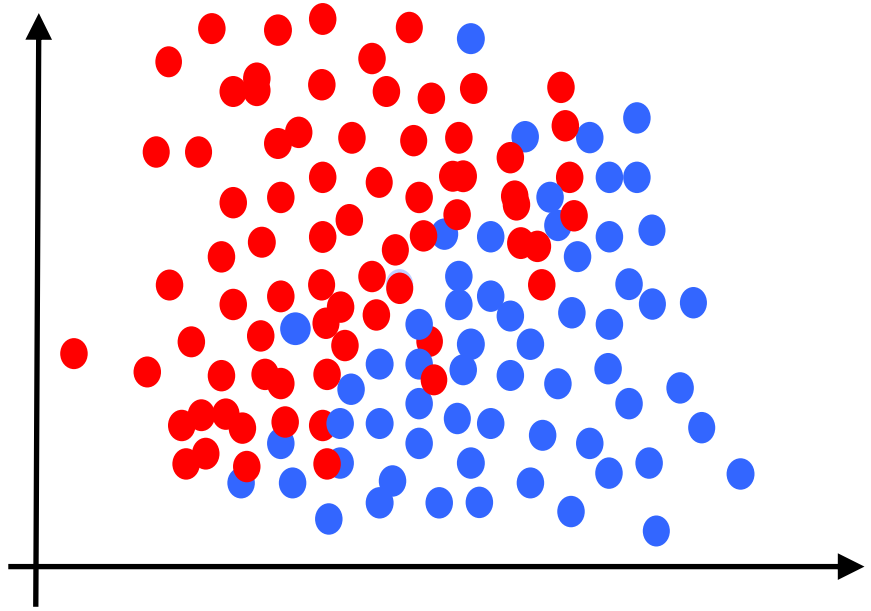


Nearest neighbor classifier

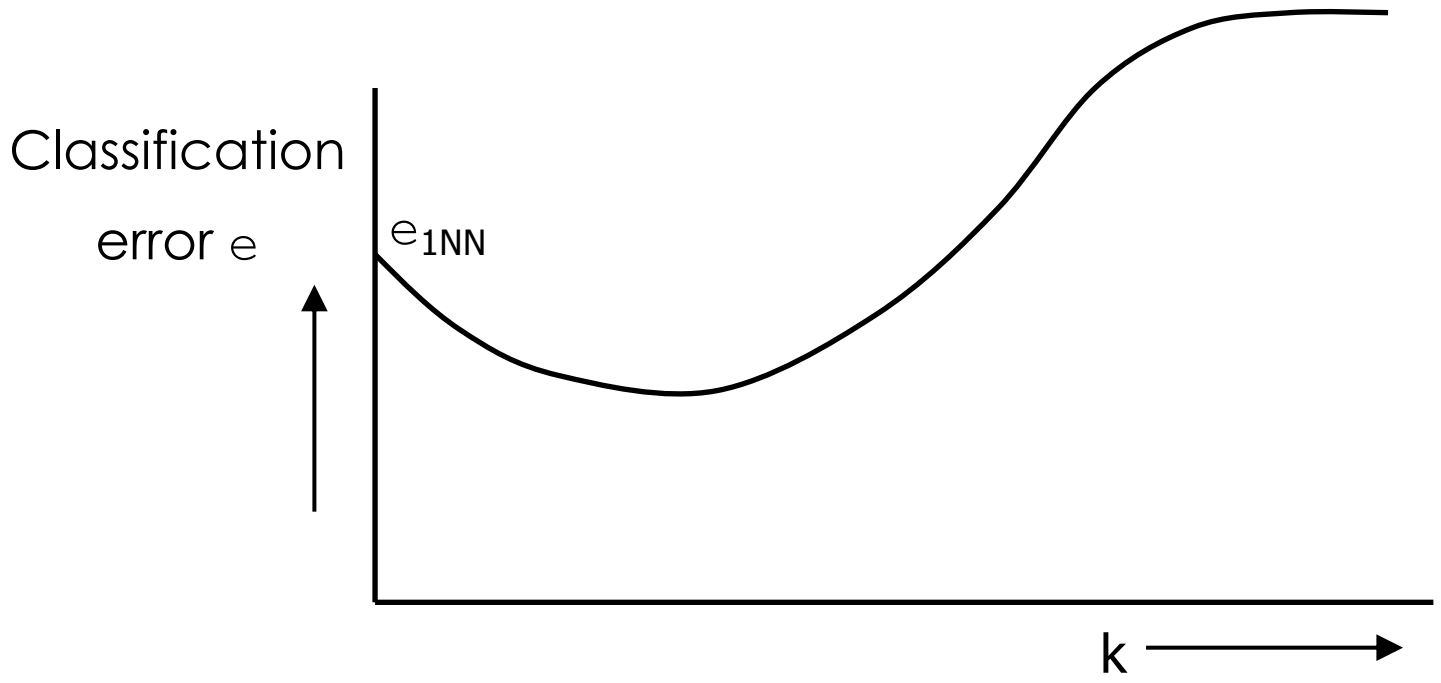
- To reduce variability, look at majority of k nearest neighbors
- Use odd k to avoid ties
- Less sensitive to local changes / boundary is more smooth
- Q: what happens if test data = training data?



- Q: What happens if k is equal to number of training points?

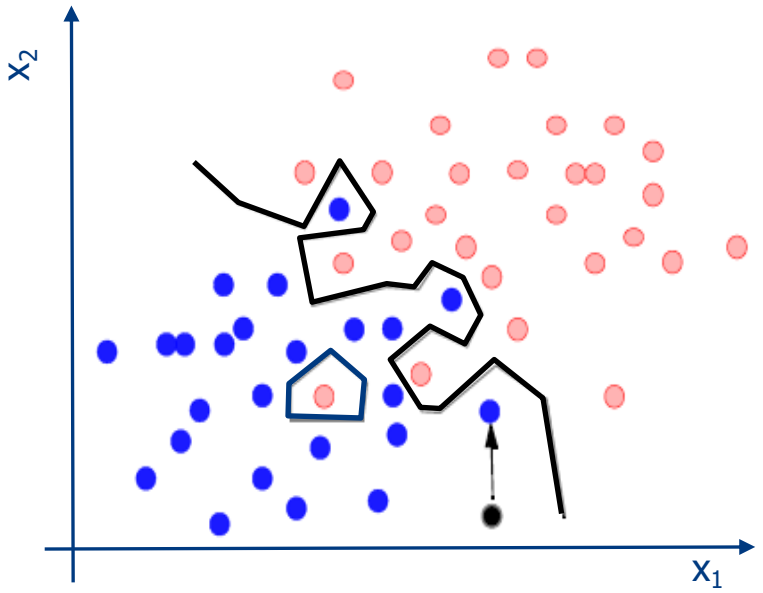


- In theory error will first decrease with k, then increase again
- This is not always the case with real-life data



Nearest neighbor classifier

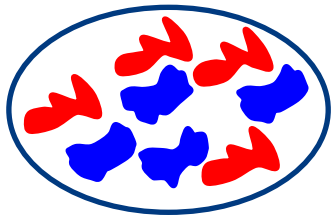
- **Not** scale invariant
  - E.g. features "weight" and "height"
  - If weight in kg, height in m, height has almost no influence on distance

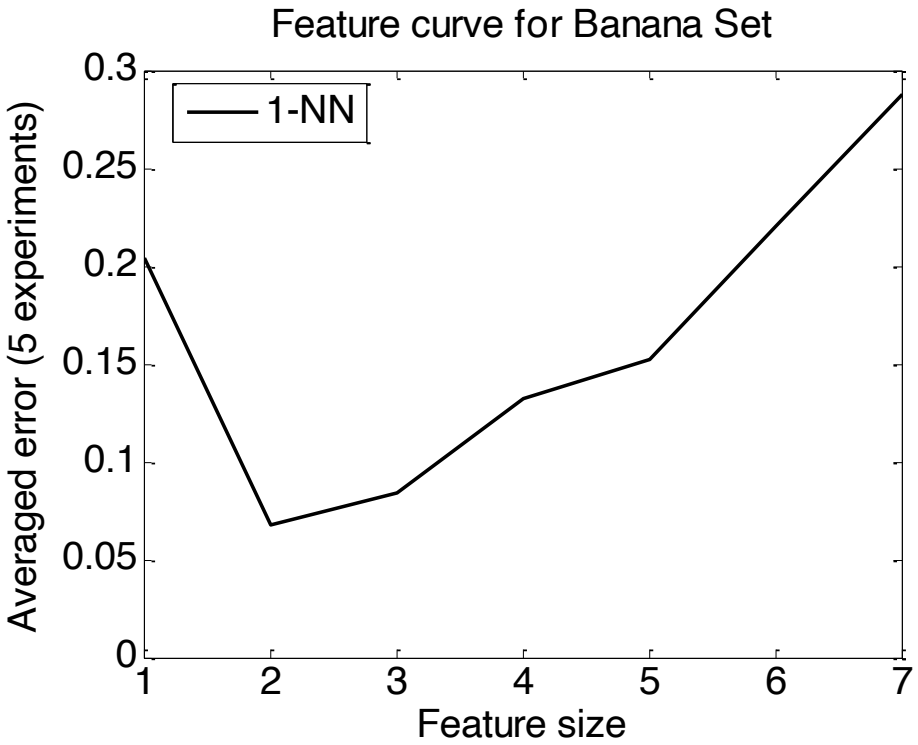


kNN sensitive to irrelevant features

Take dataset with 2 informative features, add random noise features

Error of kNN increases





Other (not deep learning) classifiers

Linear discriminant
Logistic
Decision tree
Random forest
Support vector machine
…

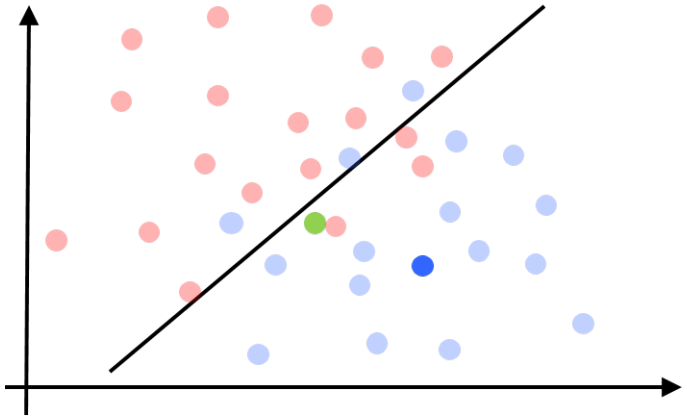
Classifier properties

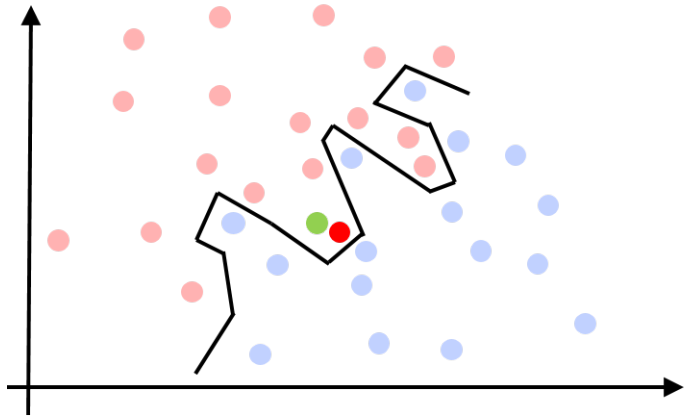
"Classifier" can refer to:

- The general type of function you could fit on your data
  - Also "untrained classifier"

- A specific classifier already trained on some data
  - Also "trained classifier"

- Any trained classifier can be visualized in 2D or 3D
  - For each (hypothetical) point, find class it belongs to
  - Draw boundary





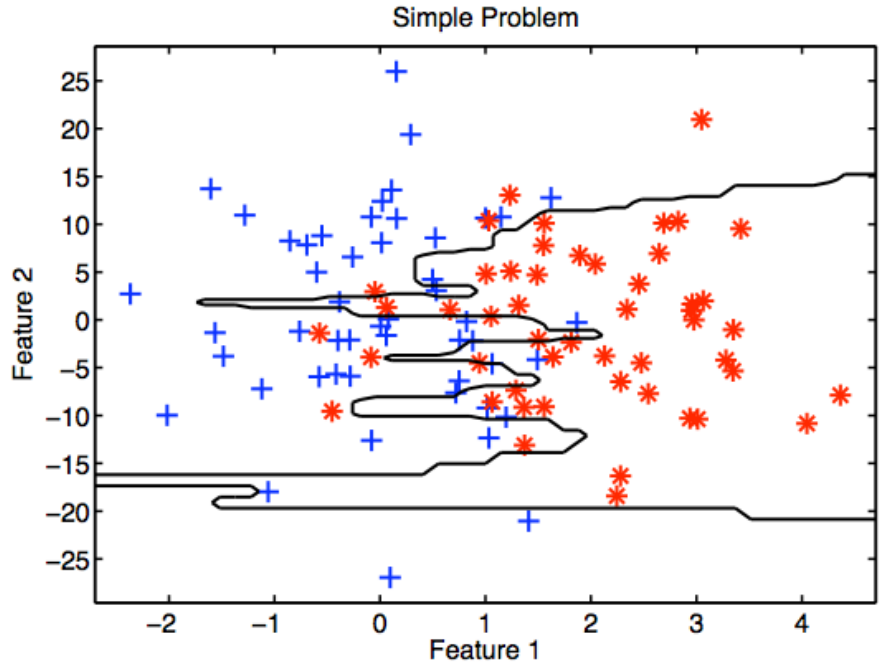
Classifier properties

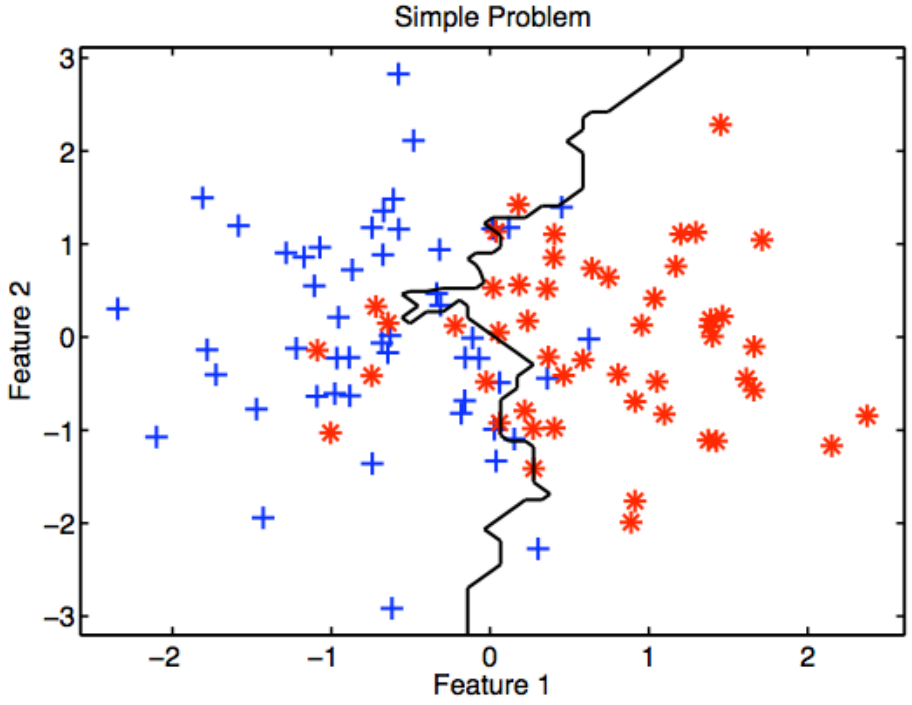
- Classifiers can be linear, or non-linear
- Some classifiers take all features into account, others thers decide the importance of features during training
- Some classifiers have hyperparameters
- Classifier flexibility/complexity

Feature scaling

- Different features might have different scales, for example
  - Color between 1 and 5
  - Area between 1000 and 5000

- Features with larger scales will have more influence on classifier



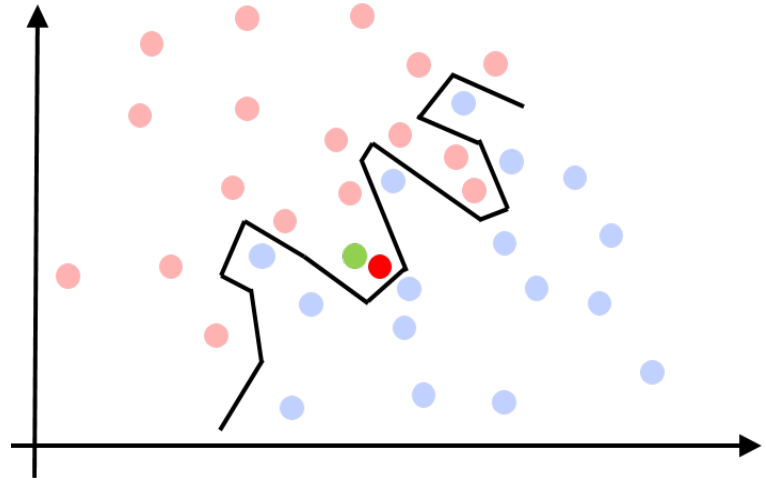


Feature scaling

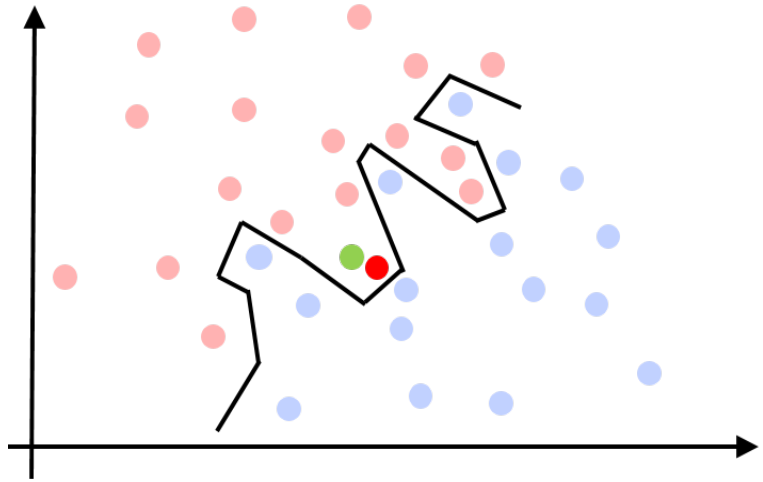
- Normalize features to ensure each feature is equally important
- For example, subtract mean, and divide by standard deviation

How to choose a classifier?

- We want to have good separation of the categories (low class overlap)
- It is always possible to design a classifier that perfectly separates your training data
- But, that classifier might **not generalize** to future data, i.e. it **overfits**



How to choose a classifier?



- Use different subsets of data for
  - Training
  - Validation (practice test set)

- Goal: similar performances training & validation sets
  - Bad: 100% training, 50% validation
  - Better: 75% training, 75% validation

- Use these sets to diagnose overfitting (decide on classifier, feature selection etc)

How to choose a classifier?

- When training & validation performances are similar, we expect similar performances on “true” test data

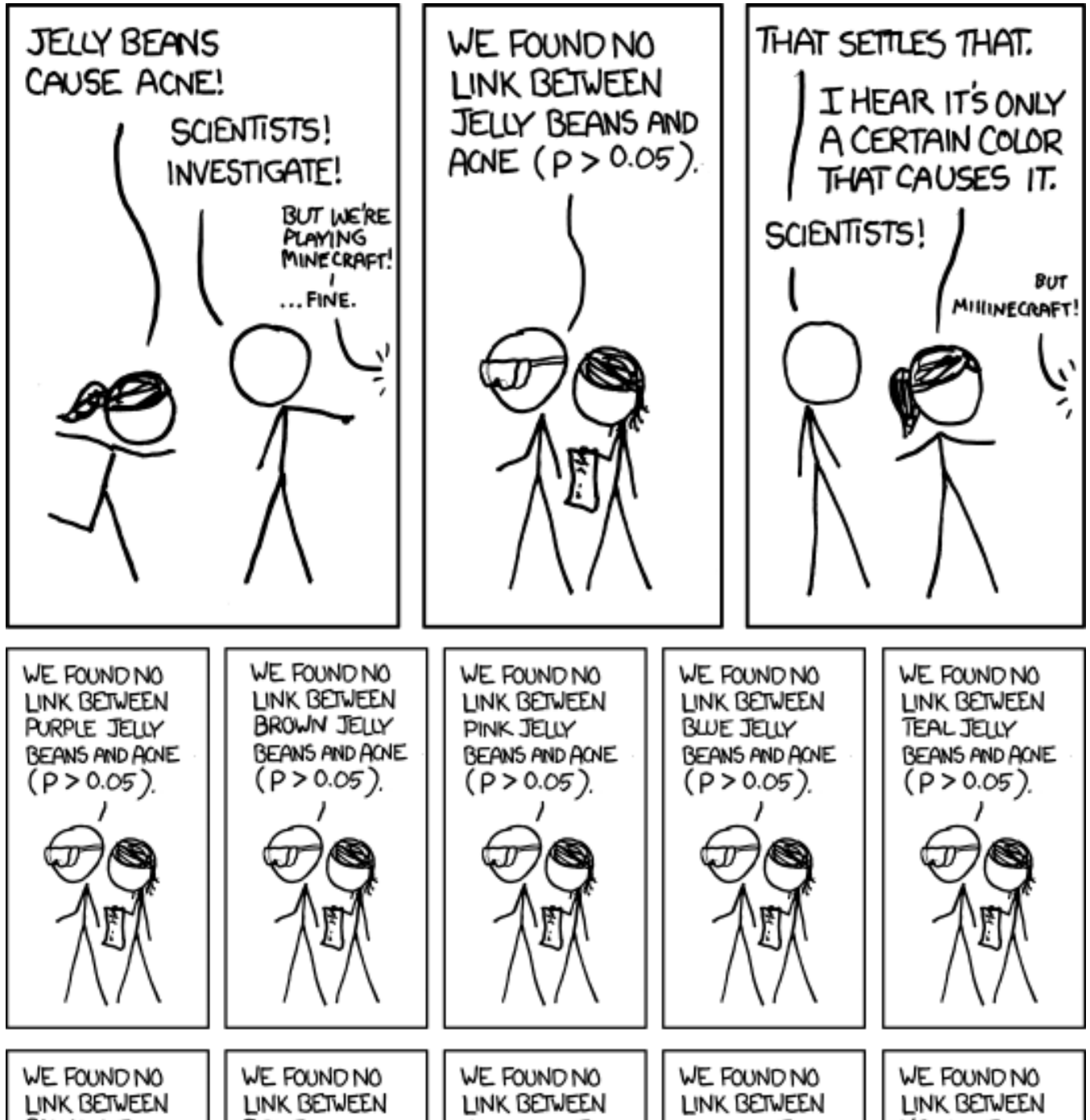
- Use a third subset - test set – to verify you made a good choice
  - Only use for reporting!
  - Do not change method after

Overfitting – 3 reasons

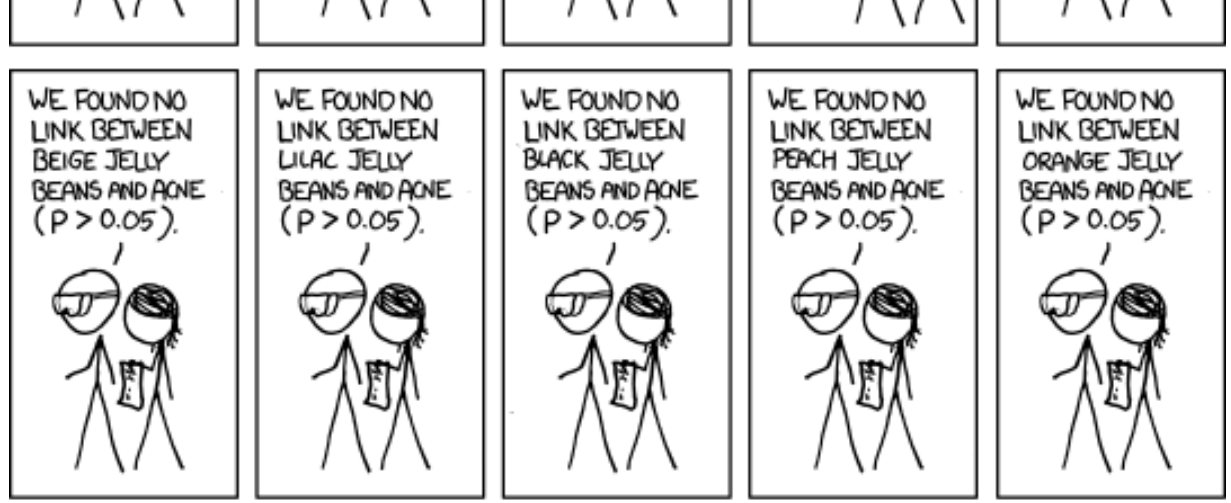
Overfitting

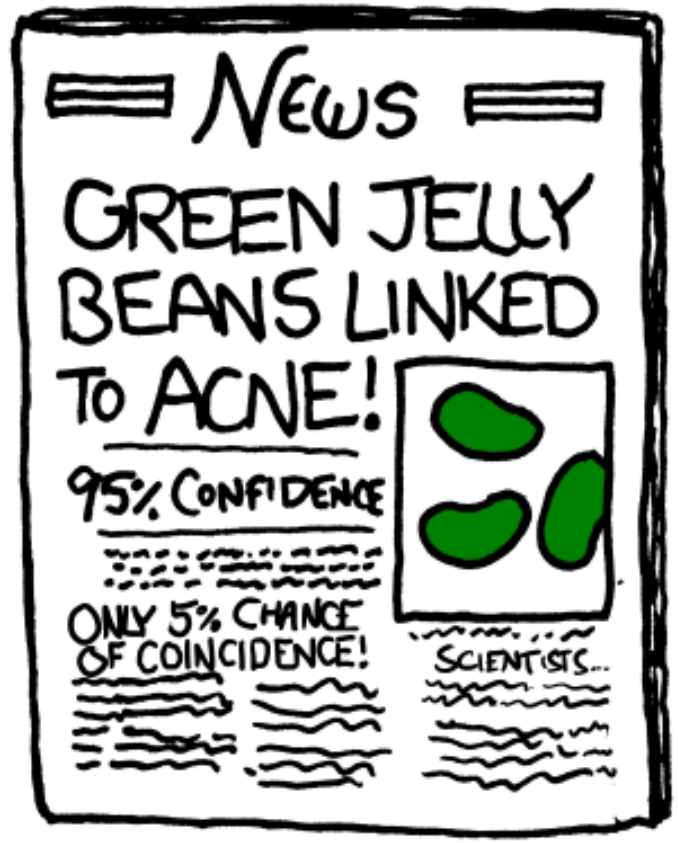
- The classifier has learnt patterns in the training data (training error is low)
- These are not the true patterns / they do not apply to the test data
- The classifier can **not** create good predictions for the test data (test error is high)

30



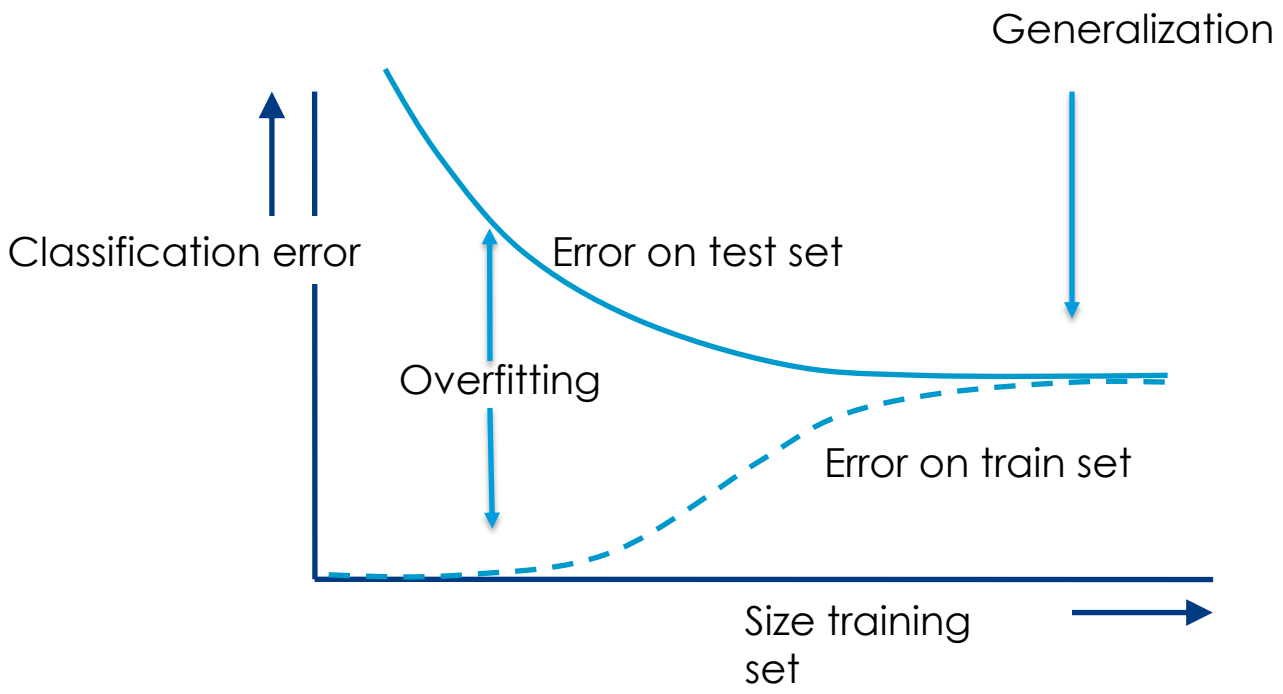
Source: xkcd.com





Source: xkcd.com

Generalization vs overfitting

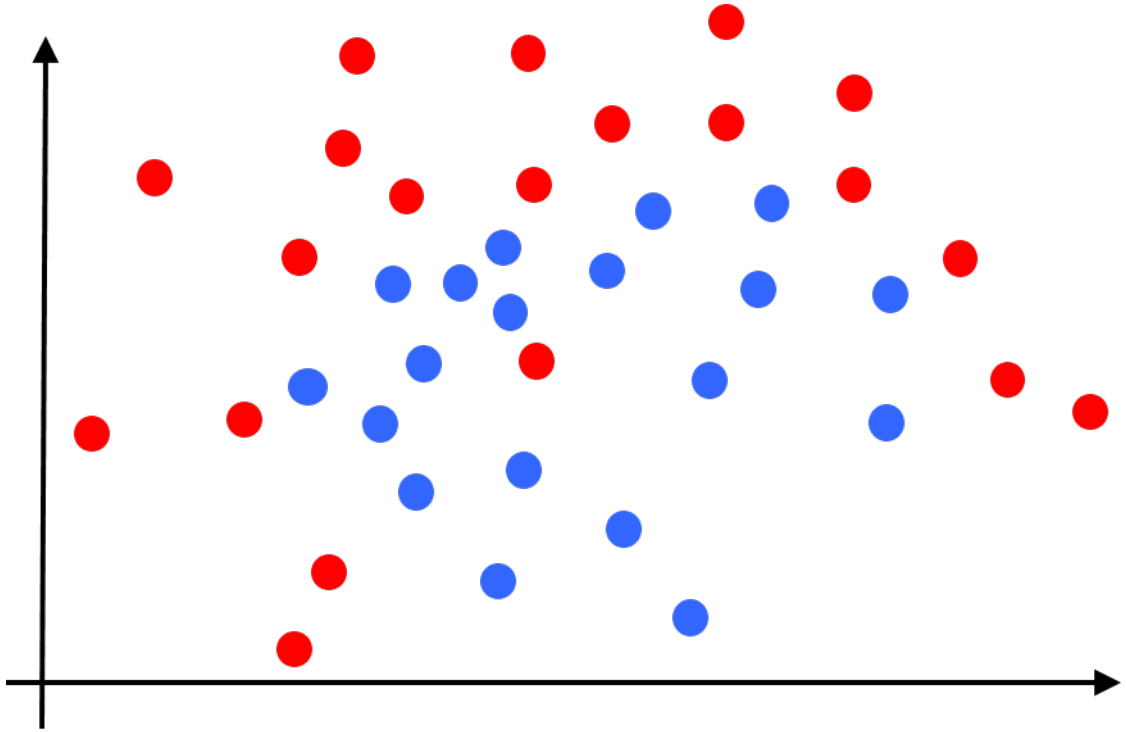


Overfitting when

1) Too few representative training examples

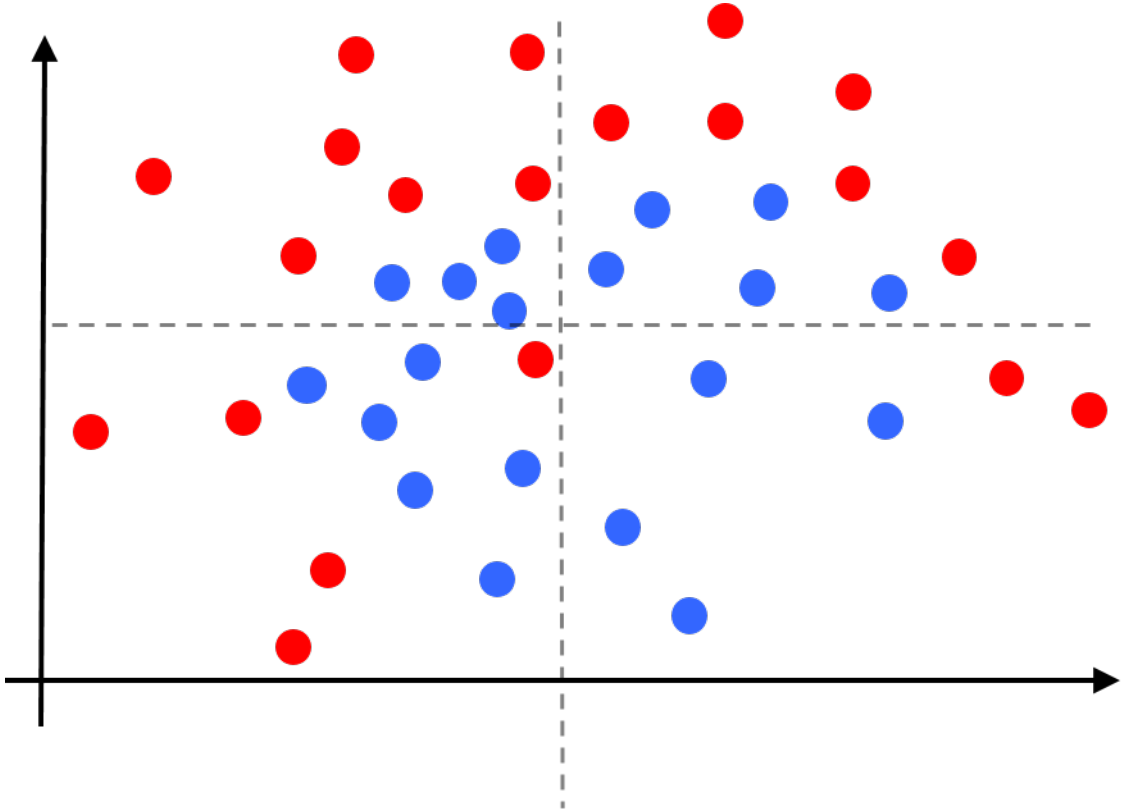
2) …

Overfitting reason 2 – Let's look at this example dataset



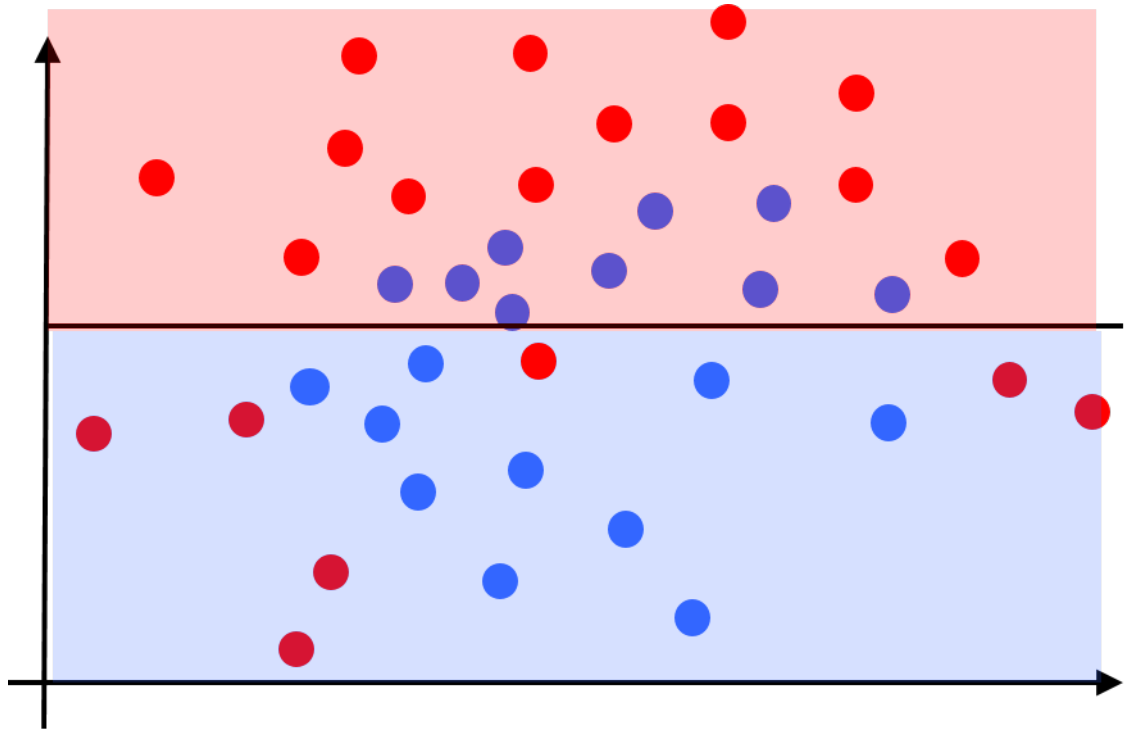
We use the “histogram-based classifier” (not used in practice, but related to kNN)

Divide space into cells, label each cell by majority of samples

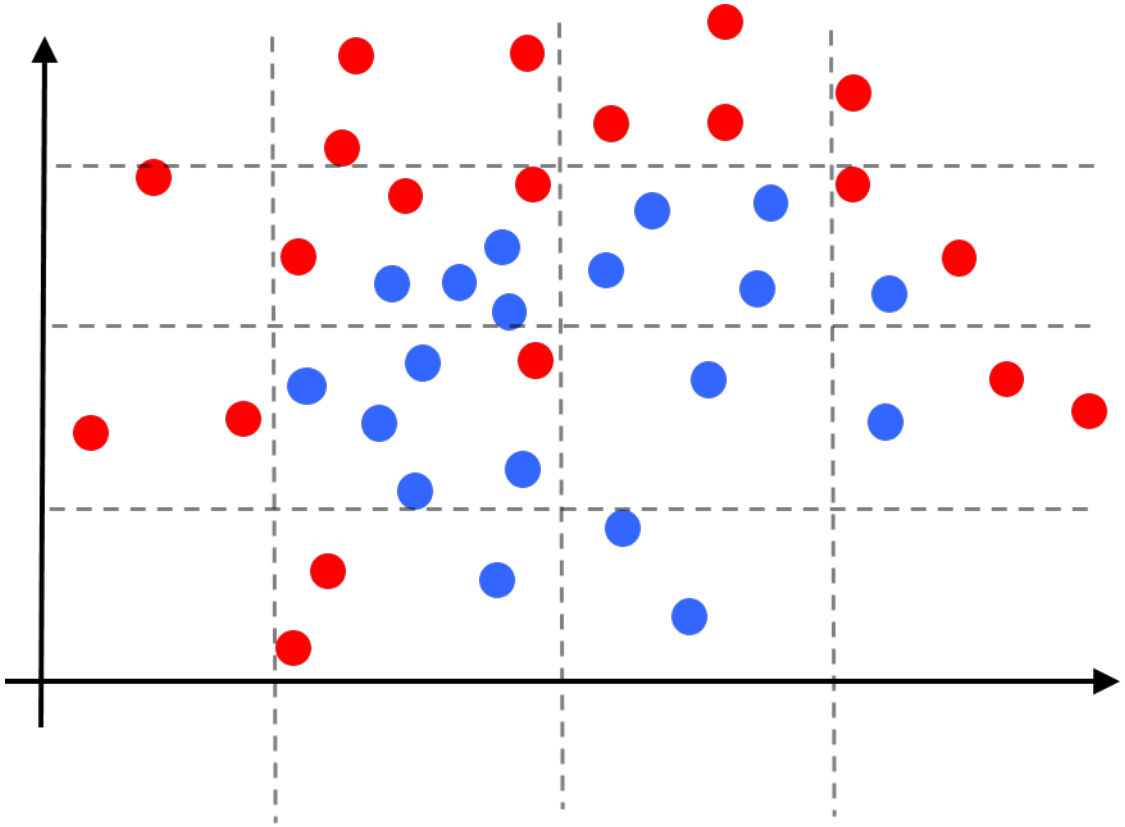


Number of bins determines how simple/complex the model is

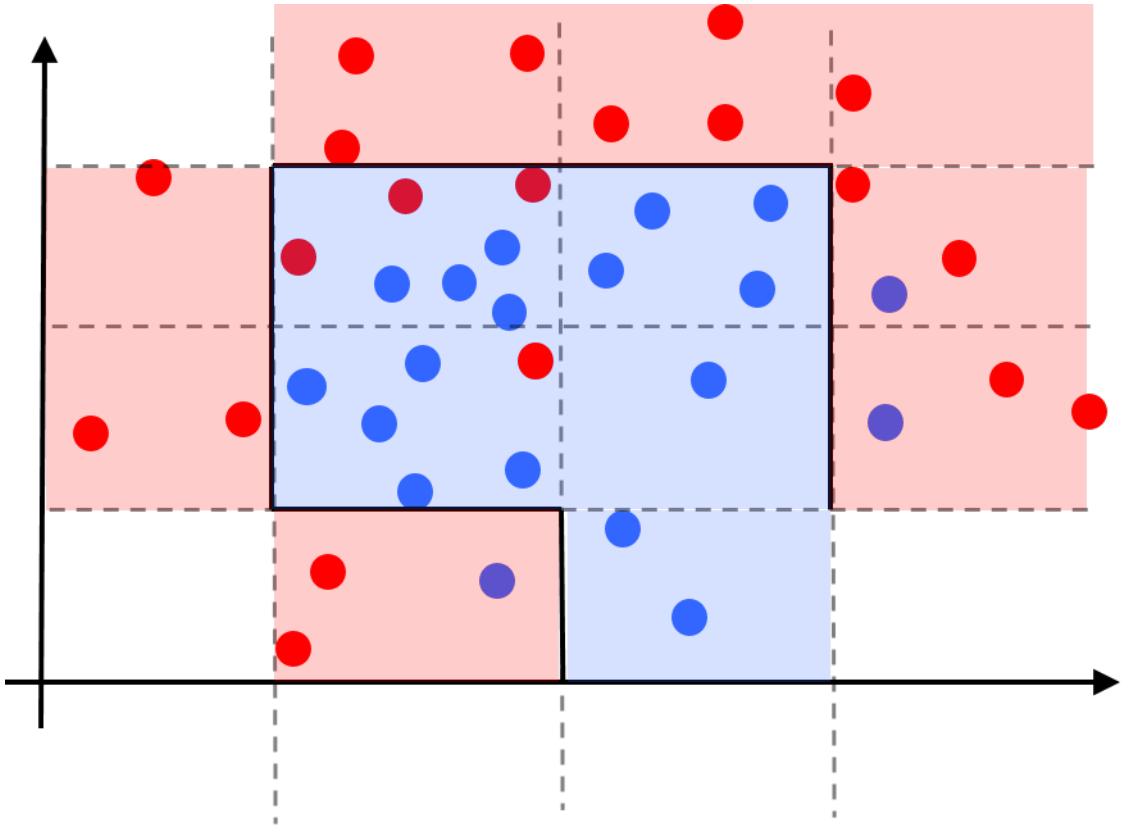
- Underfitting: model is too simple, cannot capture the patterns in the data
- High training error, high test error



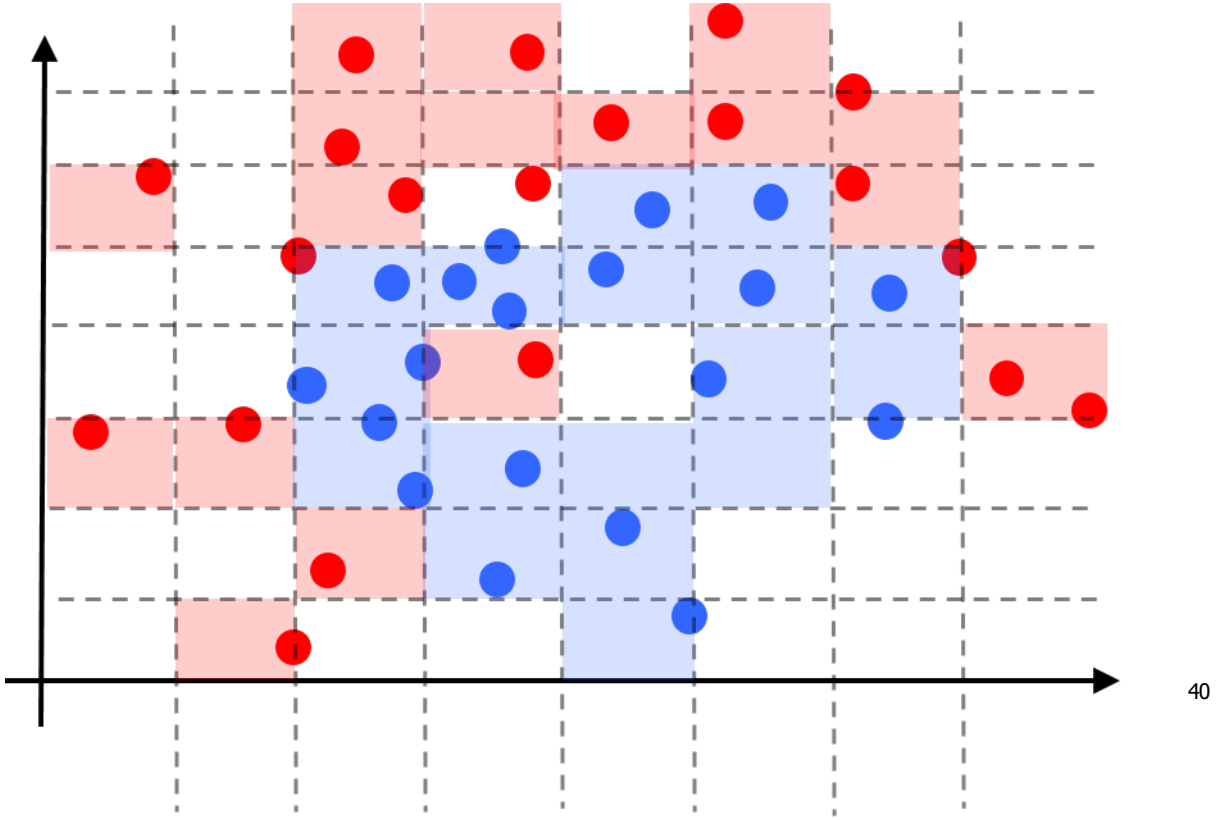
We can increase complexity (and fit the data better) by adding more cells



This is a better model, but there are still some errors

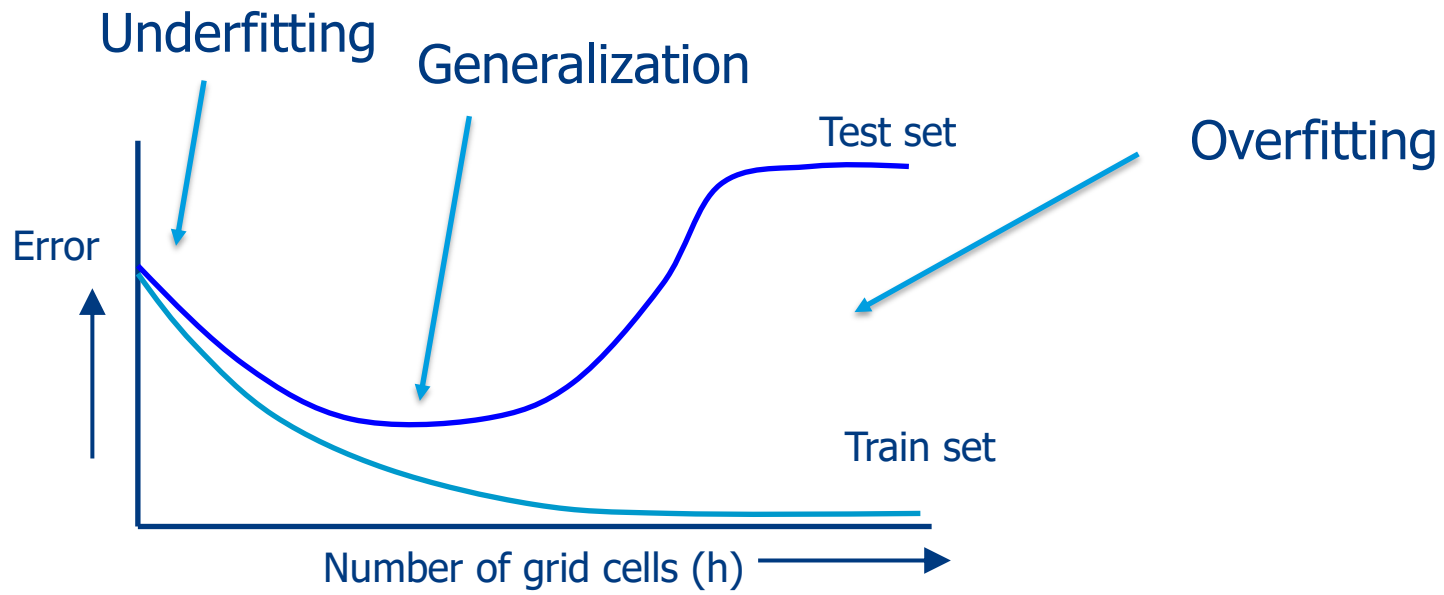


Increasing complexity again will fit training data perfectly, but the decisions will be noisy - overfitting has occured



Overfitting

- More cells = more complexity = can more easily capture patterns in data



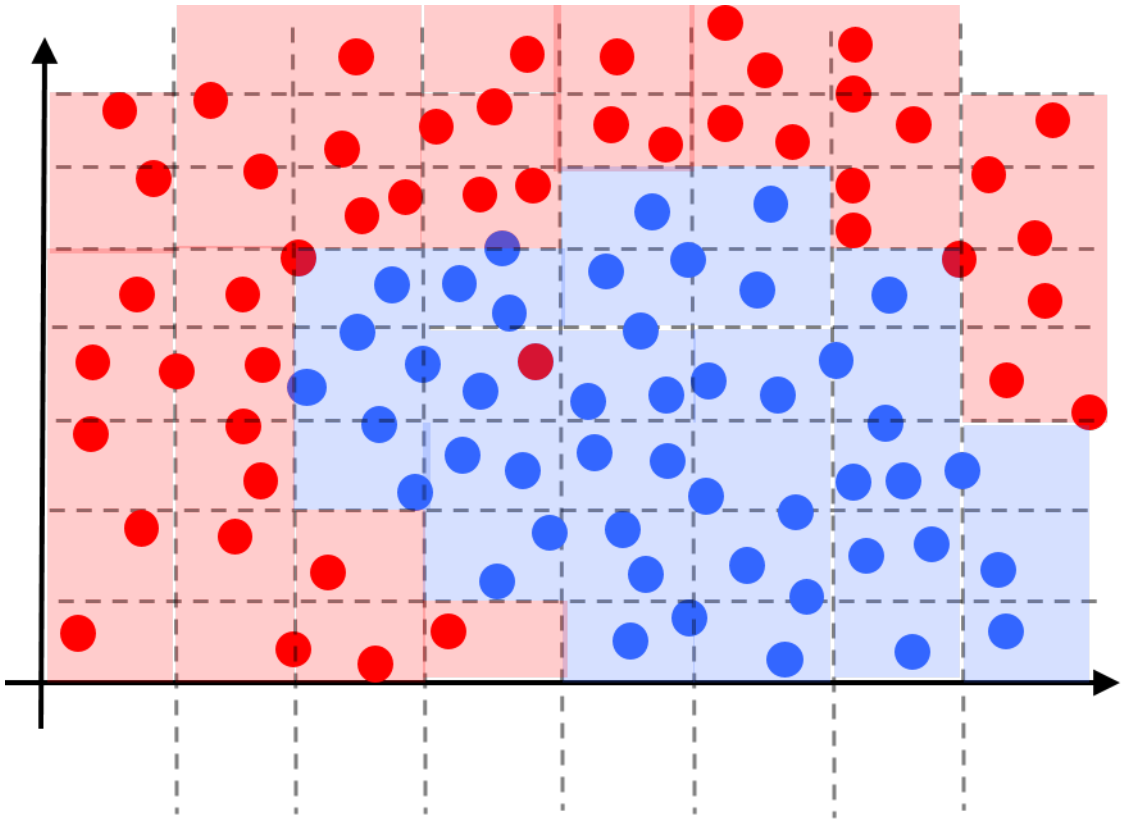
Overfitting when

1) Too few representative training examples +

2) Too complex classifier +

3) …

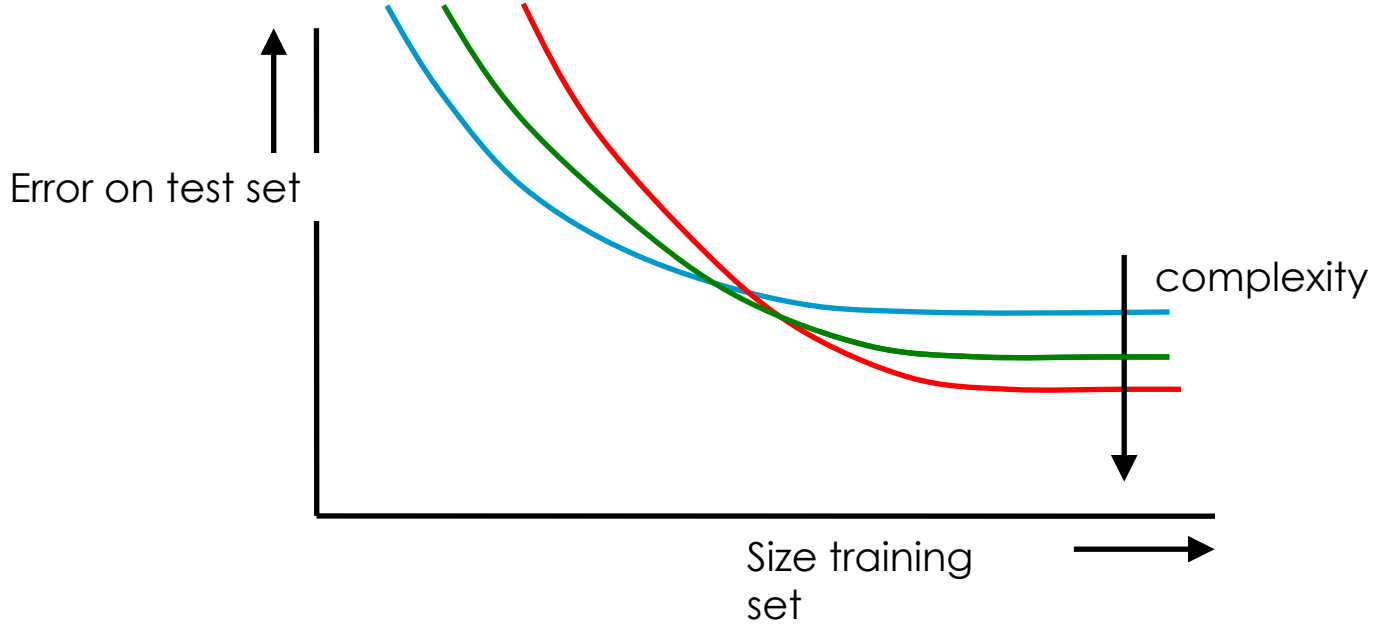
The complexity the classifier can have, depends on the number of samples



43

Trade-off sample size and classifier complexity

- Use simple classifiers (e.g. nearest mean or another linear classifier) when data is small



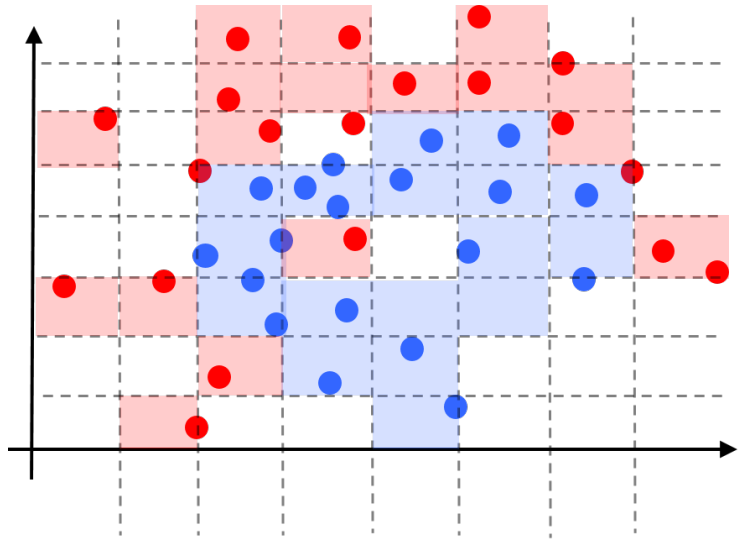
Nearest mean could be the blue classifier, 1-nearest neighbor the red one

Overfitting reason 3 –

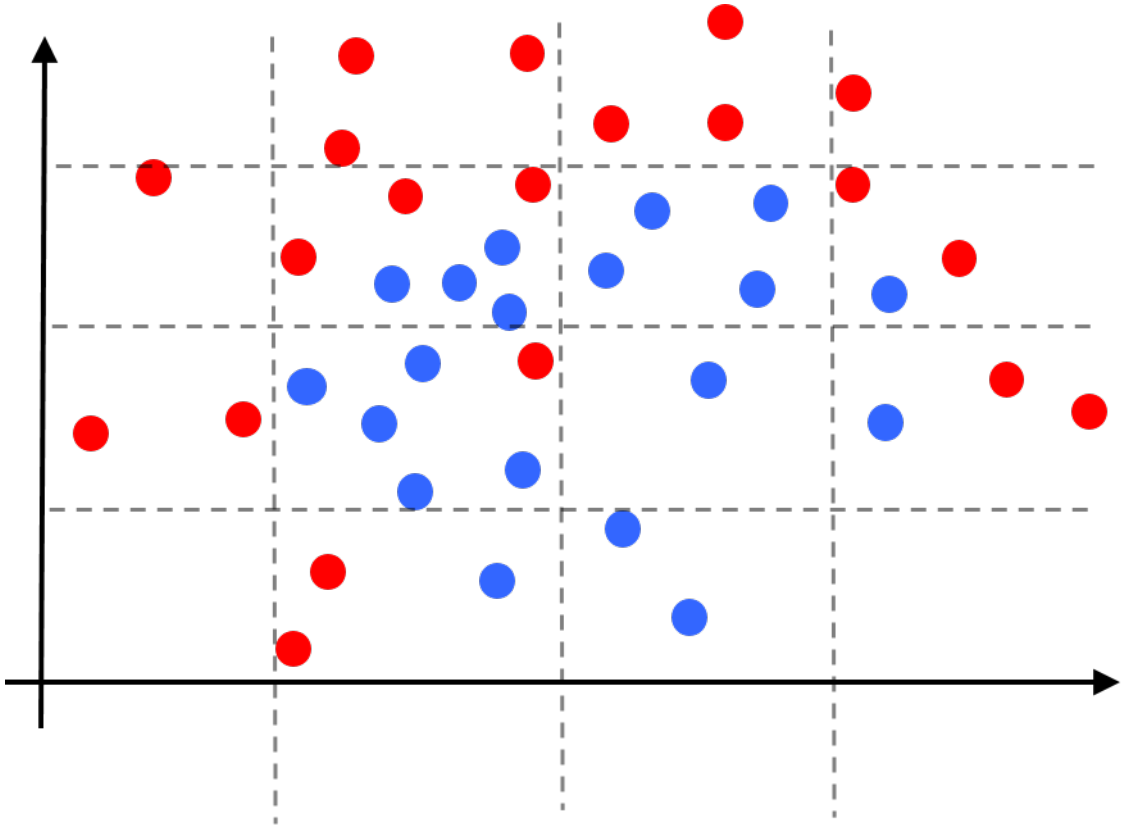
What if instead of adding more compexity to the model, we add more features?

Imagine we add a third feature, “how close is this point to the center of the data”

Blue points will have lower values on this feature (closer to center)

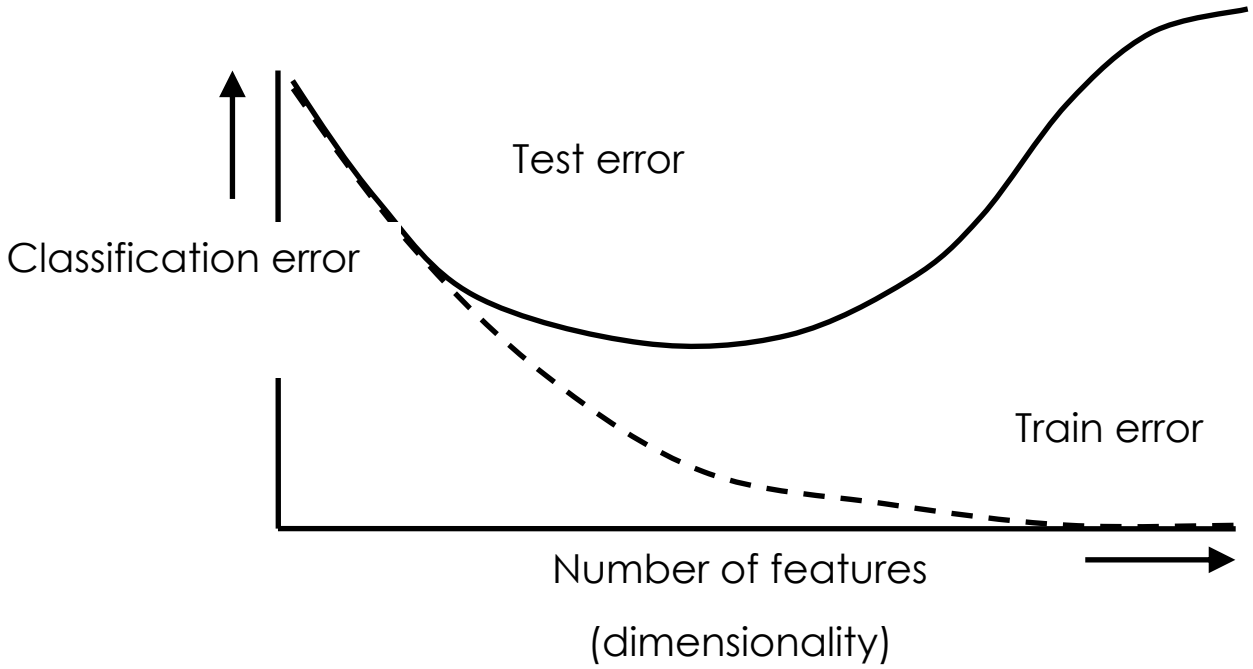


- In 2D we need at least 16 samples to fill each cell
- With a third dimension, we need more samples (64 if we use 4 bins)

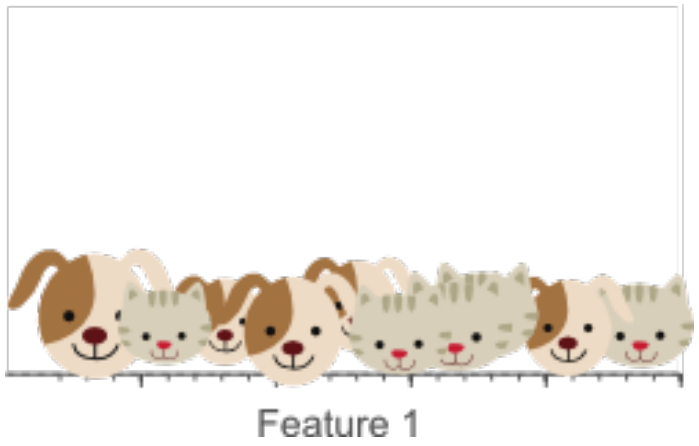


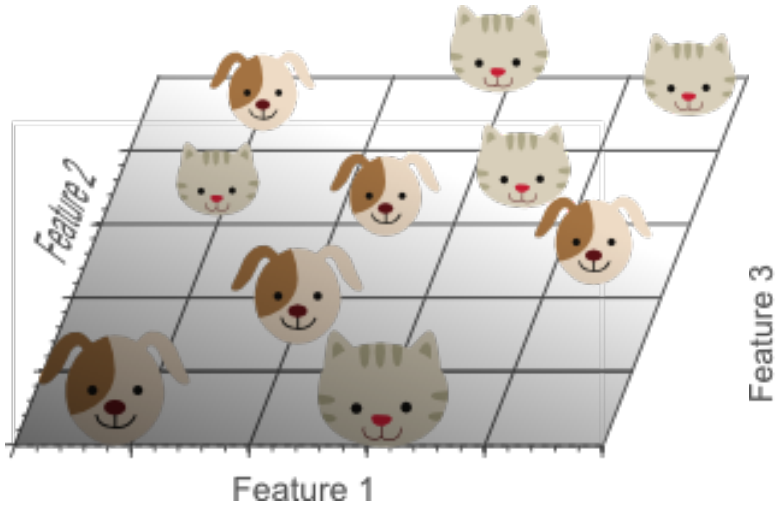
"**Curse of dimensionality**"

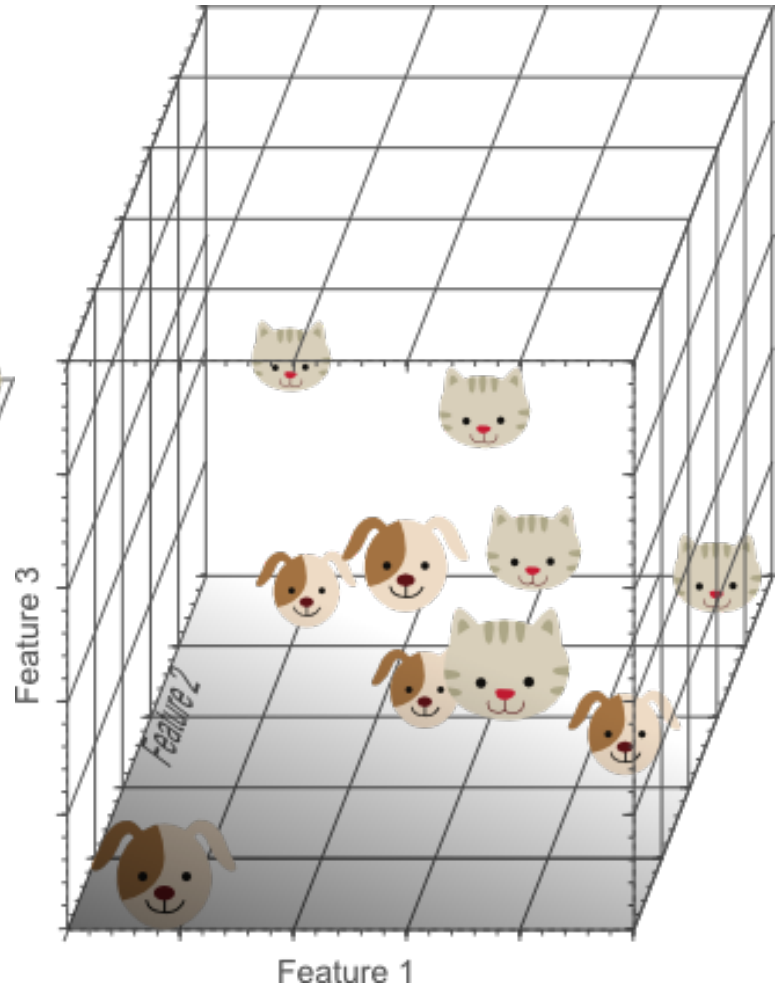
The more dimensions you add, the "emptier" the space becomes, the easier it is to perfectly fit a model to the training data



Further reading:
https://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/



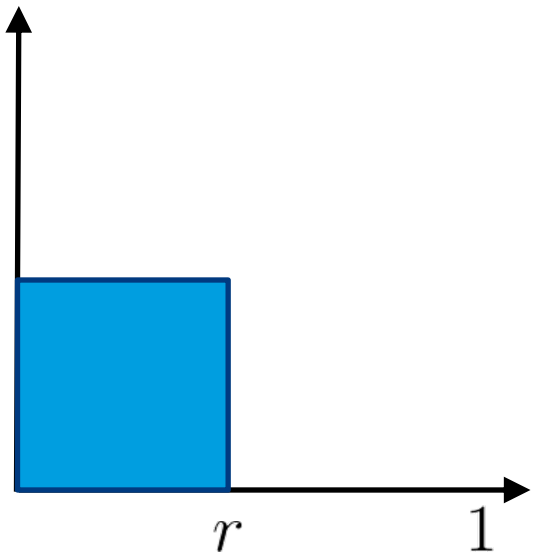




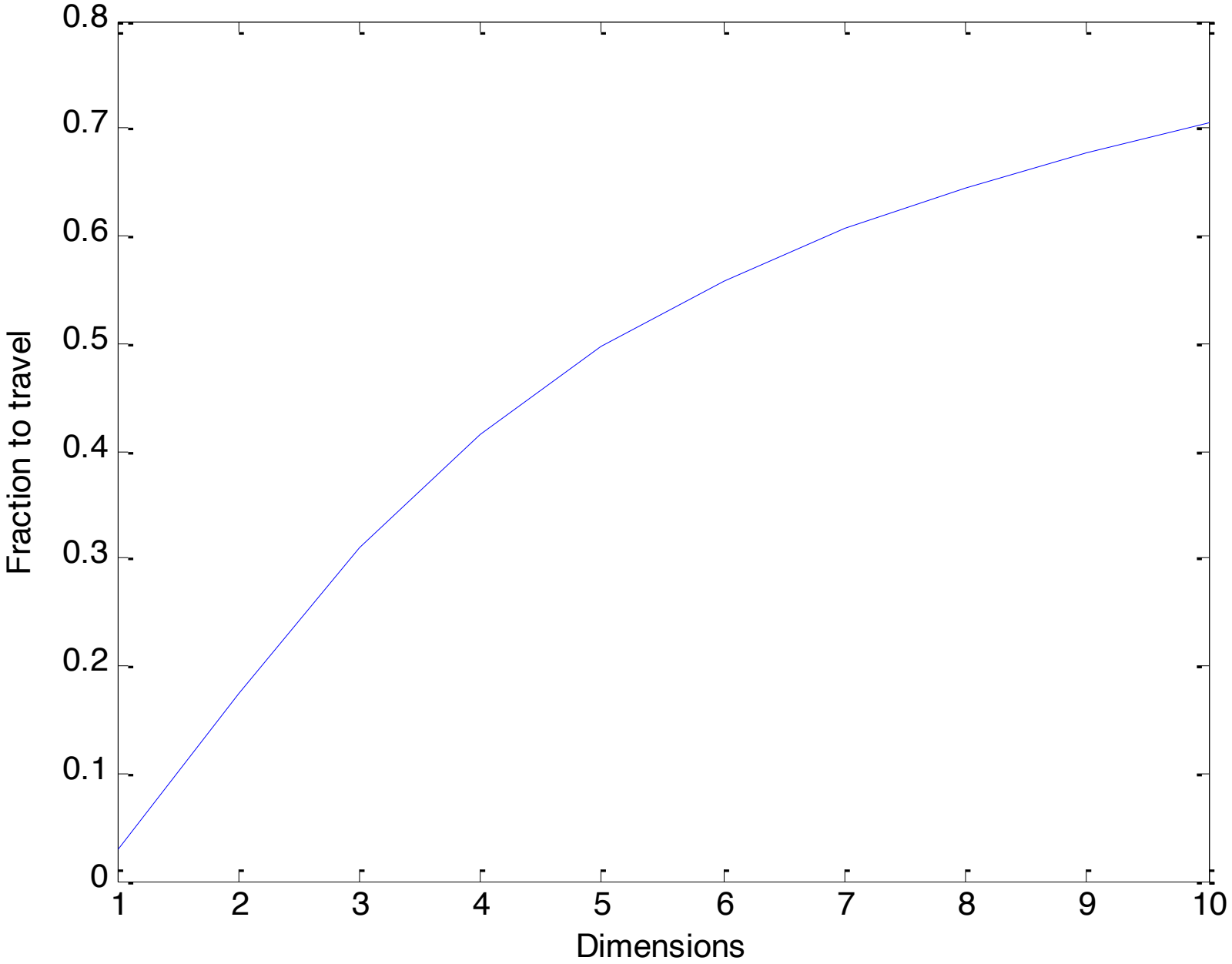
Thought exercise

- Consider 100 uniformly distributed points in a unit line / square / cube …
- Your goal is to find 1 nearest neighbor (1% of the points, V = 0.01)
- How far (what is the value of r) do you need to travel on a line? (m = 1)
- How far to travel per dimension, when m = 2?

$$r^m = V$$



- In 10 dimensions, need to explore 70% of each dimension to find a neighbor!



Overfitting when

1) Too few representative training examples +

2) Too complex classifier +

3) Too many features

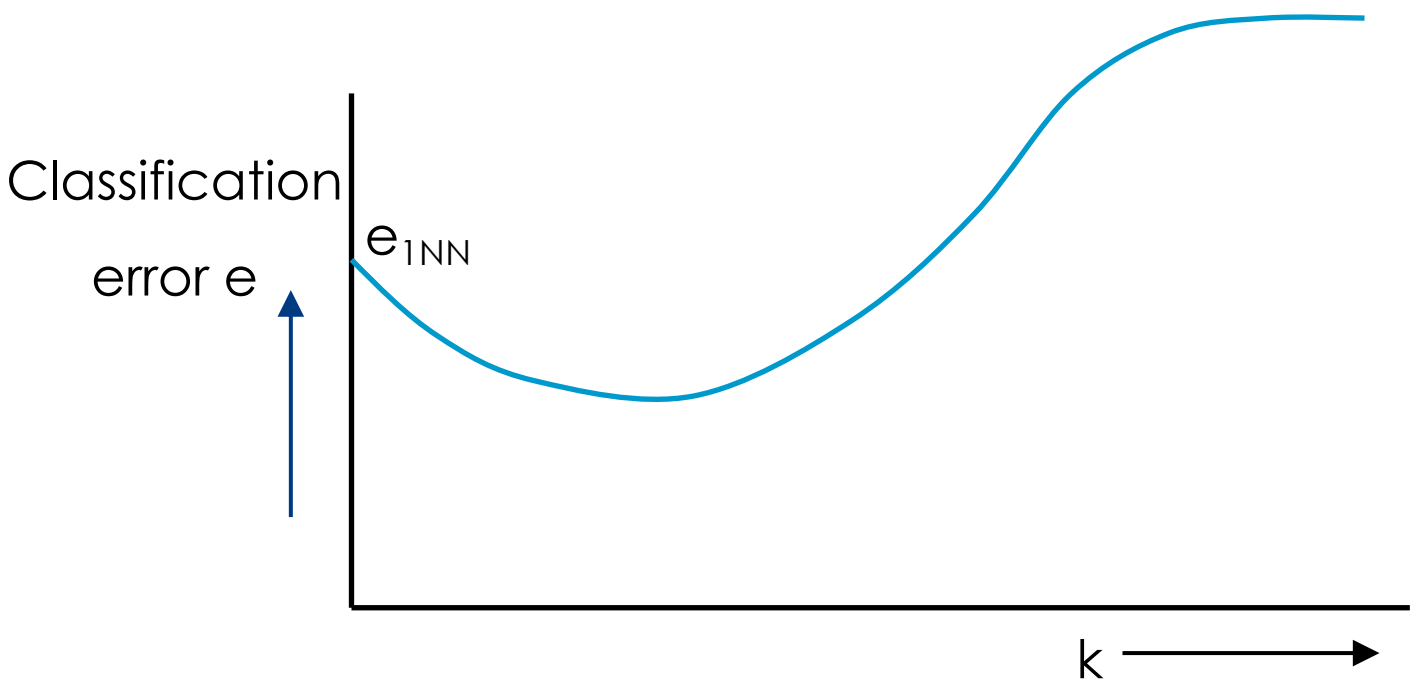
- Generalization
  - Training error ≈ test error

- Overfitting
  - Training error << test error

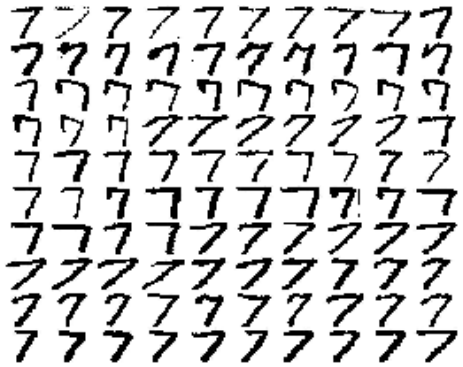
  - Too few (representative) objects
  - Too complex classifier
  - Too many dimensions

K-NN & generalization/overfitting:

- Likely overfitting when k=1
- Likely underfitting when k=N



- Because of its complexity, kNN is more suitable for datasets with many training objects
  - E.g. number recognition
  - Pixel as feature, PCA to 0.9 variance, 5-NN → 1% error



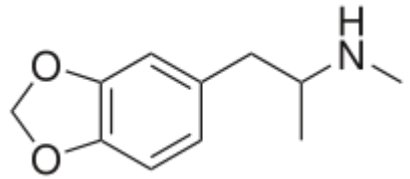
For datasets with few training objects, linear classifiers like **nearest mean** might be more suitable

- Brain data (79 in 2 classes)

Stoeckel, Jonathan, et al. "Classification of SPECT images of normal subjects versus images of Alzheimer's disease patients." *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2001*. Springer Berlin Heidelberg, 2001.

- Mass spectrometry (57 in 3 classes)

Veenman, Cor J., and Annabel Bolck. "A sparse nearest mean classifier for high dimensional multi-class problems." *Pattern Recognition Letters* 32.6 (2011): 854-859.



- We can “diagnose” generalization or overfitting by looking at the training error and test error
- **But** by looking at the test error, the test data becomes part of the training data!
- Use train & validation to choose classifier, test only for reporting
- More on this in the next lecture