Lecture 8: Evaluation

COMP 343, Spring 2022

Victoria Manfredi

WESLEYAN
UNIVERSITY

Acknowledgements: These slides are based primarily on material from the book Machine Learning by Tom Mitchell (and associated slides), the book Machine Learning, An Applied Mathematics Introduction by Paul Wilmott, slides created by Vivek Srikumar (Utah), Dan Roth (Penn), Jessica Wu (Harvey Mudd) and C. David Page (U of Wisconsin-Madison)

Today's Topics

Homework 4 (almost) out

- Due Thursday, March 3 by 5p

Overfitting

- What is overfitting?
- Avoiding overfitting

Evaluation

- Cross-validation
- Evaluation metrics
- (Bias and variance)

vumanfredi@wesleyan.edu

Homework 3 discussion

Decision tree questions?

Python questions?

Homework 4 discussion

https://scikit-learn.org/stable/modules/tree.html#tree

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Decision Trees

OVERFITTING

Overfitting

Goal for today

- What is overfitting?
- How we can combat overfitting for decision trees?

Keep in mind

- Overfitting is an issue not just to decision trees but pretty much every single machine learning algorithm

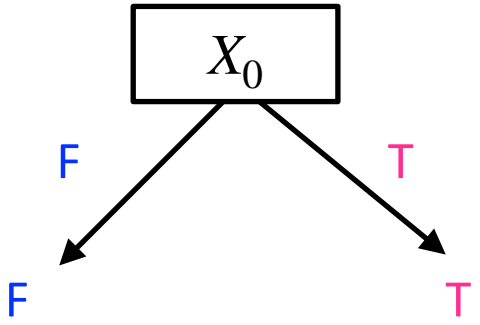
The "First Bit" function

A Boolean function with n inputs

Simply returns the value of the first input, all others irrelevant

X_0	X_1	Y
F	F	F
F	T	F
T	F	T
T	T	T

What is the decision tree for this function?



$$Y = X_0$$

X_1 is irrelevant

Noisy data

What if the data is noisy? And we have all 2^n examples

X_0	X_1	X_2	Y
F	F	F	F
F	F	T	F T
F	T	F	F
F	T	T	F
T	F	F	T
T	F	T	T F
T	T	F	T
T	T	T	T

Suppose, the outputs of both training and test sets are randomly corrupted

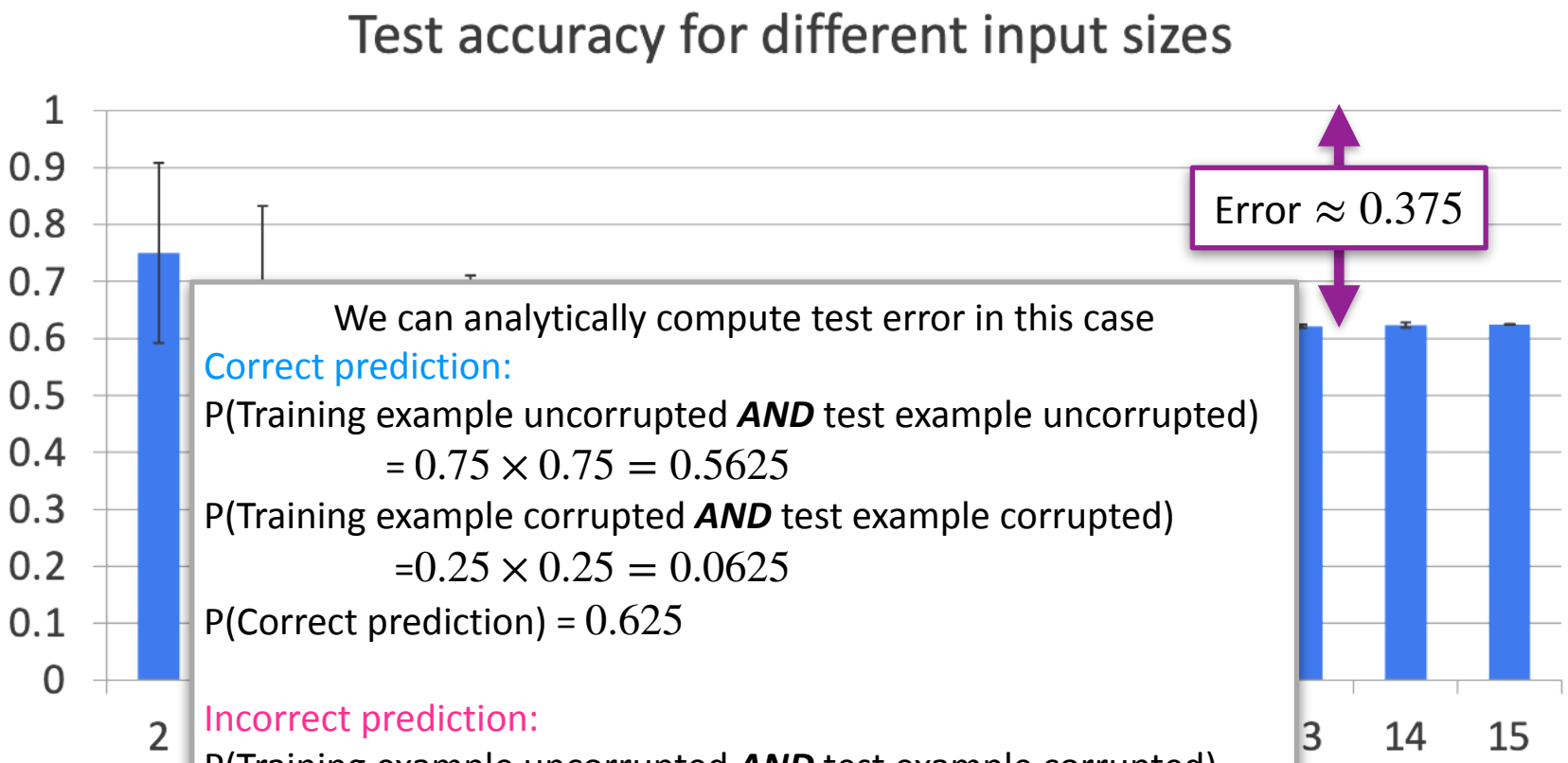
Train and test sets are no longer identical

Both have noise, possibly different

Prediction accuracy drops because there is noise!

Suppose output corrupted with probability 0.25

Suppose data is noisy and we have all 2^n examples



We can analytically compute test error in this case

Correct prediction:

P(Training example uncorrupted ***AND*** test example uncorrupted)

$= 0.75 \times 0.75 = 0.5625$

P(Training example corrupted ***AND*** test example corrupted)

$= 0.25 \times 0.25 = 0.0625$

P(Correct prediction) = 0.625

Incorrect prediction:

P(Training example uncorrupted ***AND*** test example corrupted)

$= 0.75 \times 0.25 = 0.1875$

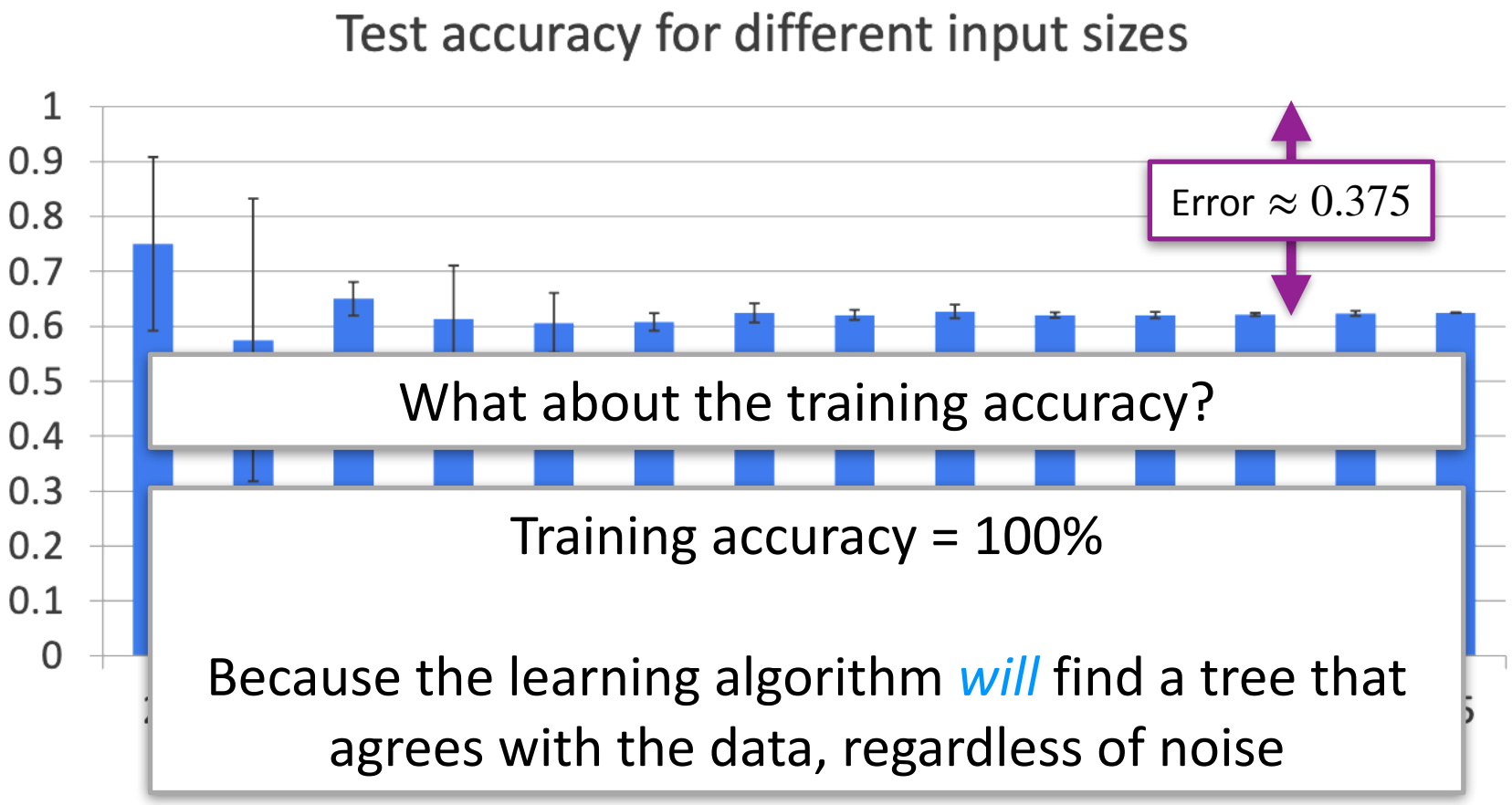
P(Training example corrupted ***AND*** test example uncorrupted)

$= 0.25 \times 0.75 = 0.1875$

P(Incorrect prediction) = 0.375

Suppose output corrupted with probability 0.25

Suppose data is noisy and we have all 2^n examples



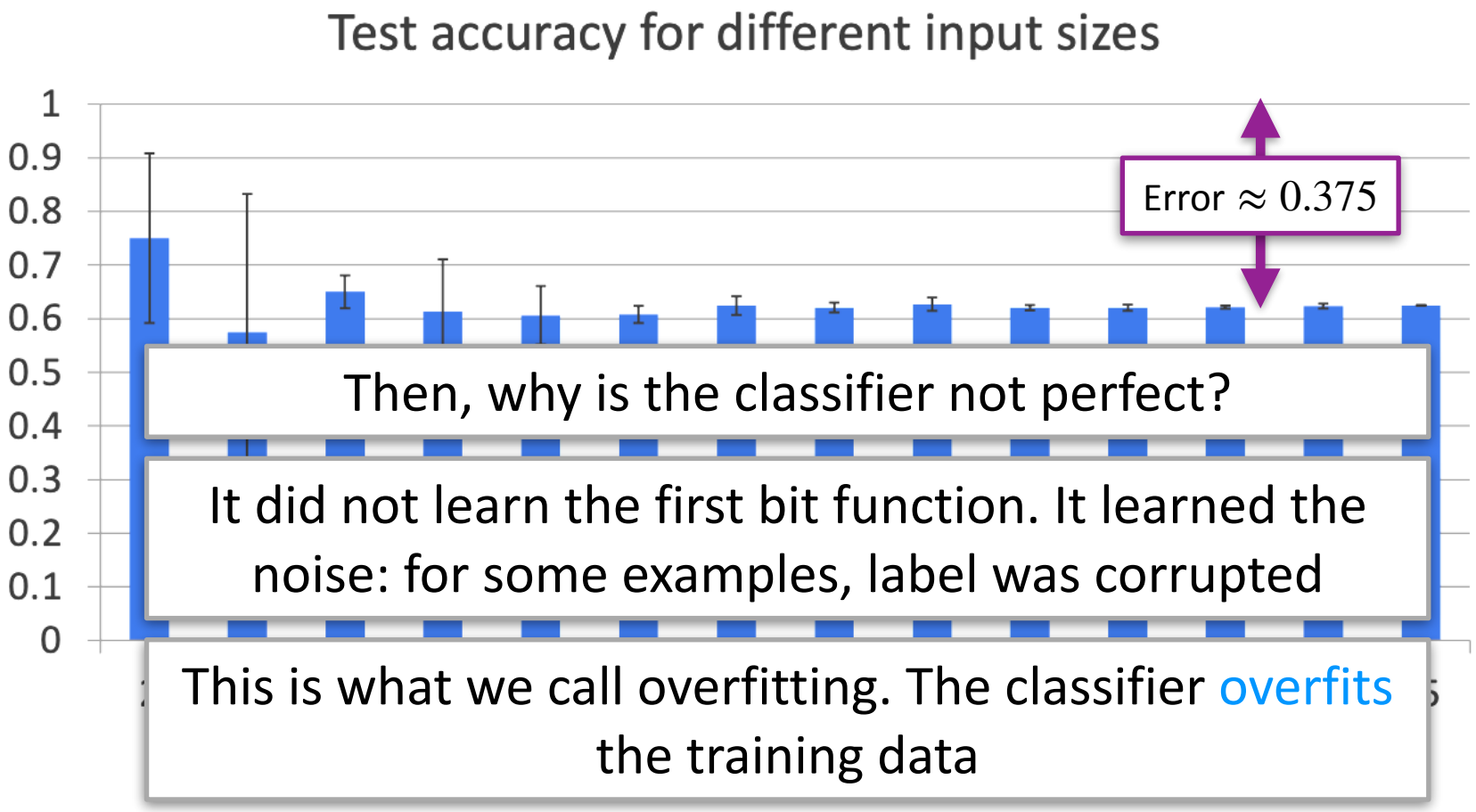
What about the training accuracy?

Training accuracy = 100%

Because the learning algorithm *will* find a tree that agrees with the data, regardless of noise

Suppose output corrupted with probability 0.25

Suppose data is noisy and we have all 2^n examples



Overfitting

You can think of overfitting as when the learning algorithm finds a hypothesis that fits the noise in the data

- Irrelevant attributes or noisy examples influence the choice of the hypothesis

Overfitting

You can think of overfitting as when the learning algorithm finds a hypothesis that fits the noise in the data

- Irrelevant attributes or noisy examples influence the choice of the hypothesis

Why is this bad?

- May lead to poor performance on future examples

Every learning algorithm needs to have a way of combatting overfitting

One definition of overfitting

Suppose our data (X, Y) is generated from a probability distribution $D(X, Y)$ over instances X and labels Y. We do not know this distribution. Suppose we are using a hypothesis space H

Two kinds of important errors:

Training error for hypothesis $h \in H$: $error_{train}(h)$

True error for $h \in H$: $error_{D}(h)$

One definition of overfitting

Suppose our data (X, Y) is generated from a probability distribution $D(X, Y)$ over instances X and labels Y. We do not know this distribution. Suppose we are using a hypothesis space H

Two kinds of important errors:

Training error for hypothesis $h \in H$: $error_{train}(h)$

- Fraction of training examples on which hypothesis h makes a mistake
- We can calculate this!

True error for $h \in H$: $error_D(h)$

One definition of overfitting

Suppose our data (X, Y) is generated from a probability distribution $D(X, Y)$ over instances X and labels Y. We do not know this distribution. Suppose we are using a hypothesis space H

Two kinds of important errors:

Training error for hypothesis $h \in H$: $error_{train}(h)$

- Fraction of training examples on which hypothesis h makes a mistake
- We can calculate this!

True error for $h \in H$: $error_D(h)$

- Expected error that hypothesis h makes on entire set of examples that exist according to underlying distribution … even examples we have not seen. Different examples weighted by how probable they are
- Mathematical concept: we cannot calculate but still care about!

One definition of overfitting

Suppose our data (X, Y) is generated from a probability distribution $D(X, Y)$ over instances X and labels Y. We do not know this distribution. Suppose we are using a hypothesis space H

Two kinds of important errors:

Training error for hypothesis $h \in H$: $error_{train}(h)$

- Fraction of training examples on which hypothesis h makes a mistake
- We can calculate this!

True error for $h \in H$: $error_D(h)$

- We use a held-out validation set to evaluate performance. If set is large enough, will approximate true error.
-

One definition of overfitting

Training error for hypothesis $h \in H$: $error_{train}(h)$

True error for $h \in H$: $error_D(h)$

A hypothesis h overfits the training data if there is another hypothesis h' such that

1. h has lower training error than the competing hypothesis h' but
   - $error_{train}(h) < error_{train}(h')$
2. h' generalizes better than h
   - $error_D(h) > error_D(h')$

One definition of overfitting

Training error for hypothesis $h \in H$: $error_{train}(h)$

True error for $h \in H$: $error_D(h)$

A hypothesis h overfits the training data if there is another hypothesis h' such that

1. h has lower training error than the competing hypothesis h' but
   - $error_{train}(h) < error_{train}(h')$
2. h' generalizes better than h
   - $error_D(h) > error_D(h')$

Which hypothesis has the lowest generalization error?

One definition of overfitting

Training error for hypothesis $h \in H$: $error_{train}(h)$

True error for $h \in H$: $error_D(h)$

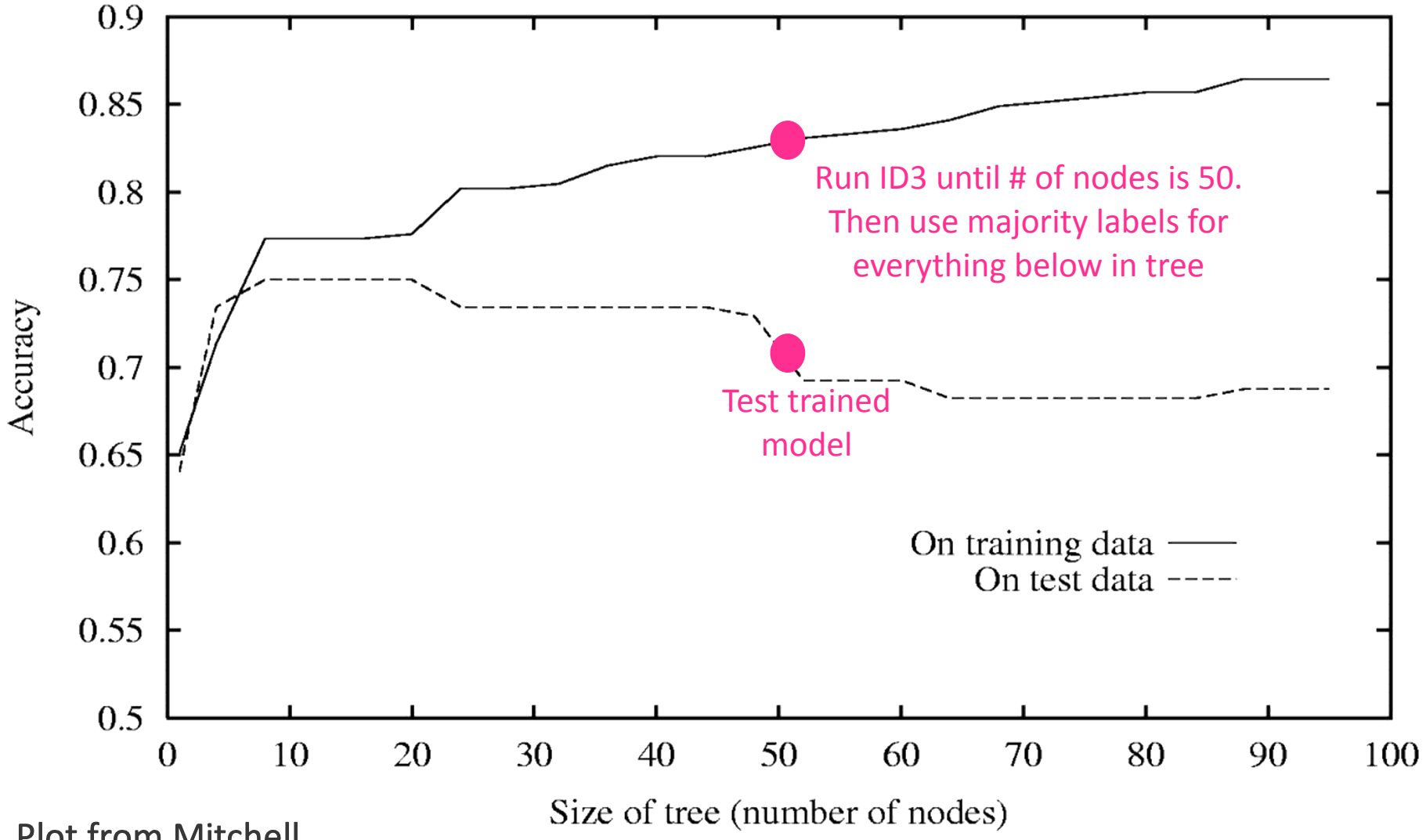
A hypothesis h overfits the training data if there is another hypothesis h' such that

1. h has lower training error than the competing hypothesis h' but
   - $error_{train}(h) < error_{train}(h')$
2. h' generalizes better than h
   - $error_D(h) > error_D(h')$

Which hypothesis has the lowest generalization error?

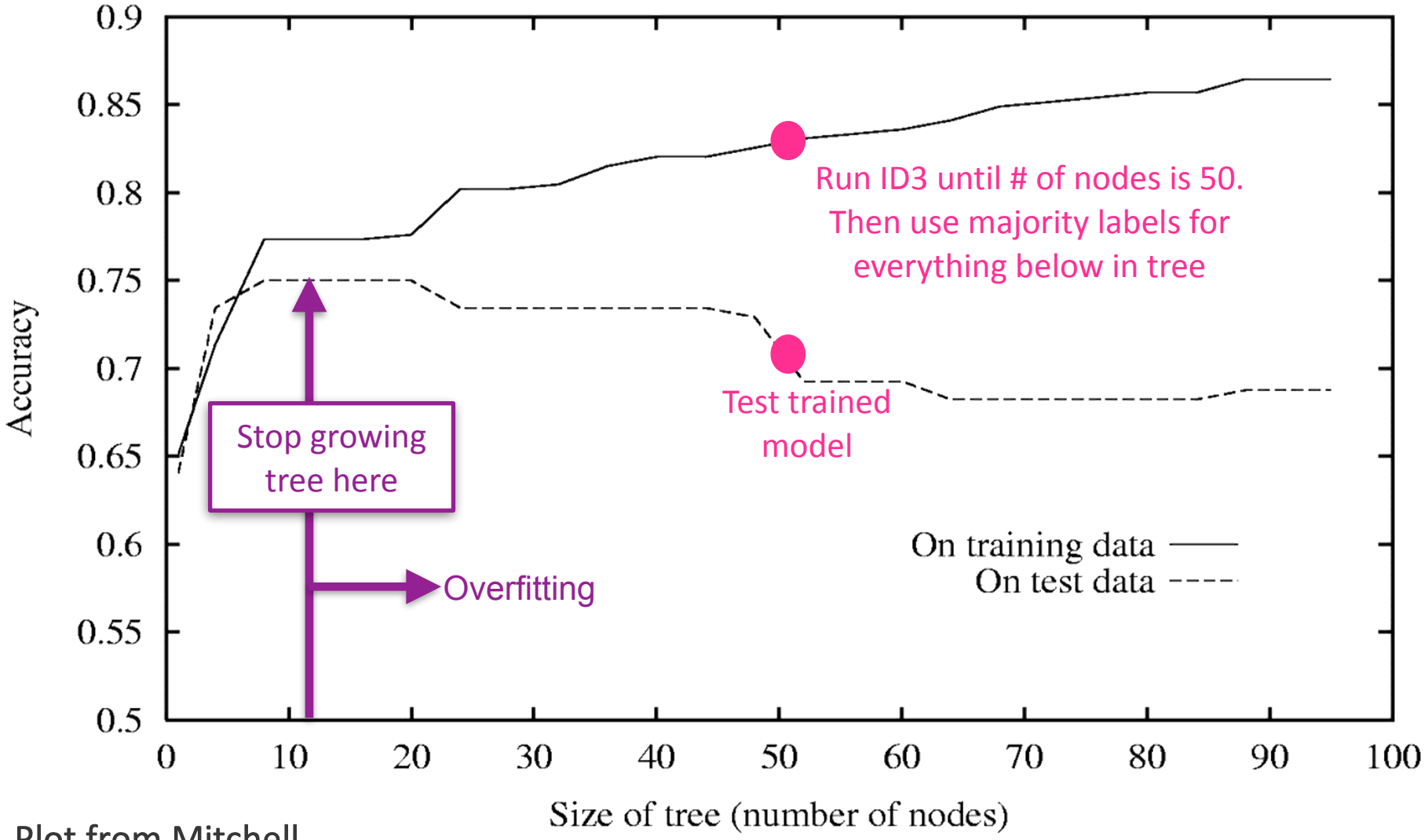
The target or true function f. But f may not be in H

Decision trees will almost certainly overfit



Plot from Mitchell

Decision trees will almost certainly overfit



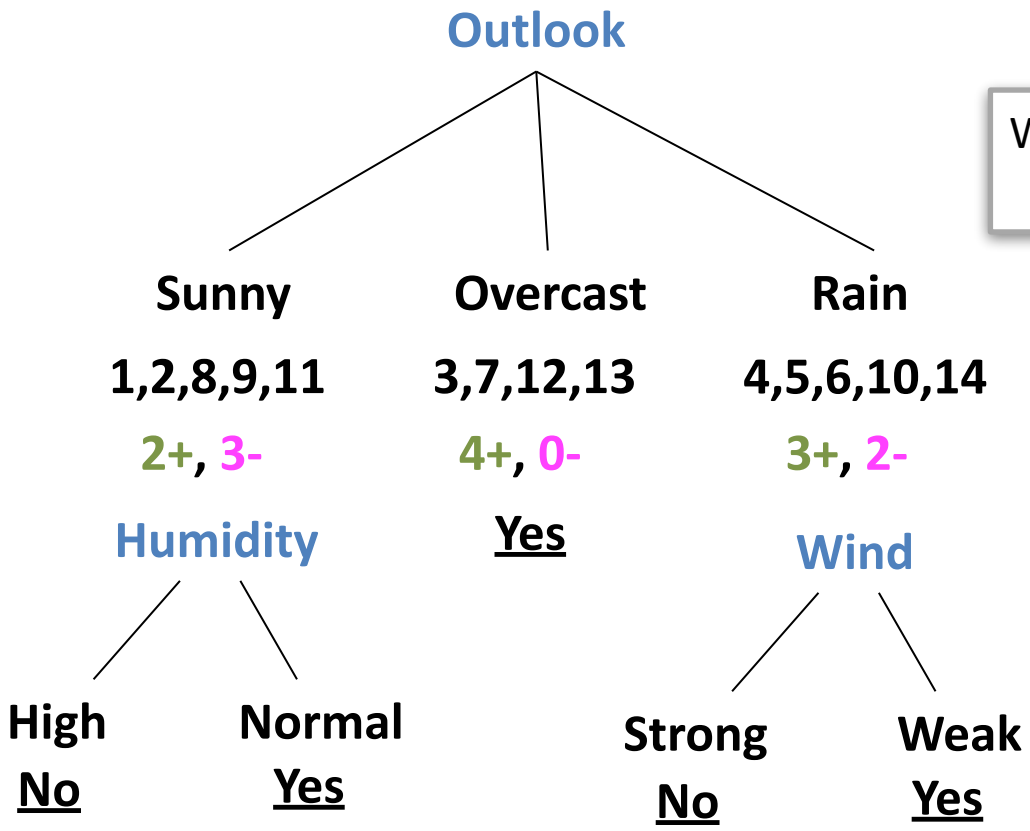
Plot from Mitchell

Decision Trees

AVOIDING OVERFITTING

Example

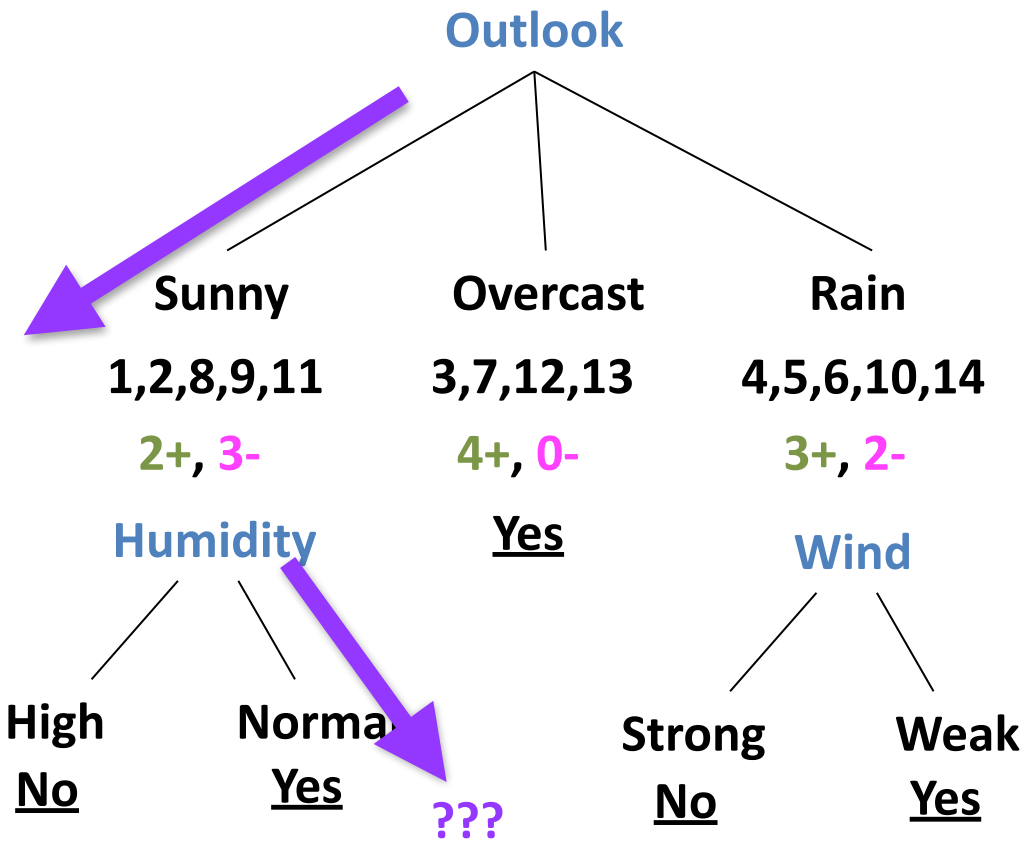
Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, **No**



What does decision tree say for this example?

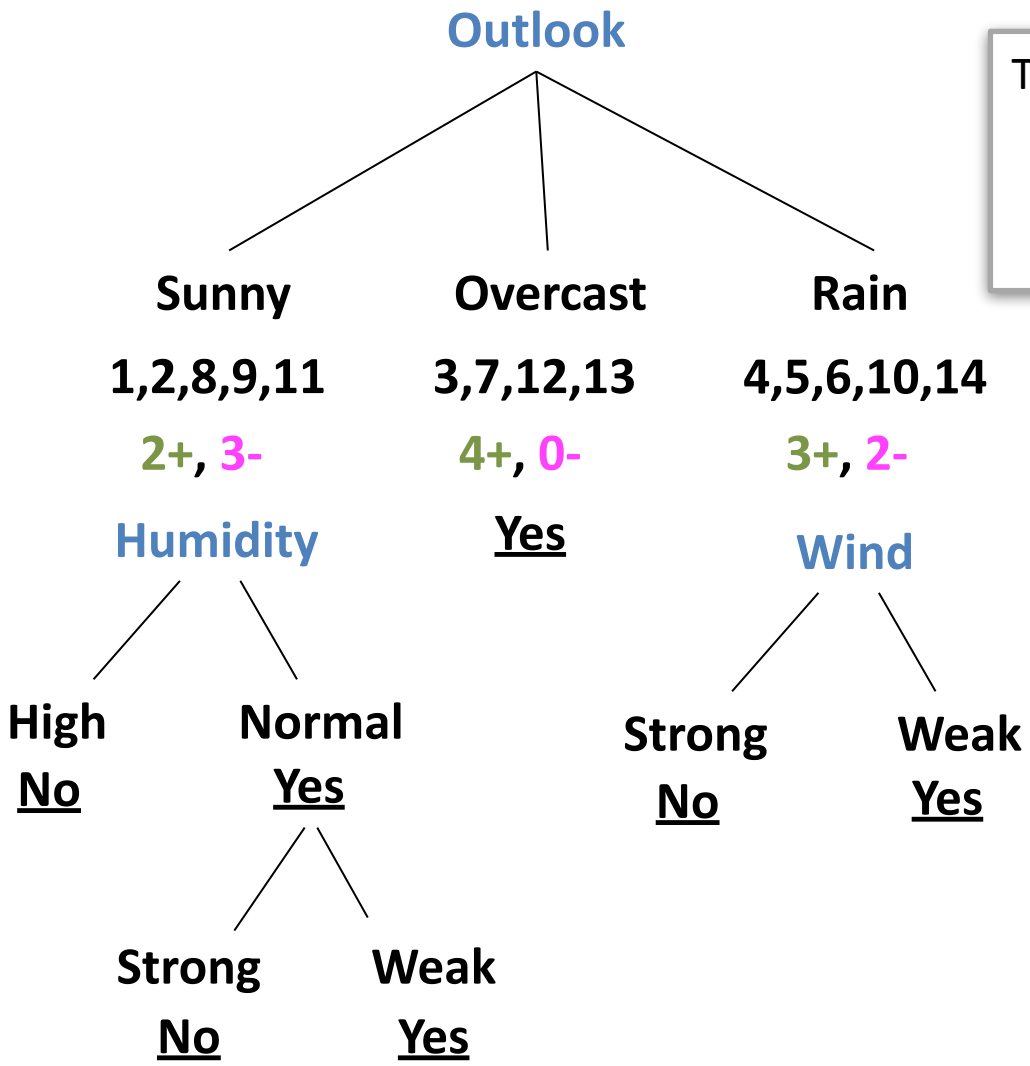
Example

Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, **No**



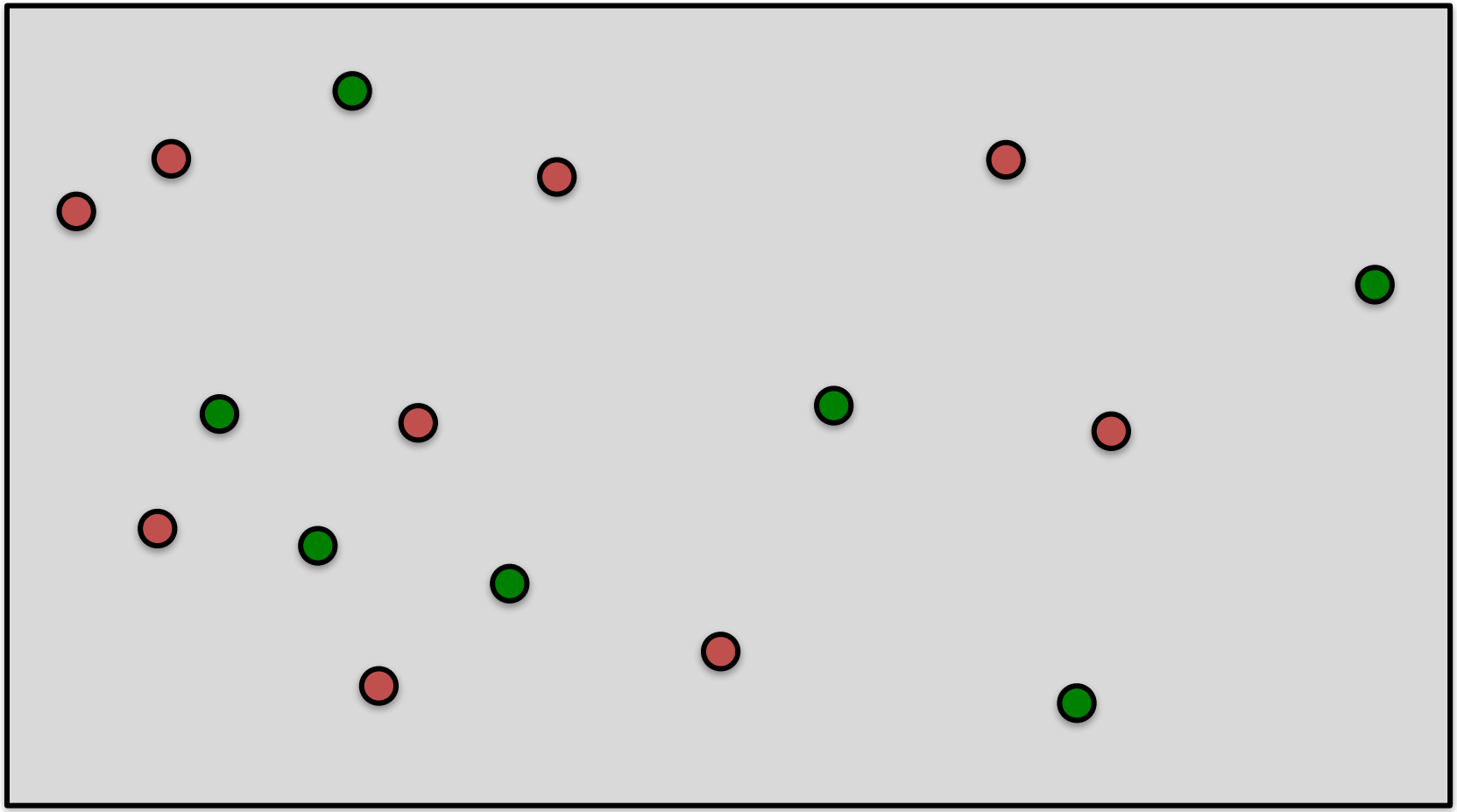
Example

Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, **No**

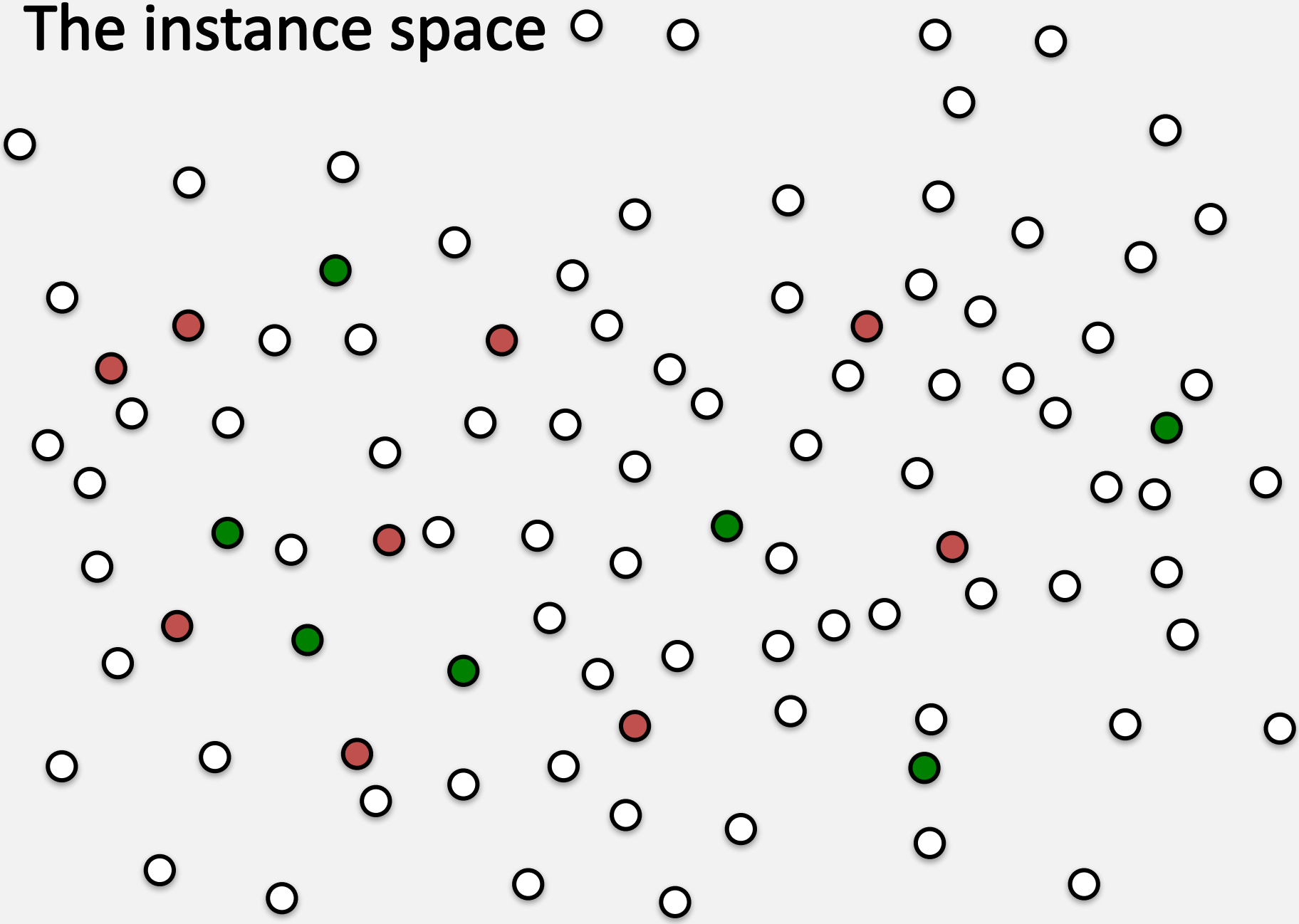


This can always be done but may fit noise or other coincidental regularities

Our training data



The instance space



Overfitting the data

Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance

Overfitting the data

Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance

- There may be noise in the training data the tree is fitting

Overfitting the data

Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance

- There may be noise in the training data the tree is fitting
- The algorithm might be making decisions based on very little data

Overfitting the data

Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance

- There may be noise in the training data the tree is fitting
- The algorithm might be making decisions based on very little data

A hypothesis h is said to overfit the training data if there is another hypothesis h', such that

- h has a smaller error than h' on the **training data** but
- h has larger error on the **test data** than h'

Reasons for overfitting

Too much variance in the training data

- Training data is not a representative sample of the instance space
- We split on features that are actually irrelevant

Reasons for overfitting

Too much variance in the training data

- Training data is not a representative sample of the instance space
- We split on features that are actually irrelevant

Too much noise in the training data

- Noise = some feature values or class labels are incorrect
- We learn to predict the noise

Reasons for overfitting

Too much variance in the training data

- Training data is not a representative sample of the instance space
- We split on features that are actually irrelevant

Too much noise in the training data

- Noise = some feature values or class labels are incorrect
- We learn to predict the noise

We'll talk about this again (bias vs. variance trade-off)

Avoiding overfitting with decision trees

Occam's Razor

Favor simpler hypotheses

- less likely to have low error by coincidence

For hypothesis class, need to define what is a simpler function

- For decision trees, simpler means smaller/shorter trees

Avoiding overfitting with decision trees

Approach 1. Fix the depth of the tree

- Decision stump = a decision tree with only one level
- Typically will not be very good by itself but short decision trees can make very good features for a second layer of learning (ensemble learning)

Avoiding overfitting with decision trees

Approach 1. Fix the depth of the tree

- Decision stump = a decision tree with only one level
- Typically will not be very good by itself but short decision trees can make very good features for a second layer of learning (ensemble learning)

Depth is a hyper-parameter: parameter we need to figure out how to set before training starts

Avoiding overfitting with decision trees

Approach 2. Optimize on a held-out set (also called development set or validation set) while growing the trees

- Split data into two parts: training set and held-out set
- Grow tree on training split and check performance on held-out set after every new node is added
- If growing tree hurts validation set performance, stop growing

Avoiding overfitting with decision trees

Approach 2. Optimize on a held-out set (also called development set or validation set) while growing the trees

- Split data into two parts: training set and held-out set
- Grow tree on training split and check performance on held-out set after every new node is added
- If growing tree hurts validation set performance, stop growing

Depth	Validation error
1	10%
2	5%
3	3%
35	17%

Avoiding overfitting with decision trees

Approach 2. Optimize on a held-out set (also called development set or validation set) while growing the trees

- Split data into two parts: training set and held-out set
- Grow tree on training split and check performance on held-out set after every new node is added
- If growing tree hurts validation set performance, stop growing

Depth	Validation error
1	10%
2	5%
3	3%
35	17%

Best depth!

Now go train your decision tree to depth 3 on all training data

Avoiding overfitting with decision trees

Approach 2. Optimize on a held-out set (also called development set or validation set) while growing the trees

- Split data into two parts: training set and held-out set
- Grow tree on training split and check performance on held-out set after every new node is added
- If growing tree hurts validation set performance, stop growing

Depth	Validation error
1	10%
2	5%
3	3%
35	17%

Problem: what if we just got a weird split between train/validation split? Is there anything we can do?

Avoiding overfitting with decision trees

Approach 2. Optimize on a held-out set (also called development set or validation set) while growing the trees

- Split data into two parts: training set and held-out set
- Grow tree on training split and check performance on held-out set after every new node is added
- If growing tree hurts validation set performance, stop growing

Depth	Validation error
1	10%
2	5%
3	3%
35	17%

Problem: what if we just got a weird split between train/validation split? Is there anything we can do?
Cross-validation: do many splits and average results

Avoiding overfitting with decision trees

Approach 3. Grow full tree and then prune as a post-processing step in one of several ways

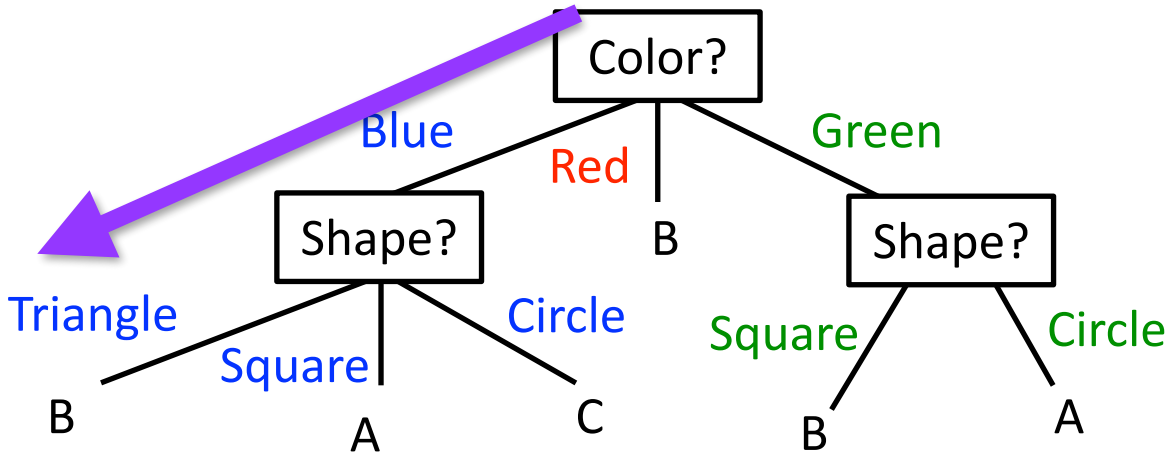
- Use a validation set for pruning from bottom up greedily
- Convert the tree into a set of rules (one rule per path from root to leaf and prune each rule independently)

Avoiding overfitting with decision trees

Approach 3. Grow full tree and then prune as a post-processing step in one of several ways

- Use a validation set for pruning from bottom up greedily
- Convert the tree into a set of rules (one rule per path from root to leaf and prune each rule independently)

Reminder: decision trees are rules



Every path from the tree to a root is a rule

The full tree is equivalent to the conjunction of all the rules

- Color=Blue AND Shape=Triangle then output Label=B
- Color=Blue AND Shape=Square then output Label=A
- Color=Blue AND Shape=Circle then output Label=C
- …

One way to prune: delete some of the rules

Avoiding overfitting with decision trees

Approach 3. Grow full tree and then prune as a post-processing step in one of several ways

- Use a validation set for pruning from bottom up greedily
- Convert the tree into a set of rules (one rule per path from root to leaf and prune each rule independently)

Pruning: remove leaves and assign majority label of parent to all items

Prune children of S if:

- All children are leaves,

and

- Accuracy on validation set does not decrease if we assign the most frequent class label to all items

Summary: avoiding overfitting

Two basic approaches

- **Pre-pruning**: Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.
- **Post-pruning**: Grow the full tree and then remove nodes that seem not to have sufficient evidence.

Summary: avoiding overfitting

Two basic approaches

- **Pre-pruning**: Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.
- **Post-pruning**: Grow the full tree and then remove nodes that seem not to have sufficient evidence.

Methods for evaluating subtrees to prune

- **Cross-validation**: Reserve hold-out set to evaluate utility
- **Statistical testing**: Test if the observed regularity can be dismissed as likely to occur by chance
- **Minimum Description Length**: Is the additional complexity of the hypothesis smaller than remembering the exceptions?

Summary: avoiding overfitting

Two basic approaches

- **Pre-pruning**: Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.
- **Post-pruning**: Grow the full tree and then remove nodes that seem not to have sufficient evidence.

Methods for evaluating subtrees to prune

- **Cross-validation**: Reserve hold-out set to evaluate utility
- **Statistical testing**: Test if the observed regularity can be dismissed as likely to occur by chance
- **Minimum Description Length**: Is the additional complexity of the hypothesis smaller than remembering the exceptions?

We will talk more about generalization, overfitting, cross-validation

Summary: decision trees

Popular machine learning tool

- Prediction is easy
- If we have Boolean features and binary classification, decision trees can represent any Boolean function

Greedy heuristics for learning

- ID3 algorithm (using information gain)
- Robust implementations of some variants (e.g., C4.5 algorithm) exist

(Can be used for regression too: leaf is no longer a class label but a function that produces some number)

Decision trees are prone to overfitting unless you take care to avoid

Evaluation

CROSS-VALIDATION

Model selection

Very broadly: choosing the best model using given data

What makes a model:

1. Features
2. Hyper-parameters that control the hypothesis space
    - Example: depth of a decision tree, neural network architecture, etc.
3. The learning algorithm, which may have its own hyperparameters
4. Actual model itself

The learning algorithms we see in this class only find the last one

- What about the rest?

Model selection strategies

Choose model that performs best on held-out validation data

Cross-validation

- estimate model performance using resampling technique

Cross-validation

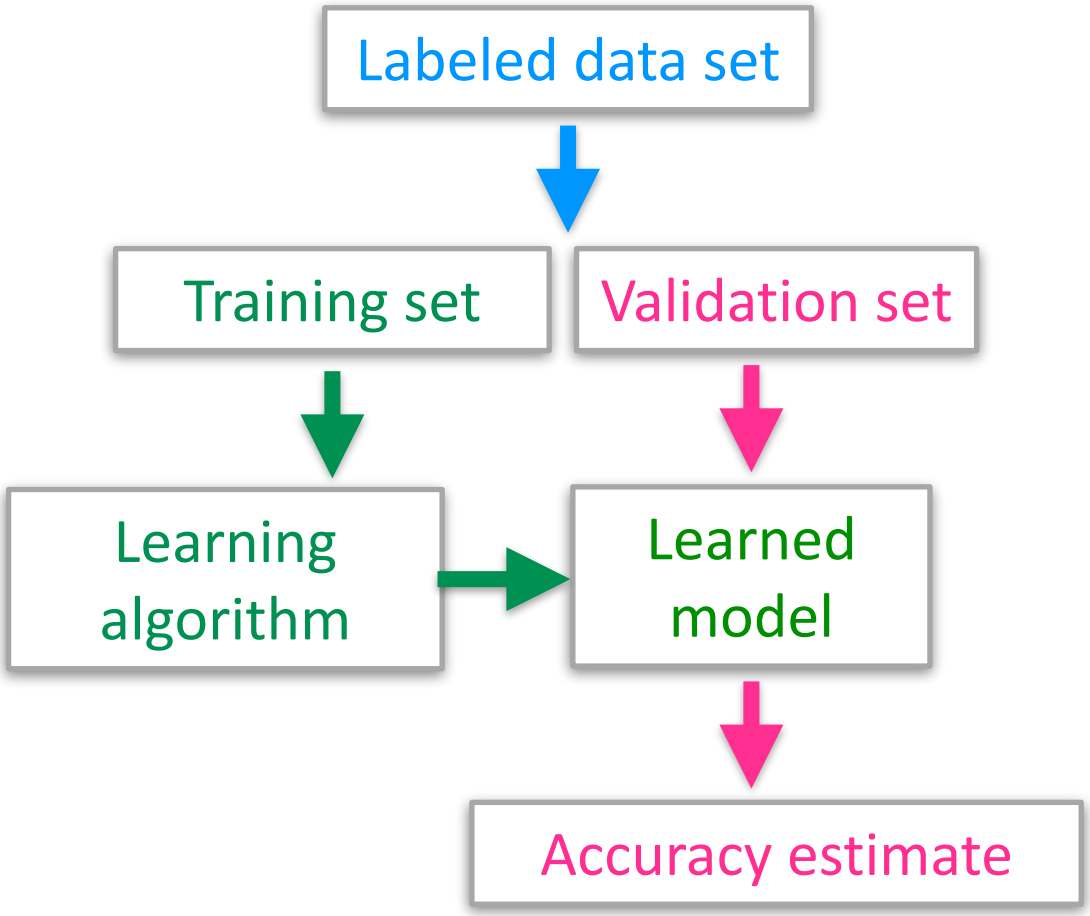
We want to train a classifier using a given dataset

We know how to train given features and hyper-parameters

How do we know what the best feature set and hyper-parameters are?

Held-out/validation sets

How can we get an unbiased estimate of the accuracy of the learned model?



Validation sets

How can we get an unbiased estimate of the accuracy of the learned model?

- When learning a model, you should pretend that you don't have the actual test data yet
- If the test set labels influence the learned model in any way, accuracy estimates will be biased.

Validation sets

How can we get an unbiased estimate of the accuracy of the learned model?

- When learning a model, you should pretend that you don't have the actual test data yet
- If the test set labels influence the learned model in any way, accuracy estimates will be biased.

Only once you are happy with your learned model, may you use it on test data and report test performance

K-fold cross-validation

Given a particular feature set and hyper-parameter setting

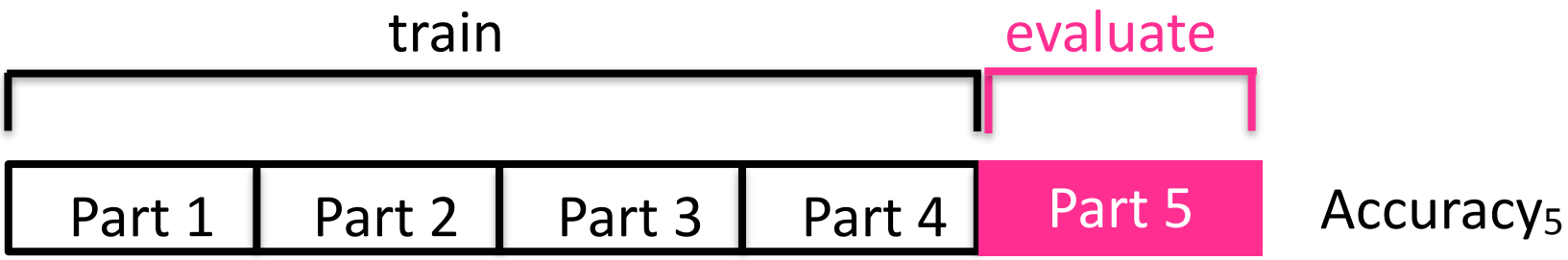
1. Split the data into K (say 5 or 10) equal sized parts

K-fold cross-validation

Given a particular feature set and hyper-parameter setting

1. Split the data into K (say 5 or 10) equal sized parts

2. Train a classifier on four parts and evaluate it on the fifth one



K-fold cross-validation

Given a particular feature set and hyper-parameter setting

1. Split the data into K (say 5 or 10) equal sized parts

2. Train a classifier on four parts and evaluate it on the fifth one

3. Repeat this using each of the K parts as the validation set

Part 1	Part 2	Part 3	Part 4	Part 5	$Accuracy_5$
Part 1	Part 2	Part 3	Part 4	Part 5	$Accuracy_4$
Part 1	Part 2	Part 3	Part 4	Part 5	$Accuracy_3$
Part 1	Part 2	Part 3	Part 4	Part 5	$Accuracy_2$
Part 1	Part 2	Part 3	Part 4	Part 5	$Accuracy_1$

K-fold cross-validation

Given a particular feature set and hyper-parameter setting

1. Split the data into K (say 5 or 10) equal sized parts

2. Train a classifier on four parts and evaluate it on the fifth one

3. Repeat this using each of the K parts as the validation set

4. The quality of this feature set/hyper-parameter is the average of these K estimates

$$\text{Performance} = (\text{Accuracy}_1 + \text{Accuracy}_2 + \text{Accuracy}_3 + \text{Accuracy}_4 + \text{Accuracy}_5) / 5$$

K-fold cross-validation

Given a particular feature set and hyper-parameter setting

1. Split the data into K (say 5 or 10) equal sized parts

2. Train a classifier on four parts and evaluate it on the fifth one

3. Repeat this using each of the K parts as the validation set

4. The quality of this feature set/hyper-parameter is the average of these K estimates

$$\text{Performance} = (\text{Accuracy}_1 + \text{Accuracy}_2 + \text{Accuracy}_3 + \text{Accuracy}_4 + \text{Accuracy}_5) / 5$$

5. Repeat for every feature set/hyper-parameter choice

Cross-validation

We want to train a classifier using a given dataset

We know how to train given features and hyper-parameters

How do we know what the best feature set and hyper-parameters are?

Cross-validation

We want to train a classifier using a given dataset

We know how to train given features and hyper-parameters

How do we know what the best feature set and hyper-parameters are?

1. Evaluate every feature set and hyper-parameter using cross-validation (could be computationally expensive)
2. Pick the best according to cross-validation performance
3. Train on full data using this setting

To avoid cross-validation pitfalls, ask:

1. Is my held-aside test data really representative of going out to collect new data?

Problems:

- Even if your methodology is fine, someone may have collected features for positive examples differently than for negatives – should be randomized
- Example: samples from cancer processed by different people or on different days than samples for normal controls

To avoid cross-validation pitfalls, ask:

2. Did I repeat my entire data processing procedure on every fold of cross-validation, using only the training data for that fold?

Problems:

- On each fold of cross-validation, did I ever access in any way the label of a test case?
- Any preprocessing done over entire data set (feature selection, parameter tuning, threshold selection) must not use labels

To avoid cross-validation pitfalls, ask:

3. On each fold of cross-validation, did I ever access in any way the label of a test case? Have I modified my algorithm so many times, or tried so many approaches, on this same data set that I (the human) am overfitting it?

Problems:

- Have I continually modified my preprocessing or learning algorithm until I got some improvement on this data set?
- If so, I really need to get some additional data now to at least test on

Leave-one-out cross-validation

Special case where $k = n$ (i.e., all training data examples)

1. Train using $n - 1$ examples, validate on remain example
2. Repeat n times, each with different validation example
3. Finally, choose model with smallest average validation error

Leave-one-out cross-validation

Special case where $k = n$ (i.e., all training data examples)

1. Train using $n - 1$ examples, validate on remain example
2. Repeat n times, each with different validation example
3. Finally, choose model with smallest average validation error

When is it used?

- Can be expensive for large n, so typically used when n is small
- Useful in domains with limited training data (maximizes data used for training)

Summary: cross-validation

Cross-validation generates an approximate estimate of how well the classifier will do on ``unseen'' data

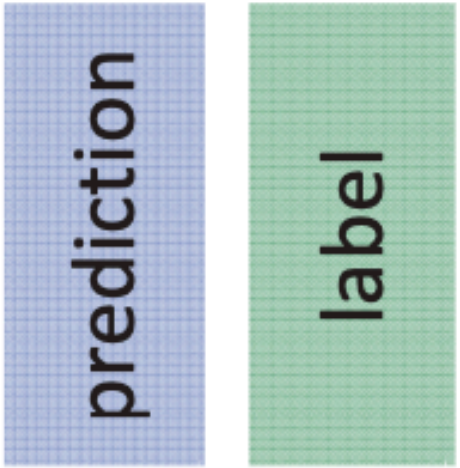
- as $k \rightarrow n$, model becomes more accurate (more training data)
- ... but, cross-validation becomes more computationally expensive (have to train k models)
- choosing $k < n$ is a compromise

Evaluation

EVALUATION METRICS

Evaluation metrics

To evaluate model, compare predicted labels to actual

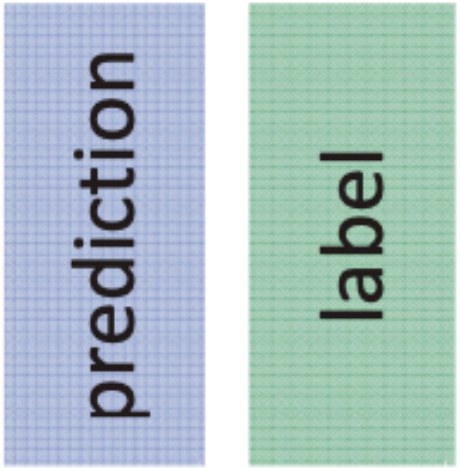


Accuracy: proportion of examples where we predicted correct label

$$\text{accuracy} = \frac{\#\text{ correct predictions}}{\#\text{ test instances}}$$

Evaluation metrics

To evaluate model, compare predicted labels to actual



Accuracy: proportion of examples where we predicted correct label

$$\text{accuracy} = \frac{\#\text{ correct predictions}}{\#\text{ test instances}}$$

Error: proportion of examples where we predicted incorrect label

$$\text{error} = 1 - \text{accuracy}$$

$$= \frac{\#\text{ incorrect predictions}}{\#\text{ test instances}}$$

Problems

What if the number of observations for one class is different than the number of observations for another class?

What if you have more than 2 classes? How do you know whether all classes are being predicted equally well?

Confusion matrices

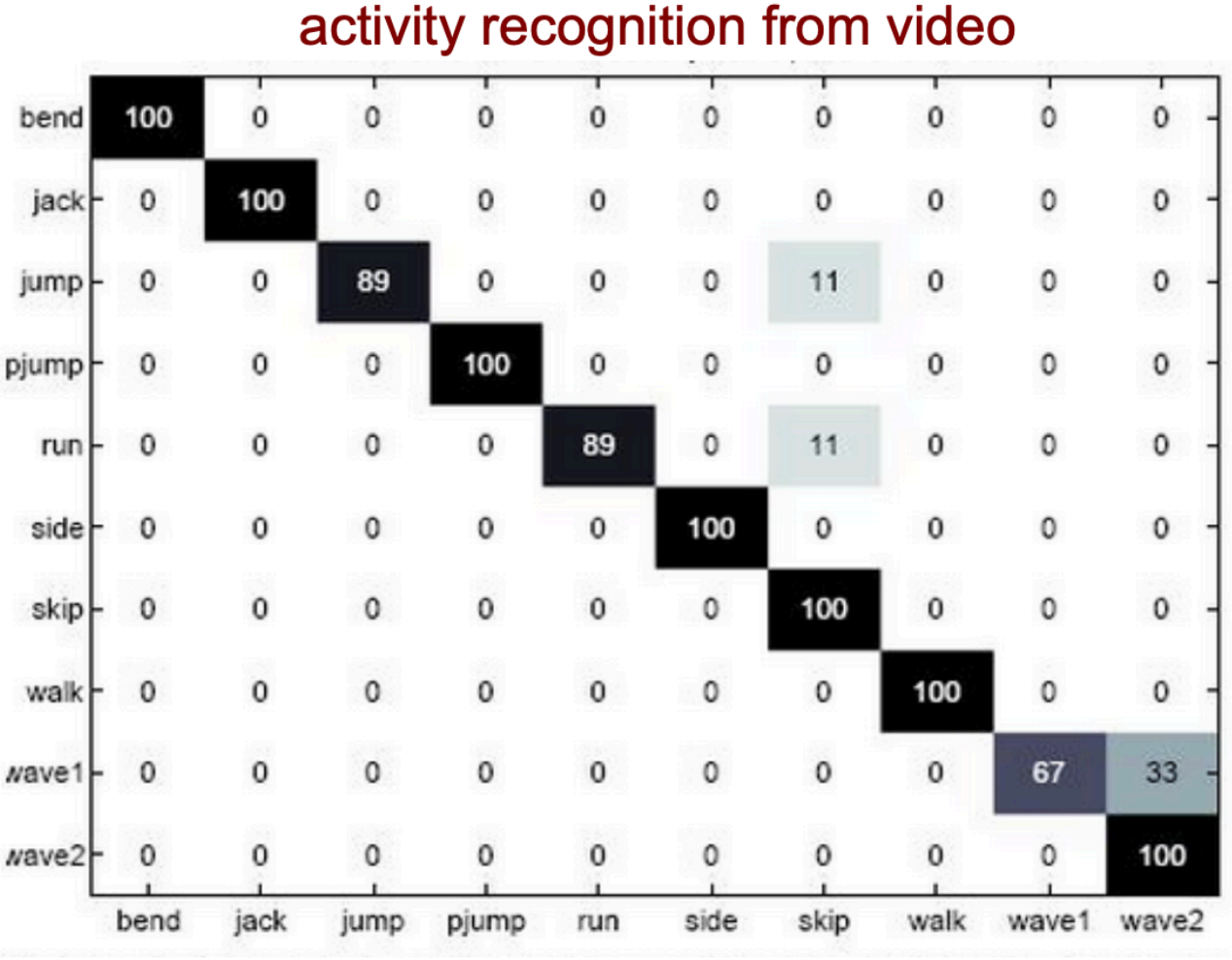
How can we understand what types of mistakes a learned model makes?

Confusion matrix

- Summarizes performance of a classification model
- Shows ways in which your model is confused (types of errors made) when model makes predictions

Confusion matrices

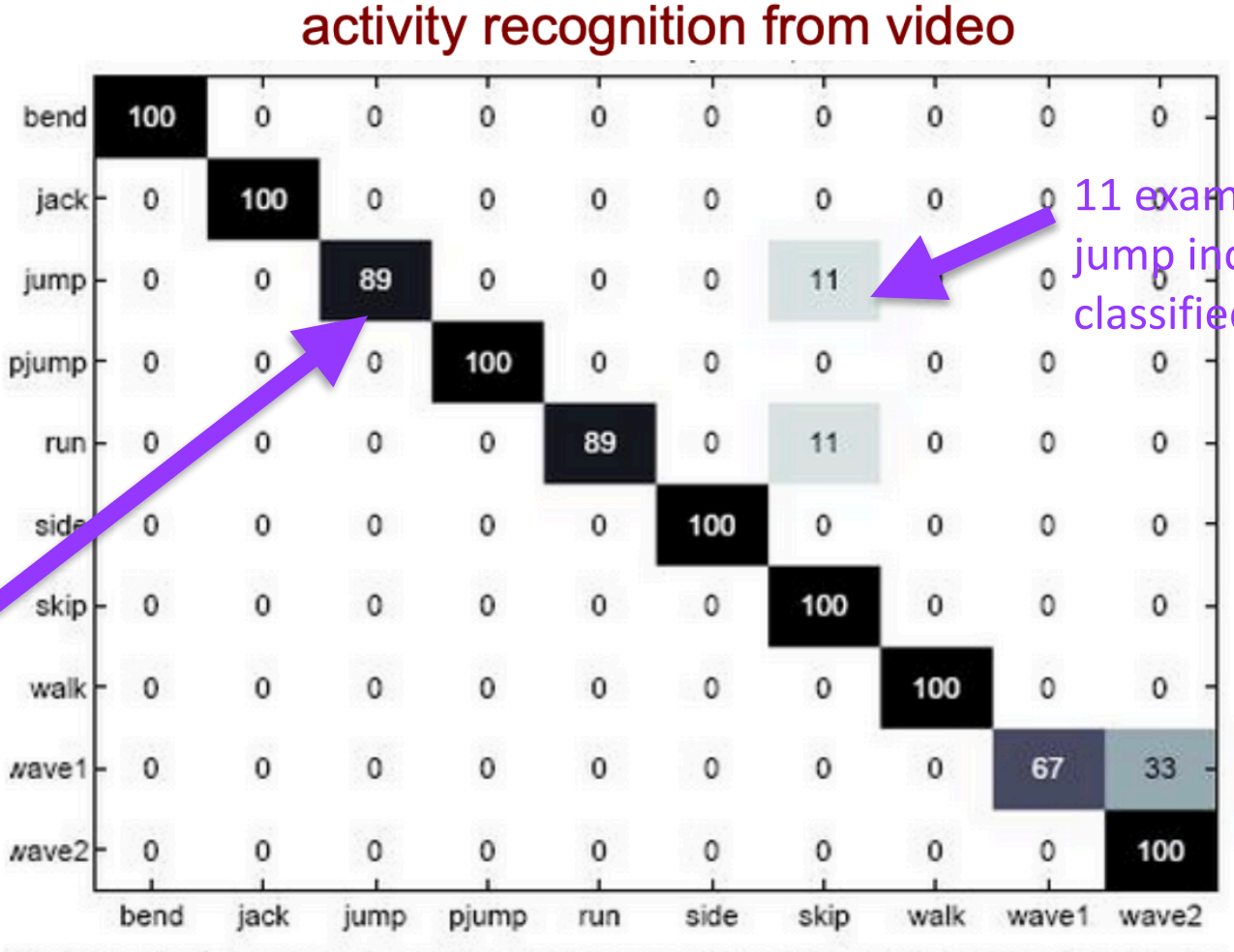
Table contains counts of correct and incorrect classifications



from http://vision.jhu.edu/

Confusion matrices

Table contains counts of correct and incorrect classifications



from http://vision.jhu.edu/

Confusion matrix for 2-class problems

Imagine a classifier that identifies presence of disease

		predicted class: Yes	predicted class: No
actual class	Yes	TP	FN
actual class	No	FP	TN

TP = person tests positive and really has disease

TN = person tests negative and really does not have disease

FP = person tests positive and does not have disease

FN = person tests negative and has disease

Confusion matrix for 2-class problems

Imagine a classifier that identifies presence of disease

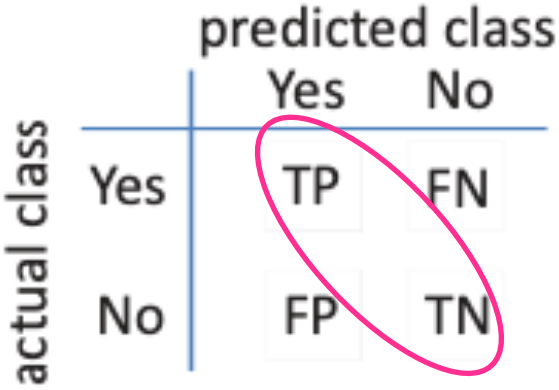
		predicted class: Yes	predicted class: No
actual class	Yes	TP	FN
actual class	No	FP	TN

TP = person tests positive and really has disease
TN = person tests negative and really does not have disease
FP = person tests positive and does not have disease
FN = person tests negative and has disease

How do we compute accuracy?

Confusion matrix for 2-class problems

Imagine a classifier that identifies presence of disease



$$accuracy = \frac{TP + TN}{P + N}$$

TP = person tests positive and really has disease
TN = person tests negative and really does not have disease
FP = person tests positive and does not have disease
FN = person tests negative and has disease

P = actual class is positive = TP + FN
N = actual class is negative = TN + FP

FN: has disease

FP: does not have disease

Is accuracy an adequate measure of performance?

Accuracy may not be useful measure in cases where …

There is a large class skew

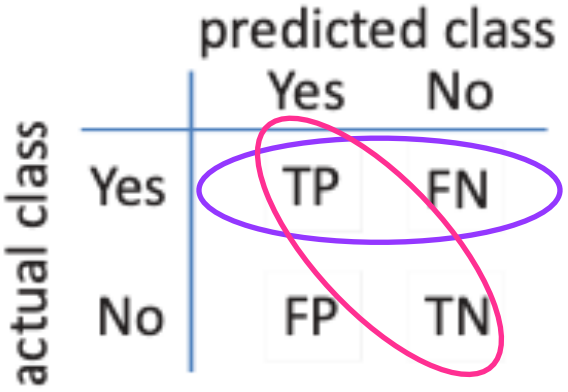
- Is 98% accuracy good if 97% of the instances are negative?
- If we just always guessed negative that would give us 97% accuracy!

There are differential misclassification costs – say, getting a positive wrong costs more than getting a negative wrong

- Consider a medical domain in which a false positive results in an extraneous test but a false negative results in a failure to treat a disease

Confusion matrix for 2-class problems

Given a dataset of P positive instance and N negative instances



$$accuracy = \frac{TP + TN}{P + N}$$

Imagine a classifier that identifies presence of disease

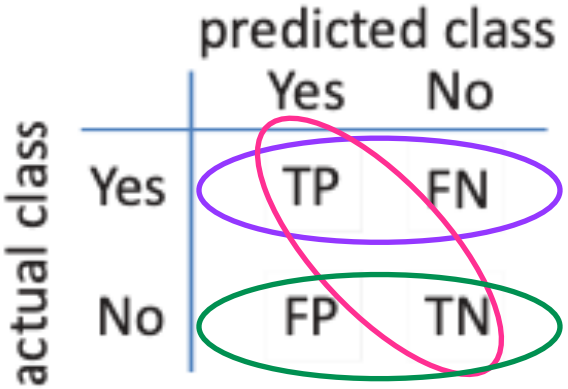
$$sensitivity = \frac{TP}{TP + FN}$$

(true positive rate) = probability of positive test given person has disease

How good is model at detecting positive cases?

Confusion matrix for 2-class problems

Given a dataset of P positive instance and N negative instances



$$accuracy = \frac{TP + TN}{P + N}$$

Imagine a classifier that identifies presence of disease

$$sensitivity = \frac{TP}{TP + FN}$$

(true positive rate) = probability of positive test given person has disease

How good is model at detecting positive cases?

$$specificity = \frac{TN}{TN + FP}$$

(true negative rate) = probability of negative test given person does not have disease

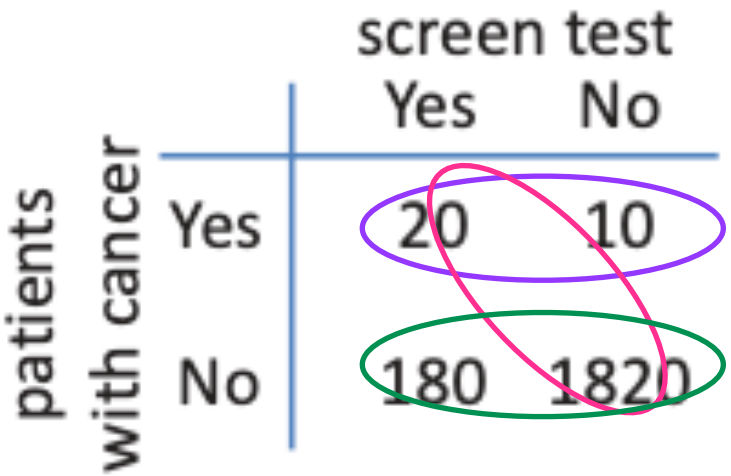
How good is model at detecting negative cases?

Confusion matrix: cancer dataset

patients with cancer	screen test: Yes	screen test: No
Yes	20	10
No	180	1820

Compute accuracy, sensitivity, and specificity

Confusion matrix: cancer dataset



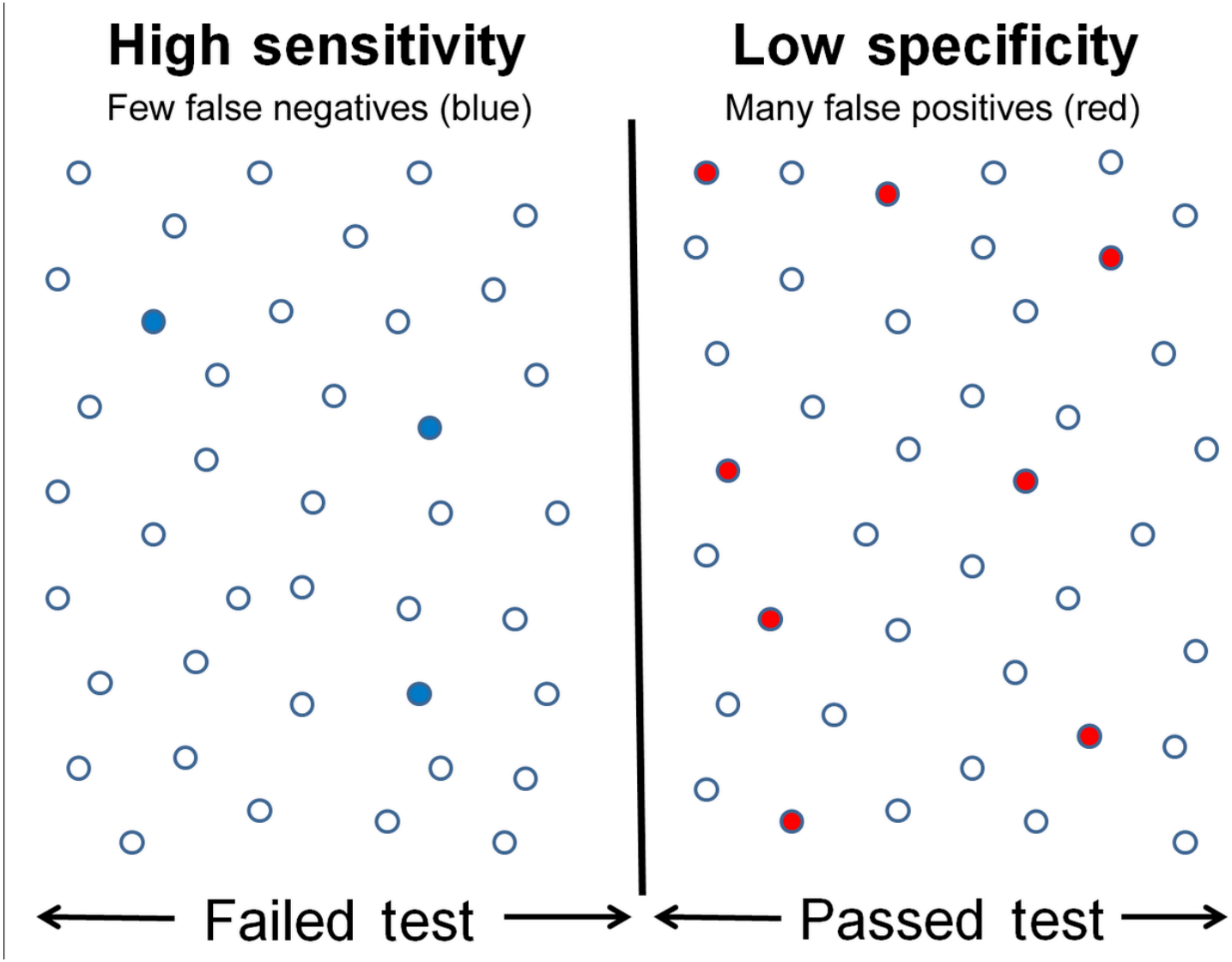
$$accuracy = \frac{TP + TN}{P + N}$$

$$sensitivity = \frac{TP}{TP + FN}$$

$$specificity = \frac{TN}{TN + FP}$$

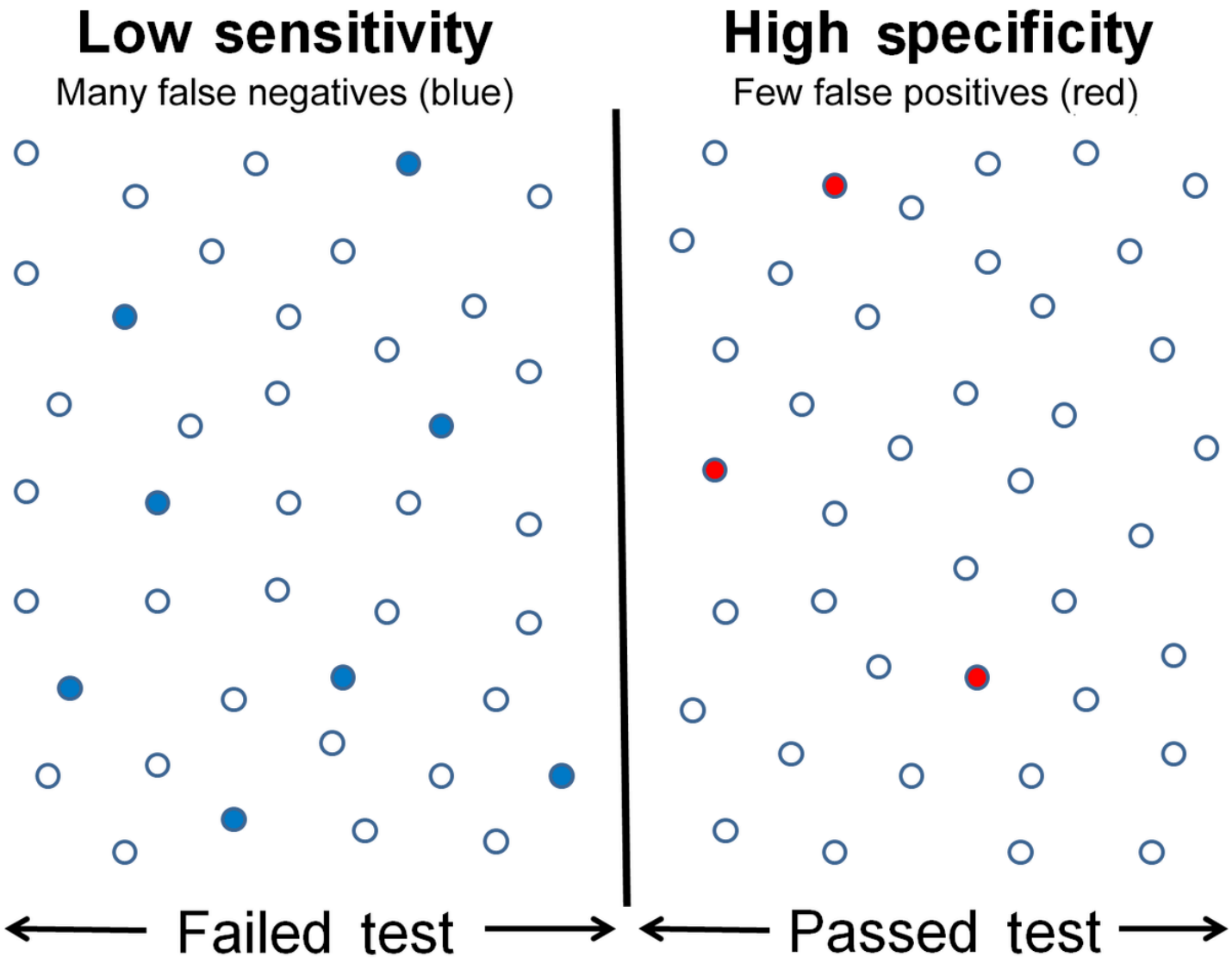
Compute accuracy, sensitivity, and specificity

Sensitivity vs. specificity



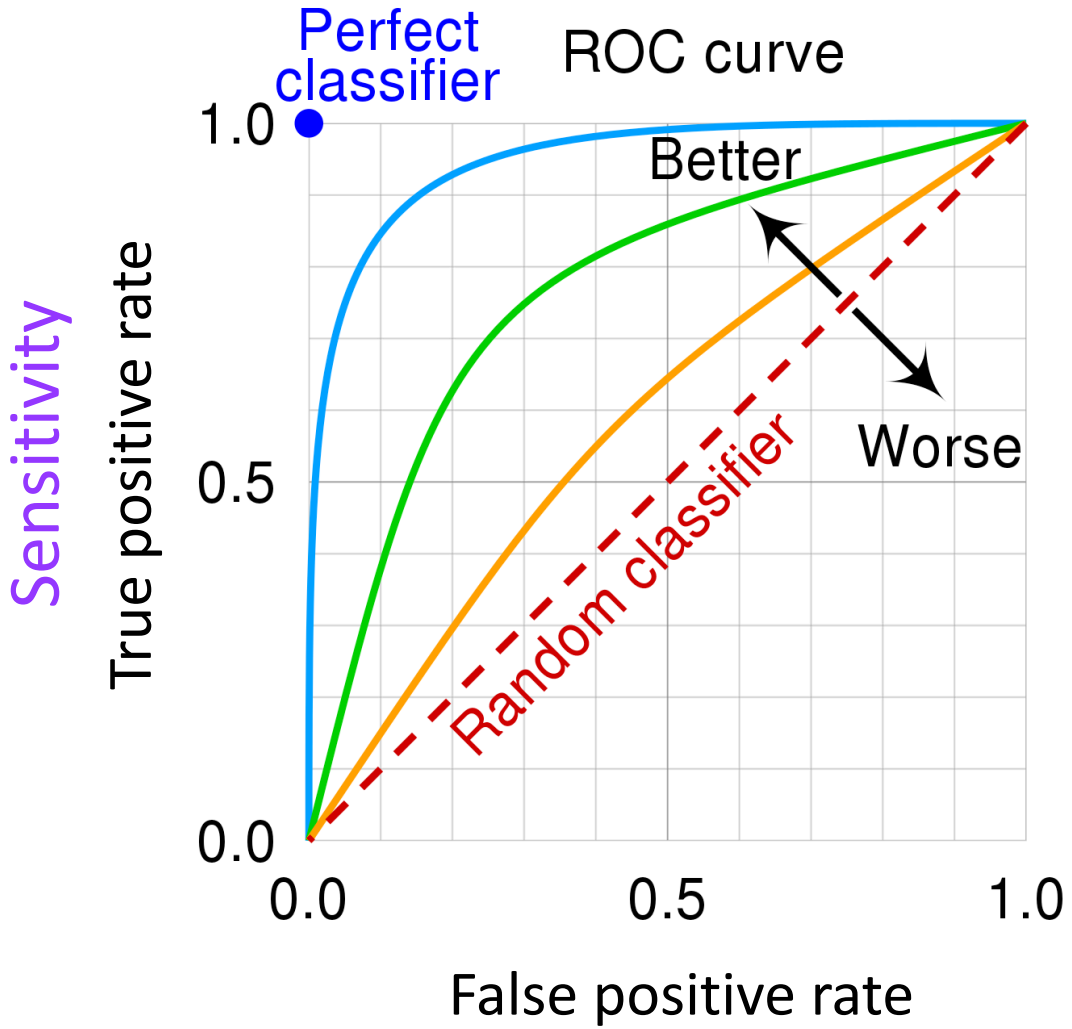
from https://en.wikipedia.org/wiki/Sensitivity_and_specificity

Sensitivity vs. specificity



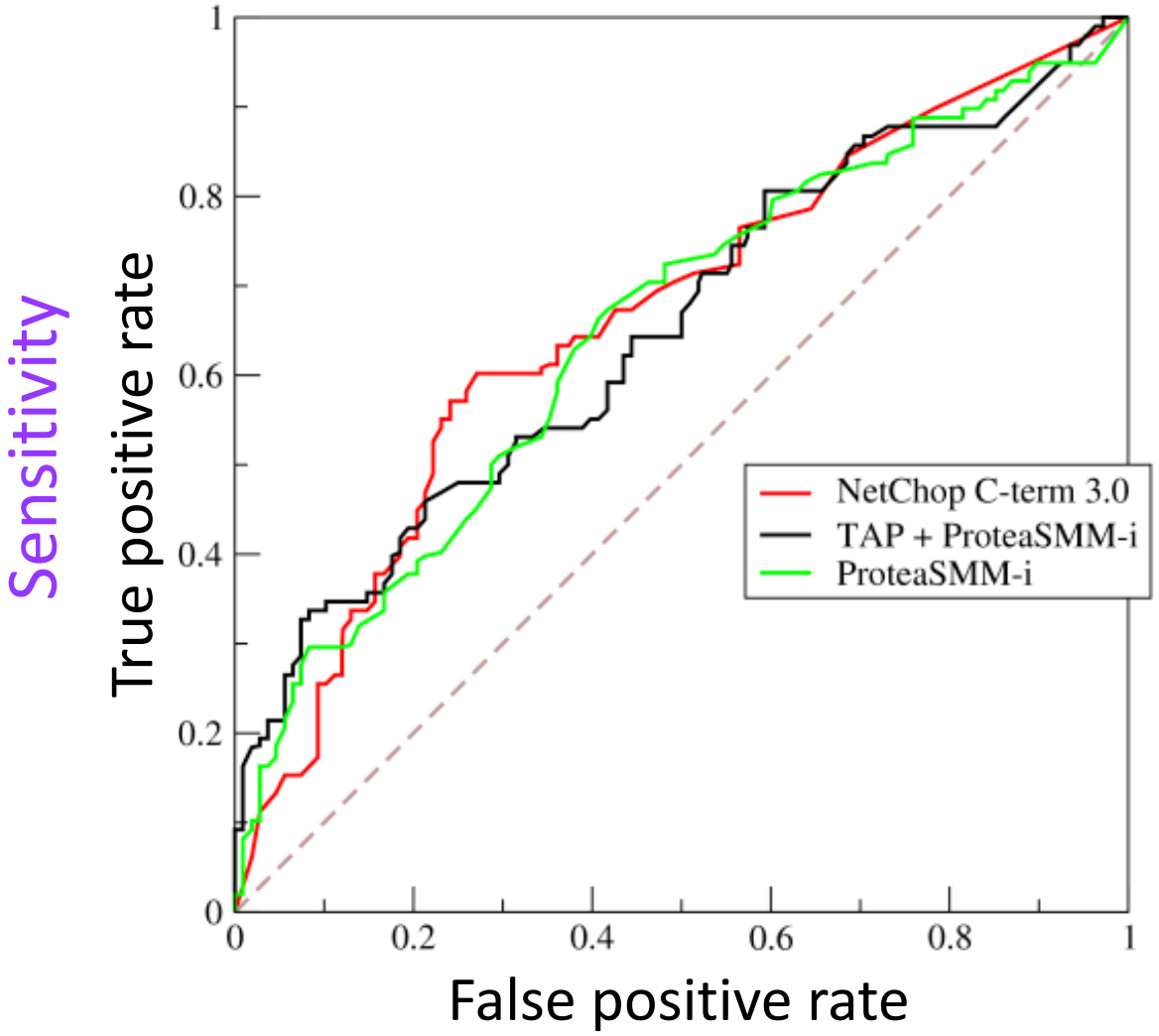
from https://en.wikipedia.org/wiki/Sensitivity_and_specificity

Receiver Operating Characteristic (ROC) curve



from https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Receiver Operating Characteristic (ROC) curve

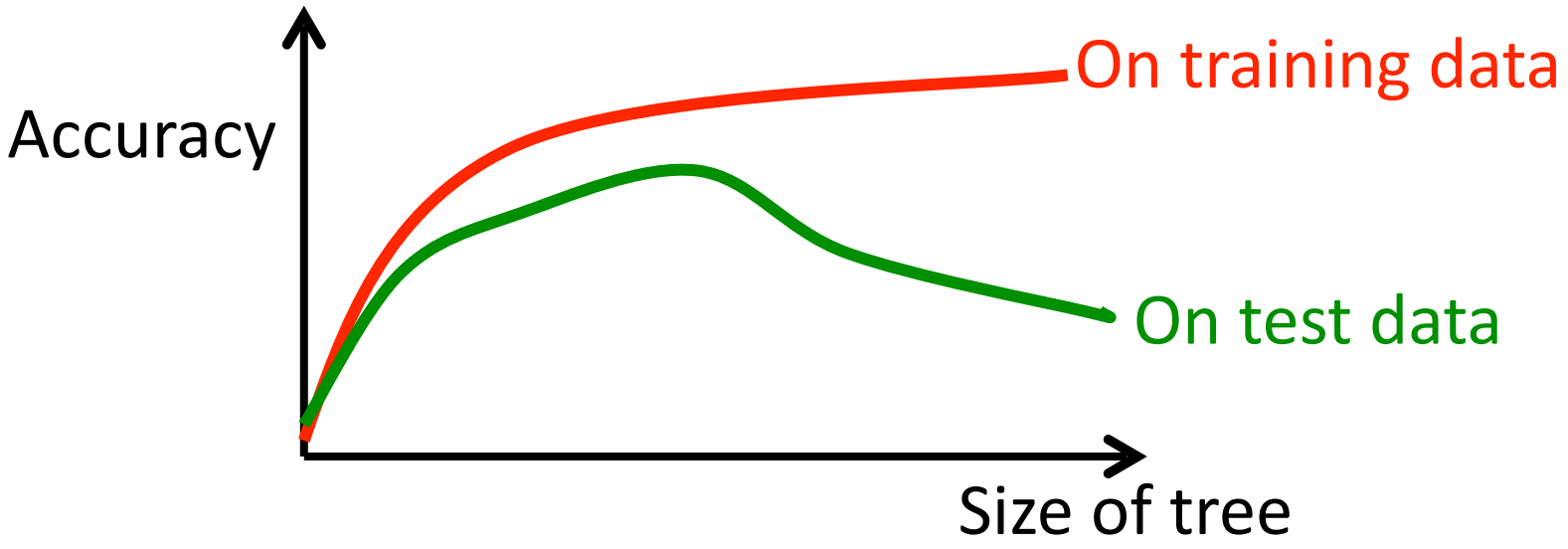


from https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Evaluation

BIAS AND VARIANCE

Overfitting the data



A decision tree overfits the training data when its accuracy on the training data goes up but its accuracy on unseen data goes down

Reasons for overfitting

Too much variance in the training data

- Training data is not a representative sample of the instance space
- We split on features that are actually irrelevant

Reasons for overfitting

Too much variance in the training data

- Training data is not a representative sample of the instance space
- We split on features that are actually irrelevant

Too much noise in the training data

- Noise = some feature values or class labels are incorrect
- We learn to predict the noise

Reasons for overfitting

Too much variance in the training data

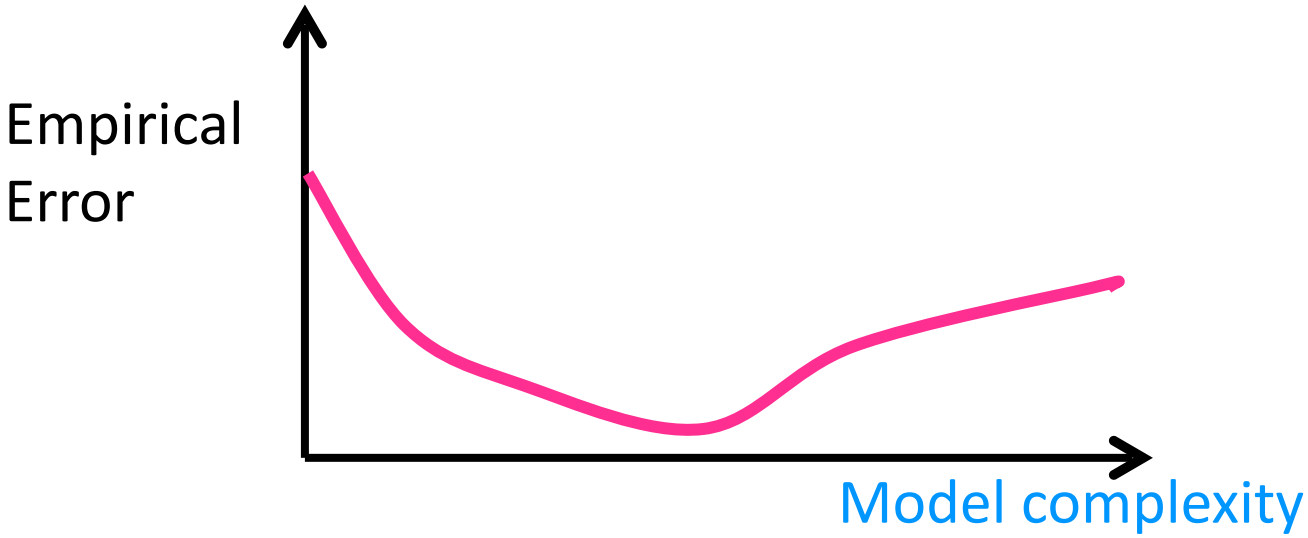
- Training data is not a representative sample of the instance space
- We split on features that are actually irrelevant

Too much noise in the training data

- Noise = some feature values or class labels are incorrect
- We learn to predict the noise

In both cases, it is a result of our will to minimize the empirical error when we learn, and the ability to do it (with DTs)

Model complexity

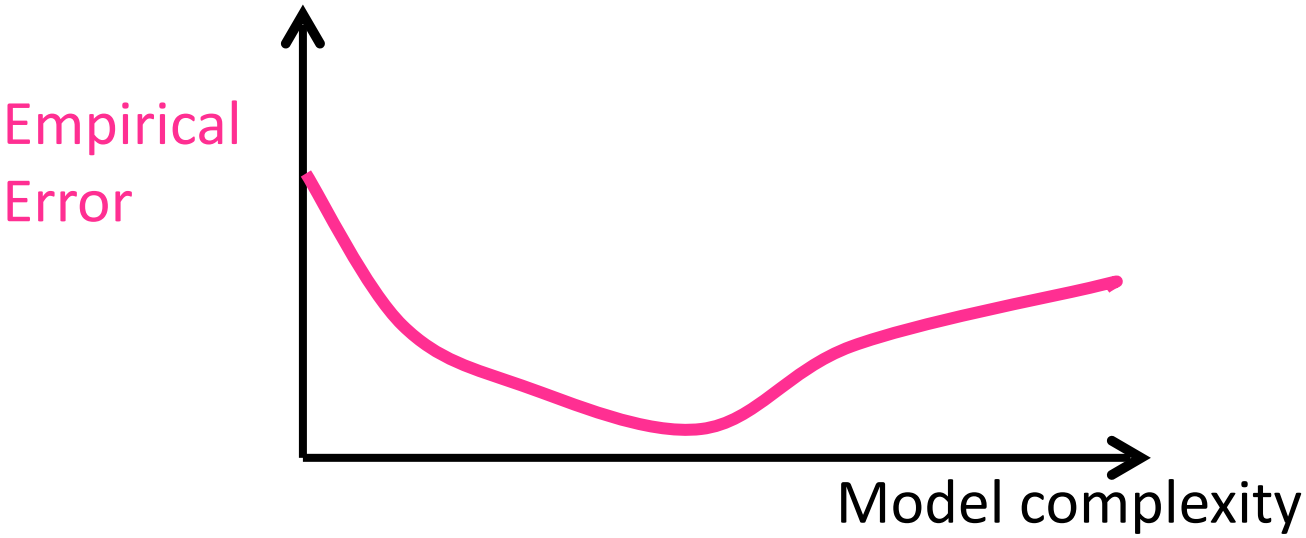


Model complexity (informally):

How many parameters do we have to learn?

Decision trees: complexity = # of nodes

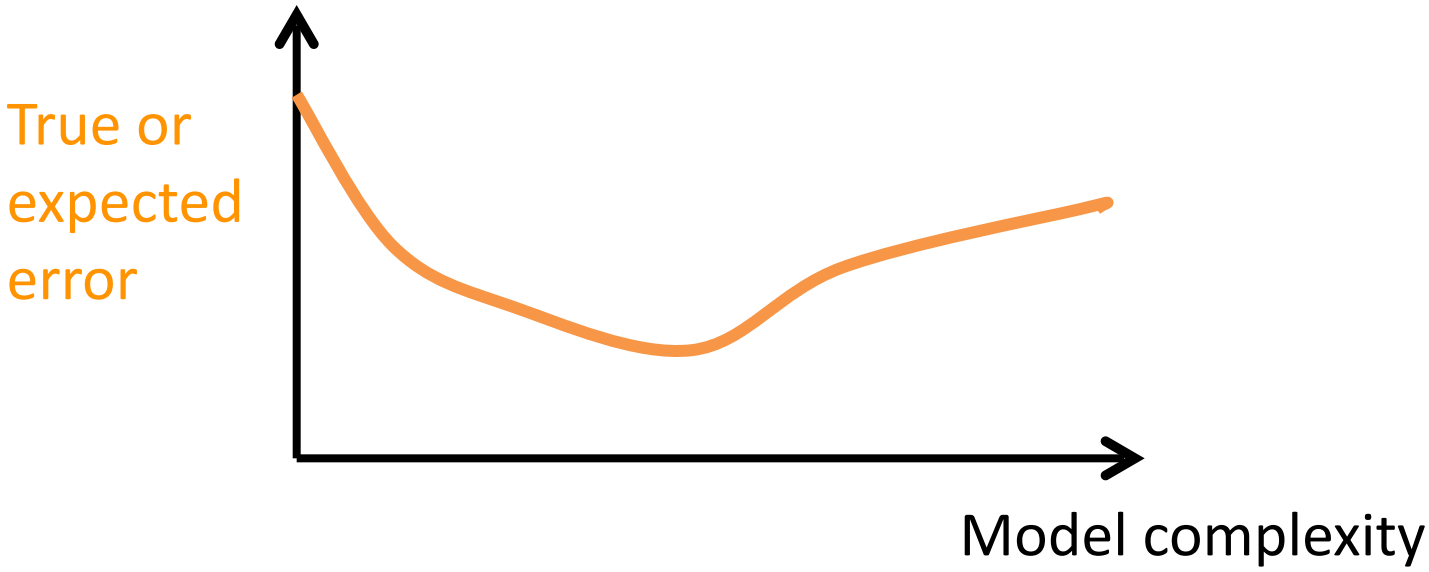
Empirical error



Empirical error (= on a given data set):

The percentage of items in this data set are misclassified by the classifier, aka hypothesis h.

True or expected error



True or expected error:

What percentage of items drawn from $D(X, Y)$ do we expect to be misclassified by hypothesis h?

(That's what we really care about – **generalization**)

What is bias?

Every learning algorithm requires assumptions about the hypothesis space

E.g., “my hypothesis space is

- linear”
- decision trees with 5 nodes”
- a three layer neural network with rectifier hidden units”

What is bias?

Bias is the true error (loss) of the best predictor in the hypothesis set

What will the bias be if the hypothesis set cannot represent the target function? (high or low?)

What is bias?

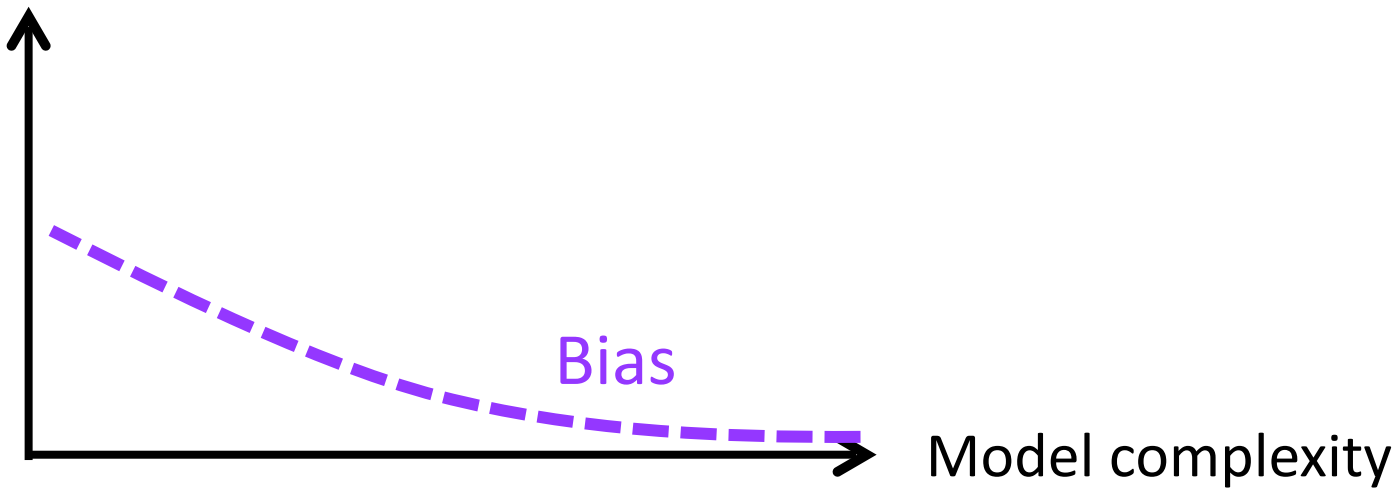
Bias is the true error (loss) of the best predictor in the hypothesis set

What will the bias be if the hypothesis set cannot represent the target function? (high or low?)

- Bias will be non-zero, possibly high

Underfitting: occurs when bias is high

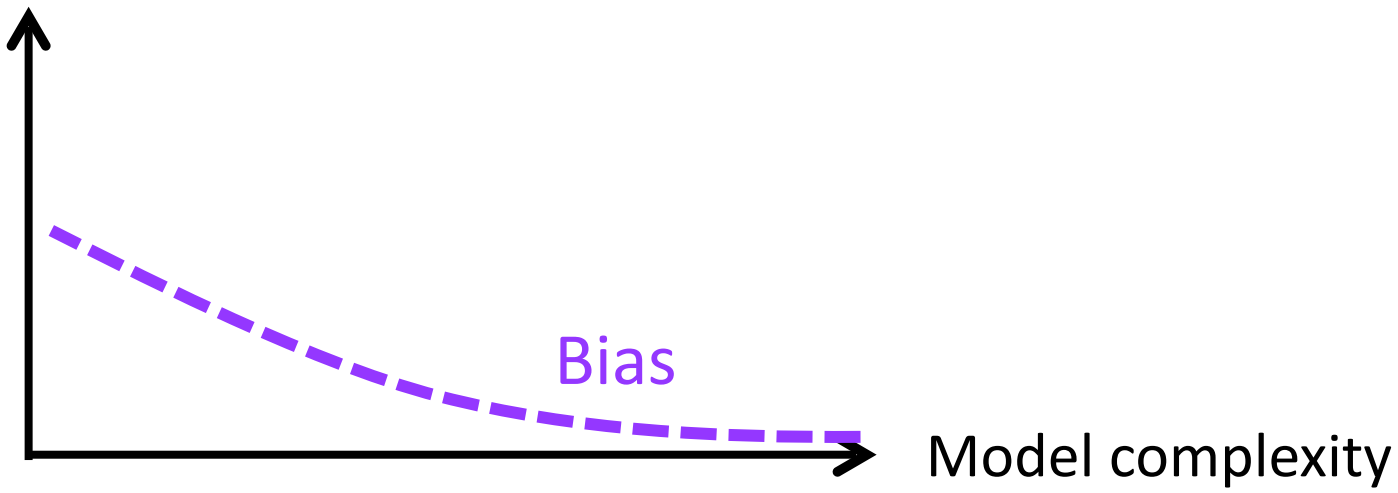
Bias of a learner (informally)



How likely is the learner to identify the **target** hypothesis?

- Bias is low when the model is expressive (low empirical error)
- Bias is high when the model is (too) simple
- The larger the hypothesis space, the easier it is to be close to the true hypothesis.

Bias of a learner (informally)



How likely is the learner to identify the **target** hypothesis?

- Bias is low when the model is expressive (low empirical error)
- Bias is high when the model is (too) simple
- The larger the hypothesis space, the easier it is to be close to the true hypothesis.
- More accurately: for each data set S, you learn a different hypothesis $h(S)$, that has a different true error $e(h)$; we are looking here at the difference of the mean of this random variable from the true error

What is variance?

The performance of a classifier is dependent on the specific training set we have. Perhaps model will change if we slightly change the training set

What is variance?

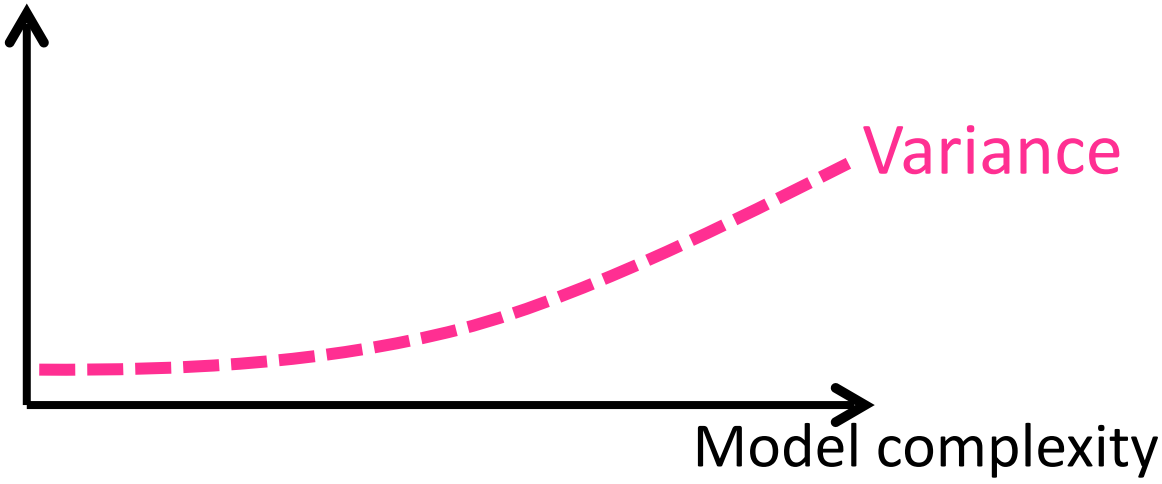
The performance of a classifier is dependent on the specific training set we have. Perhaps model will change if we slightly change the training set

Variance: describes how much the best classifier depends on the training set

- Variance increases when the classifiers become more complex
- Variance decreases with larger training sets

Overfitting: occurs when variance is high

Variance of a learner (informally)



How susceptible is the learner to minor changes in the training data?

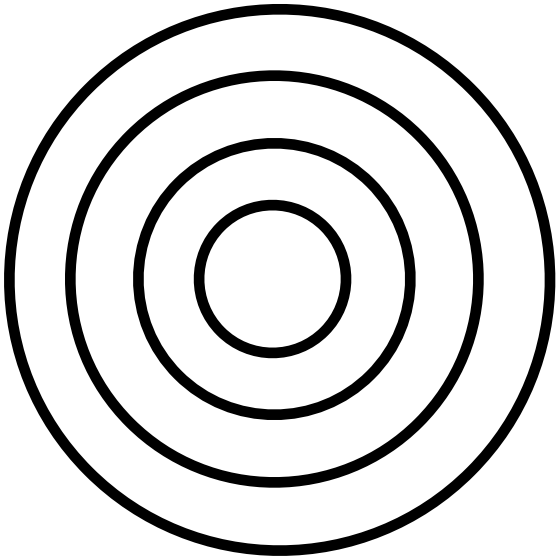
- i.e., to different samples from $D(X, Y)$

Variance increases with model complexity

- Think about extreme cases: hypothesis space with one function vs. all functions.
- Or, adding the "wind" feature in the decision tree earlier.
- The larger the hypothesis space, the more flexible the selection of the chosen hypothesis as a function of the data.
- More accurately: for each sample data set D, you will learn a different hypothesis $h(D)$, that will have a different sample error $e(h)$; we are looking here at the variance of this random variable from the true error.

Let's play darts

Suppose the true concept is the center



High bias

Hypothesis cannot represent target function

Low bias

Hypothesis can represent target function

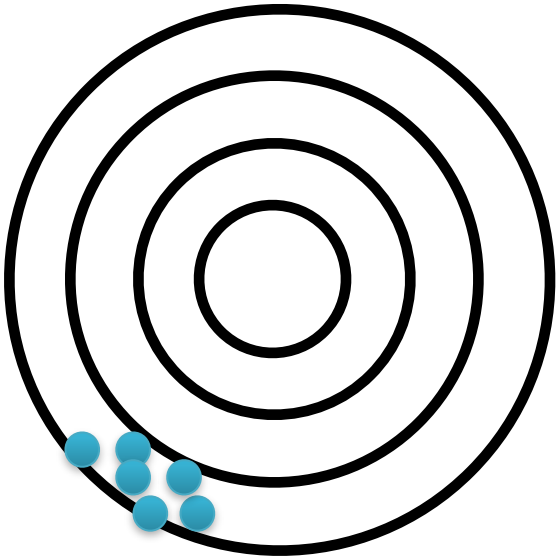
Low variance

High variance

How much does classifier depend on training set?

Let's play darts

Suppose the true concept is the center



High bias

Hypothesis cannot represent target function

Low bias

Hypothesis can represent target function

Each dot is a model that is learned from a a different dataset

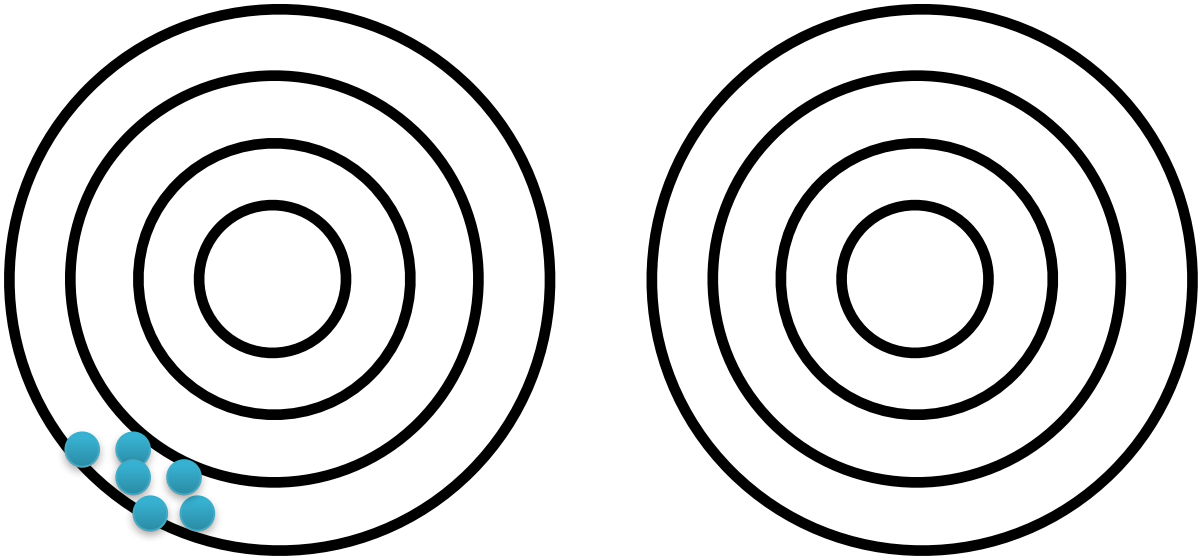
Low variance

High variance

How much does classifier depend on training set?

Let's play darts

Suppose the true concept is the center



High bias

Hypothesis cannot represent target function

Low bias

Hypothesis can represent target function

Each dot is a model that is learned from a a different dataset

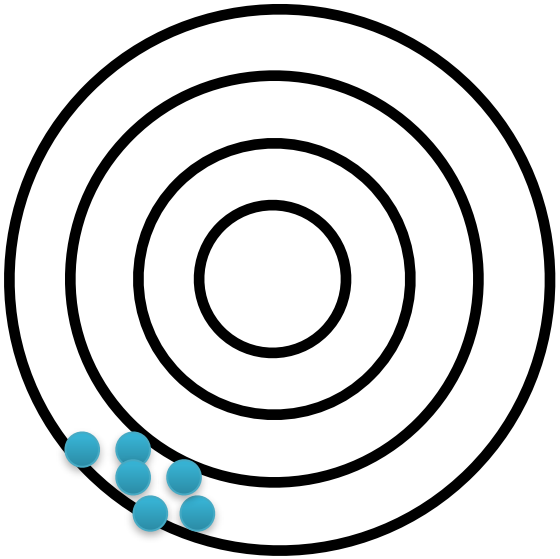
Low variance

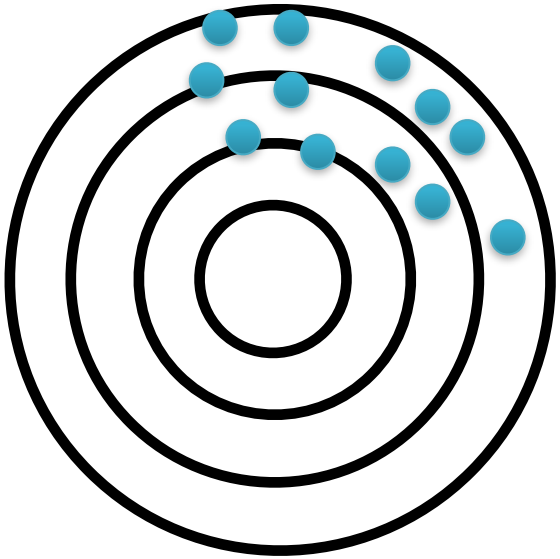
High variance

How much does classifier depend on training set?

Let's play darts

Suppose the true concept is the center





High bias

Hypothesis cannot represent target function

Low bias

Hypothesis can represent target function

Each dot is a model that is learned from a a different dataset

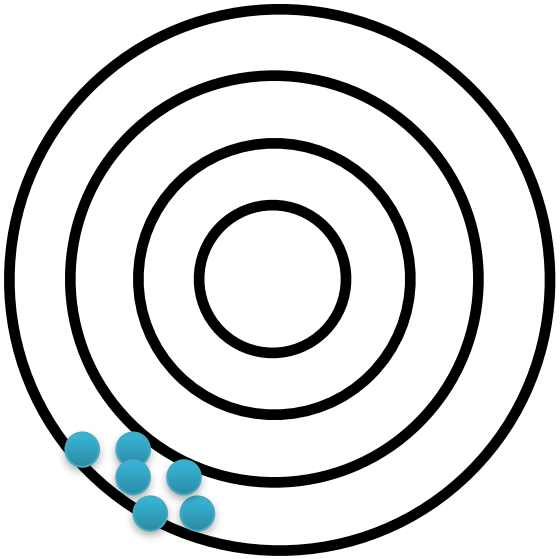
Low variance

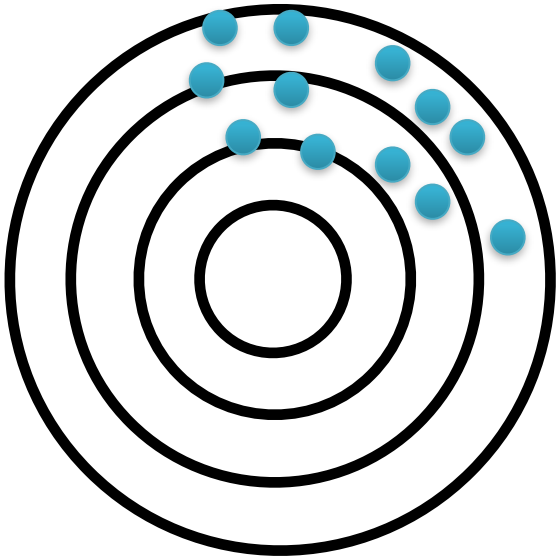
High variance

How much does classifier depend on training set?

Let's play darts

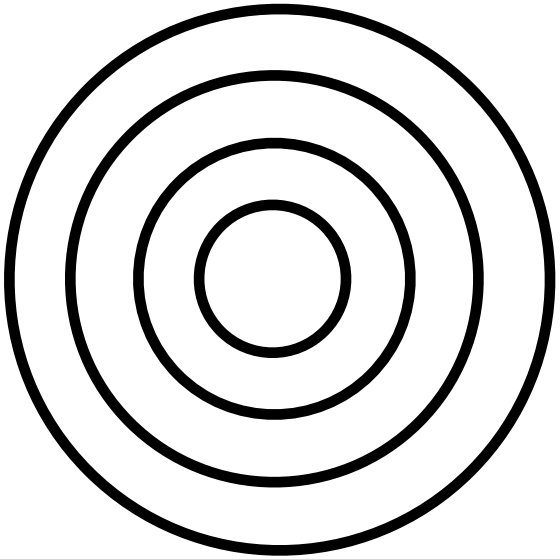
Suppose the true concept is the center





High bias

Hypothesis cannot represent target function



Low bias

Hypothesis can represent target function

Each dot is a model that is learned from a a different dataset

Low variance

High variance

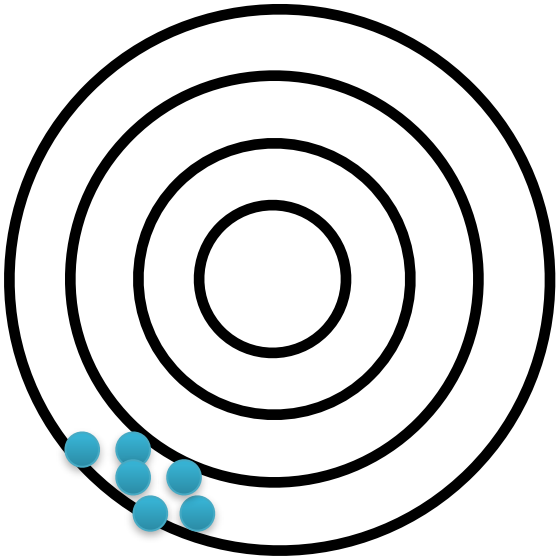
How much does classifier depend on training set?

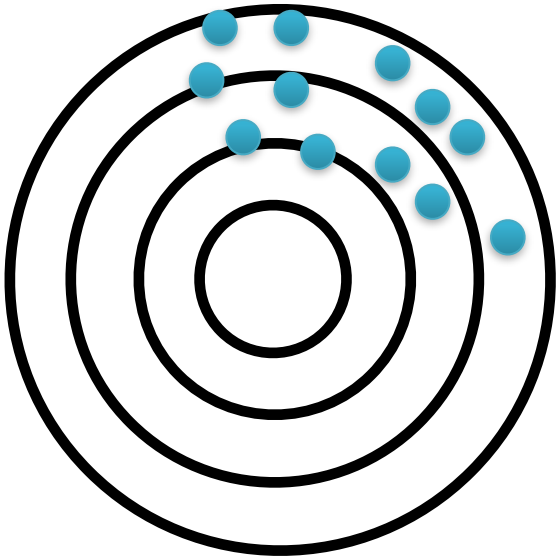
Let's play darts

Suppose the true concept is the center

High bias

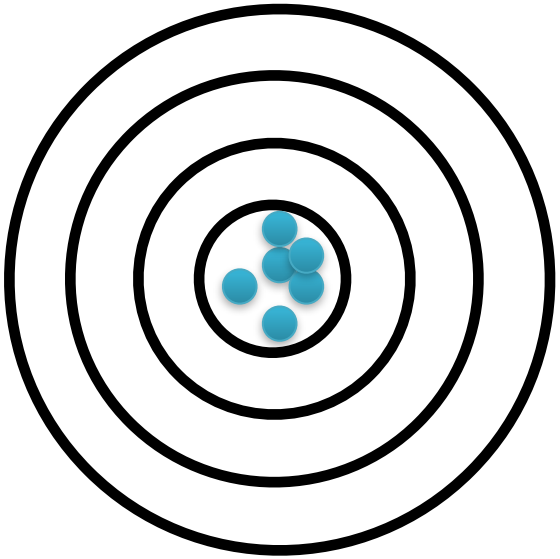
Hypothesis cannot represent target function





Low bias

Hypothesis can represent target function



Each dot is a model that is learned from a a different dataset

Low variance

High variance

How much does classifier depend on training set?

Let's play darts

Suppose the true concept is the center

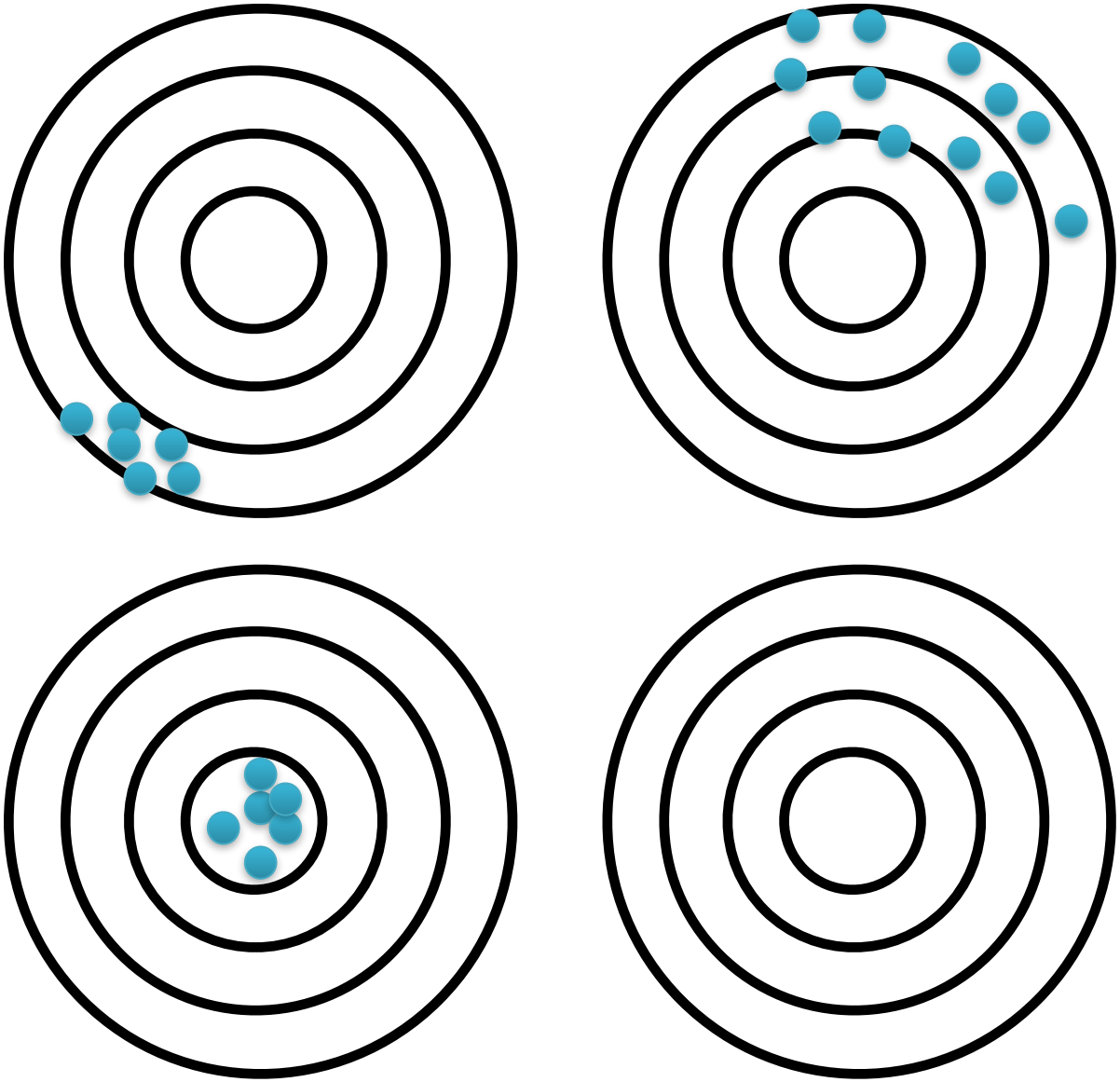
High bias

Hypothesis cannot represent target function

Low bias

Hypothesis can represent target function

Each dot is a model that is learned from a a different dataset



Low variance

High variance

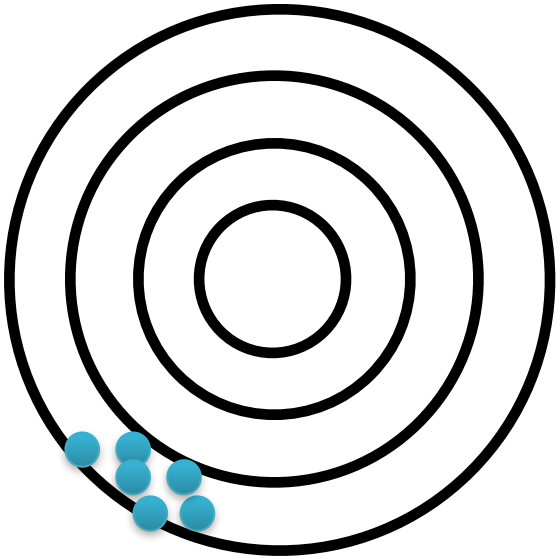
How much does classifier depend on training set?

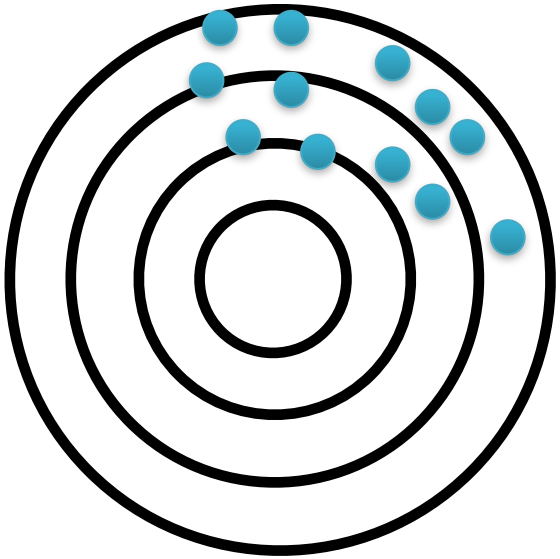
Let's play darts

Suppose the true concept is the center

High bias

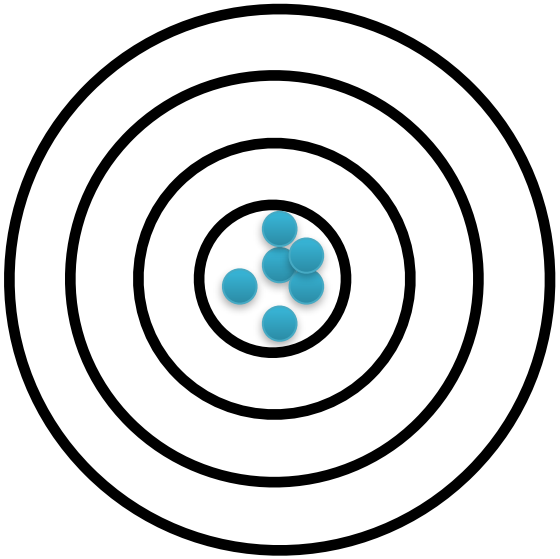
Hypothesis cannot represent target function

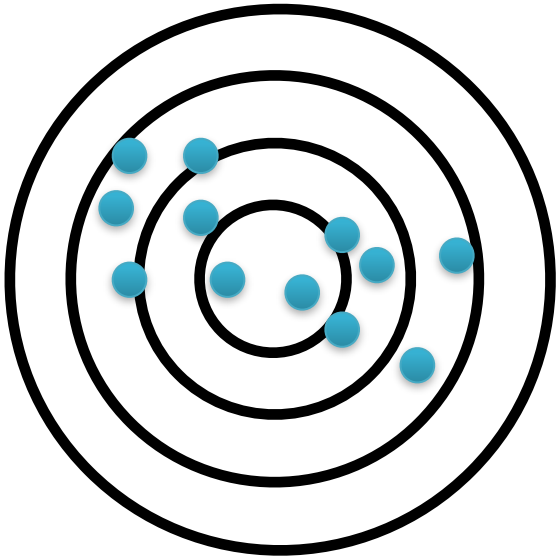




Low bias

Hypothesis can represent target function





Each dot is a model that is learned from a a different dataset

Low variance

High variance

How much does classifier depend on training set?

Bias variance tradeoffs

Error = bias + variance (+ noise)

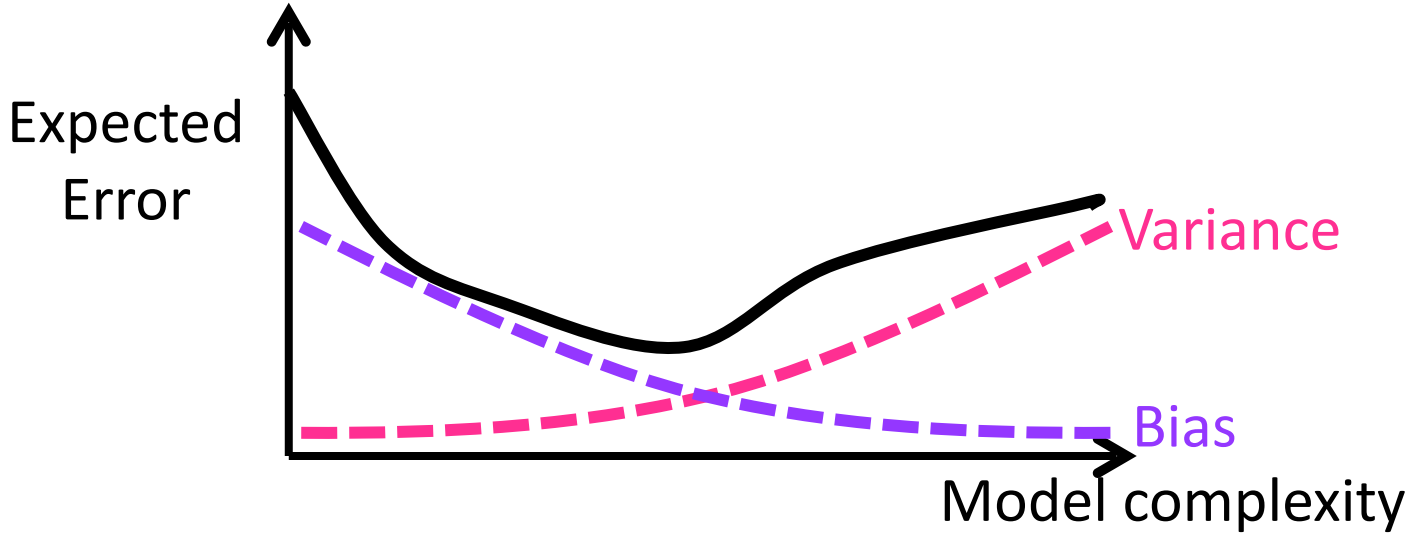
High bias → both training and test error can be high

- Arises when the classifier cannot represent the data

High variance → training error can be low, but test error will be high

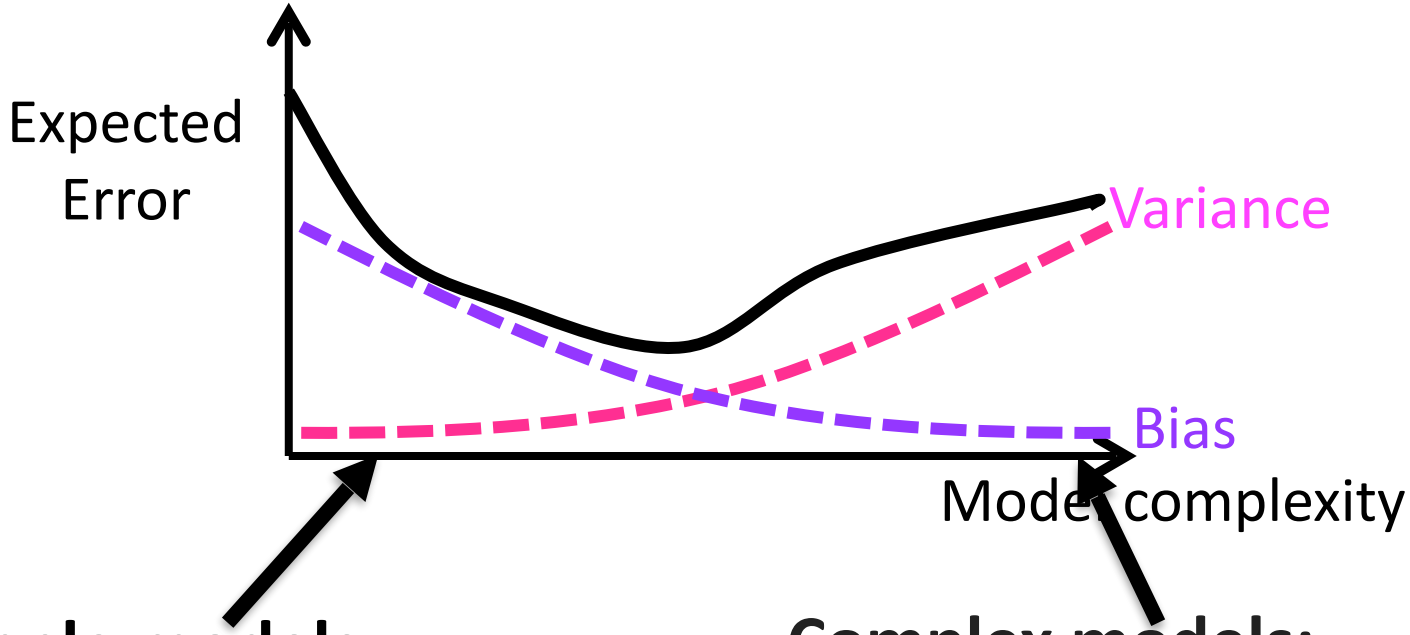
- Arises when the learner overfits the training set

Impact of bias and variance



Expected error ≈ bias + variance

Model complexity



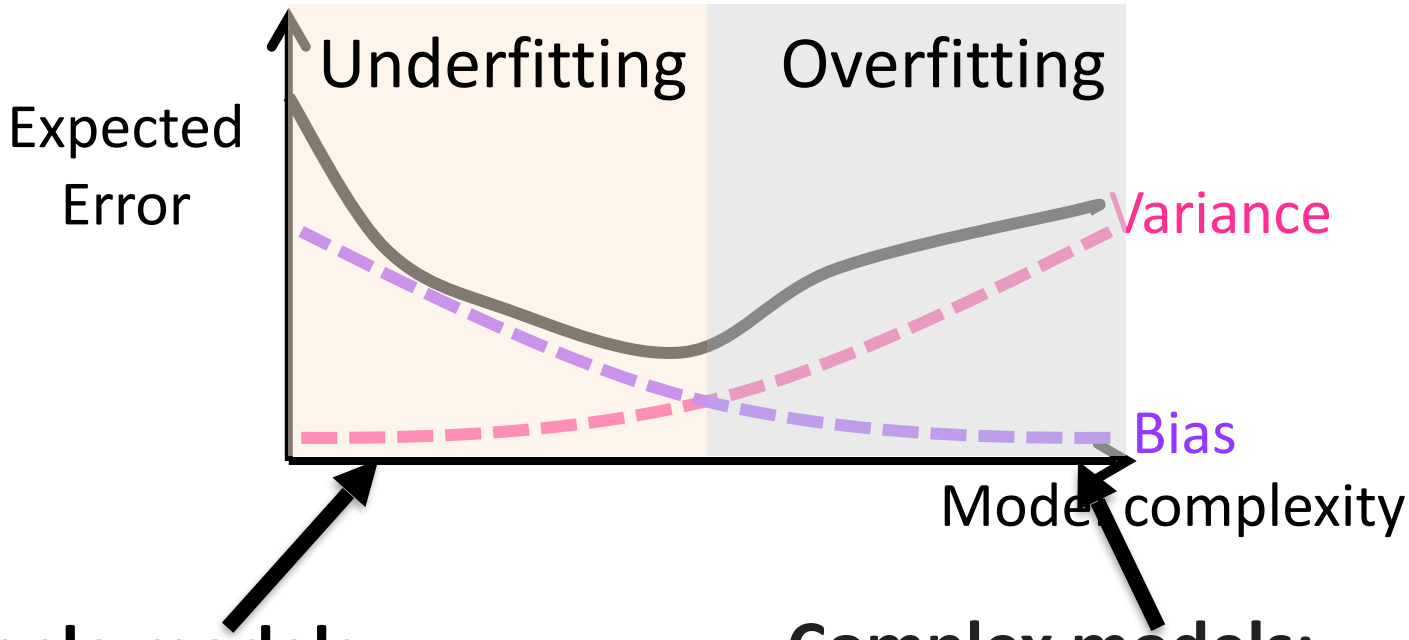
Simple models:

High bias and low variance

Complex models:

High variance and low bias

Model complexity



Simple models:

High bias and low variance

Complex models:

High variance and low bias

This can be made more accurate for some loss functions

Managing of bias and variance

Ensemble methods reduce variance

- Multiple classifiers are combined
- E.g., bagging, boosting

Decision trees of a given depth

- Increasing depth decreases bias, increases variance

Neural networks

- Deeper models decrease bias but can increase variance