CS 760: Machine Learning
Model Evaluation

Josiah Hanna

University of Wisconsin — Madison

9/21/2023

Announcements

Announcements

- Homework 2 is due at 9:30 AM on September 28.

Announcements

- Homework 2 is due at 9:30 AM on September 28.
- **Midterm** will be on October 18th from 5:45-7:15pm.

Announcements

- Homework 2 is due at 9:30 AM on September 28.
- **Midterm** will be on October 18th from 5:45-7:15pm.
  - If you have a known conflict that you have not communicated about yet, please do so this week.

Announcements

- Homework 2 is due at 9:30 AM on September 28.
- **Midterm** will be on October 18th from 5:45-7:15pm.
  - If you have a known conflict that you have not communicated about yet, please do so this week.
  - Make-ups will only be considered for emergencies or conflicts already communicated.

Outline

- **Wrapping up decision trees**
  - Variations, information gain, regression
  - Evaluation in decision trees: overfitting, pruning, variations
- **Evaluation: Generalization**
  - Train/test split, random sampling, cross validation
- **Evaluation: Metrics**
  - Confusion matrices, ROC curves, precision/recall

Outline

- **Wrapping up decision trees**
  - Variations, information gain, regression
  - Evaluation in decision trees: overfitting, pruning, variations
- **Evaluation: Generalization**
  - Train/test split, random sampling, cross validation
- **Evaluation: Metrics**
  - Confusion matrices, ROC curves, precision/recall

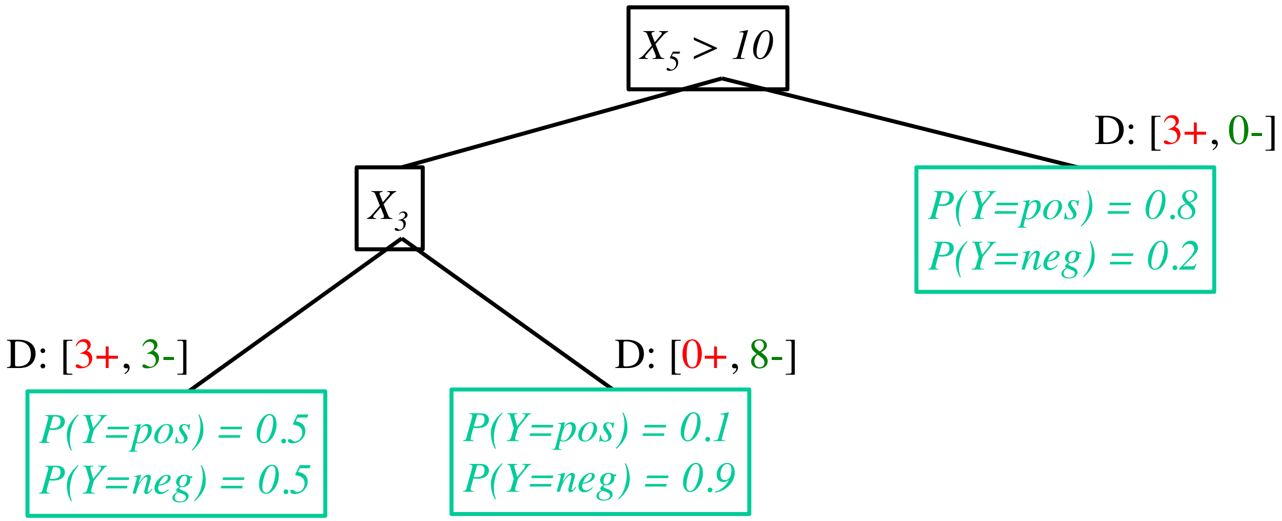
Variations

Variations

- Probability estimation trees
  - Leaves: estimate the probability of each class instead of a single class.

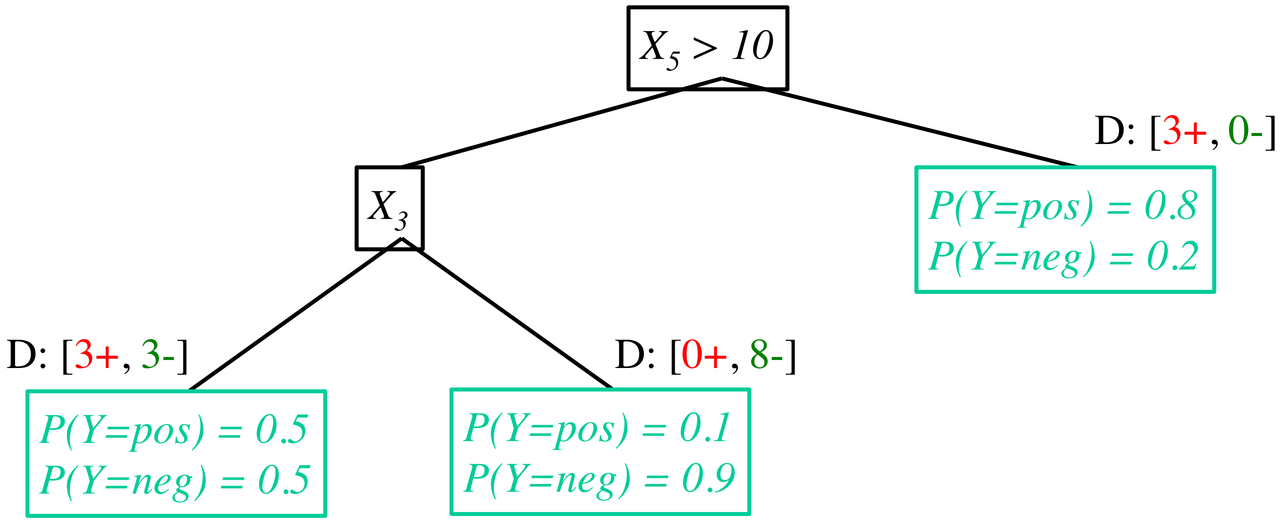
Variations

- Probability estimation trees
  - Leaves: estimate the probability of each class instead of a single class.



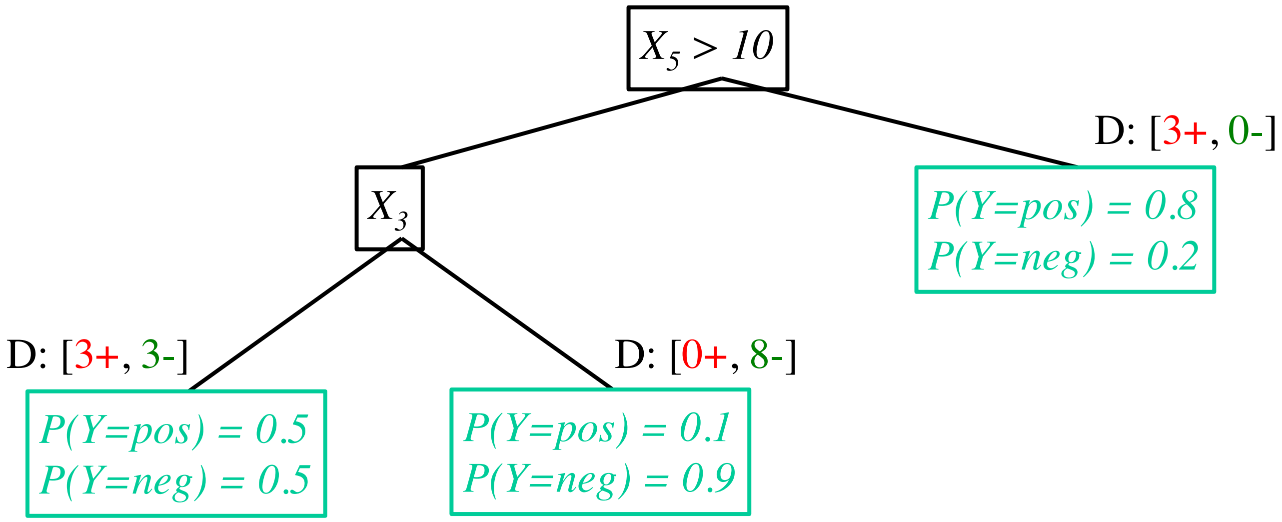
Variations

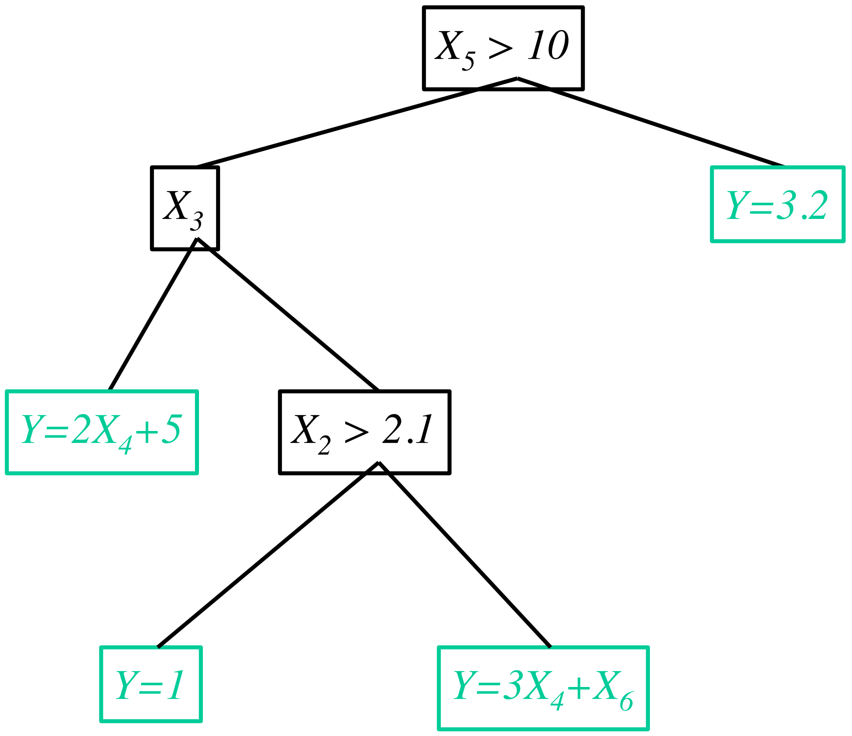
- Probability estimation trees
  - Leaves: estimate the probability of each class instead of a single class.
- Regression trees
  - Either numeric values (e.g. average label) or functions (e.g., linear functions) at each leaf.



Variations

- Probability estimation trees
  - Leaves: estimate the probability of each class instead of a single class.
- Regression trees
  - Either numeric values (e.g. average label) or functions (e.g., linear functions) at each leaf.





DT Learning: InfoGain Limitations

DT Learning: InfoGain Limitations

- InfoGain is biased towards tests with many outcomes
  - Splitting on it results in many branches, each of which is "pure" (has instances of only one class)
  - In the extreme: A feature that uniquely identifies each instance
  - **Maximal** information gain!

- Use **GainRatio**: normalize information gain by entropy

DT Learning: InfoGain Limitations

- InfoGain is biased towards tests with many outcomes
  - Splitting on it results in many branches, each of which is “pure” (has instances of only one class)
  - In the extreme: A feature that uniquely identifies each instance
  - **Maximal** information gain!

- Use **GainRatio**: normalize information gain by entropy

$$\mathrm{GainRatio}(D, S) = \frac{\mathrm{InfoGain}(D,S)}{H_D(S)} = \frac{H_D(Y) - H_D(Y|S)}{H_D(S)}$$

DT Learning: GainRatio

- Why?

DT Learning: GainRatio

- Why?
  - Suppose S is a *binary split*. InfoGain limited to 1 bit, no matter what.

DT Learning: GainRatio

- Why?
  - Suppose S is a *binary split*. InfoGain limited to 1 bit, no matter what.

$$\mathrm{InfoGain}(D, S) = H_D(Y) - H_D(Y|S)$$

DT Learning: GainRatio

- Why?
  - Suppose S is a *binary split*. InfoGain limited to 1 bit, no matter what.

$$\text{InfoGain}(D, S) = H_D(Y) - H_D(Y|S)$$

Intuition: at most, S tells us Y is in one half of its classes or the other

DT Learning: GainRatio

- Why?
  - Suppose S is a *binary split*. InfoGain limited to 1 bit, no matter what.

$$\text{InfoGain}(D, S) = H_D(Y) - H_D(Y|S)$$

Intuition: at most, S tells us Y is in one half of its classes or the other

  - Now suppose S is different for each instance (i.e., student number).

DT Learning: GainRatio

- Why?
  - Suppose S is a *binary split*. InfoGain limited to 1 bit, no matter what.

$$\text{InfoGain}(D, S) = H_D(Y) - H_D(Y|S)$$

Intuition: at most, S tells us Y is in one half of its classes or the other

  - Now suppose S is different for each instance (i.e., student number).
    - Uniquely determines Y for each point, but useless for generalization.

DT Learning: GainRatio

- Why?
  - Suppose S is a *binary split*. InfoGain limited to 1 bit, no matter what.

$$\mathrm{InfoGain}(D, S) = H_D(Y) - H_D(Y|S)$$

Intuition: at most, S tells us Y is in one half of its classes or the other

  - Now suppose S is different for each instance (i.e., student number).
    - Uniquely determines Y for each point, but useless for generalization.
    - But, then $H_D(Y|S) = 0$, so maximal information gain!

DT Learning: GainRatio

- Why?
  - Suppose S is a *binary split*. InfoGain limited to 1 bit, no matter what.

$$\mathrm{InfoGain}(D, S) = H_D(Y) - H_D(Y|S)$$

Intuition: at most, S tells us Y is in one half of its classes or the other

  - Now suppose S is different for each instance (i.e., student number).
    - Uniquely determines Y for each point, but useless for generalization.
    - But, then $H_D(Y|S) = 0$, so maximal information gain!
  - Control this by normalizing by $H_D(S)$.

DT Learning: GainRatio

- Why?
  - Suppose S is a *binary split*. InfoGain limited to 1 bit, no matter what.

$$\mathrm{InfoGain}(D, S) = H_D(Y) - H_D(Y|S)$$

Intuition: at most, S tells us Y is in one half of its classes or the other

  - Now suppose S is different for each instance (i.e., student number).
    - Uniquely determines Y for each point, but useless for generalization.
    - But, then $H_D(Y|S) = 0$, so maximal information gain!
  - Control this by normalizing by $H_D(S)$.
    - Above: for *n* instances, $H_D(S) = \log_2(n)$

DT Learning: GainRatio

- Why?
  - Suppose S is a *binary split*. InfoGain limited to 1 bit, no matter what.

$$\mathrm{InfoGain}(D,S) = H_D(Y) - H_D(Y|S)$$

Intuition: at most, S tells us Y is in one half of its classes or the other

  - Now suppose S is different for each instance (i.e., student number).
    - Uniquely determines Y for each point, but useless for generalization.
    - But, then $H_D(Y|S) = 0$, so maximal information gain!
  - Control this by normalizing by $H_D(S)$.
    - Above: for *n* instances, $H_D(S) = \log_2(n)$

$$\mathrm{GainRatio}(D,S) = \frac{\mathrm{InfoGain}(D,S)}{H_D(S)} = \frac{H_D(Y) - H_D(Y|S)}{H_D(S)}$$

Candidate splits for regression

Last time we discussed candidate splits for numeric features for classification. These methods depended on discrete labels.

Several options:

- Candidate split at every data point.
- Candidate splits along a grid.
- In either case, may need to filter splits using some heuristic.

Decision Trees: Comments

- Decision trees are a widely used approach

Decision Trees: Comments

- Decision trees are a widely used approach
  - Many variations

Decision Trees: Comments

- Decision trees are a widely used approach
  - Many variations
- Provides humanly comprehensible models

Decision Trees: Comments

- Decision trees are a widely used approach
  - Many variations
- Provides humanly comprehensible models
  - When trees not too big!

Decision Trees: Comments

- Decision trees are a widely used approach
  - Many variations
- Provides humanly comprehensible models
  - When trees not too big!
- Training is insensitive to monotone transformations of numeric features.

Decision Trees: Comments

- Decision trees are a widely used approach
  - Many variations
- Provides humanly comprehensible models
  - When trees not too big!
- Training is insensitive to monotone transformations of numeric features.
- Implementation can (and does) vary, performance may depend on specific choices.

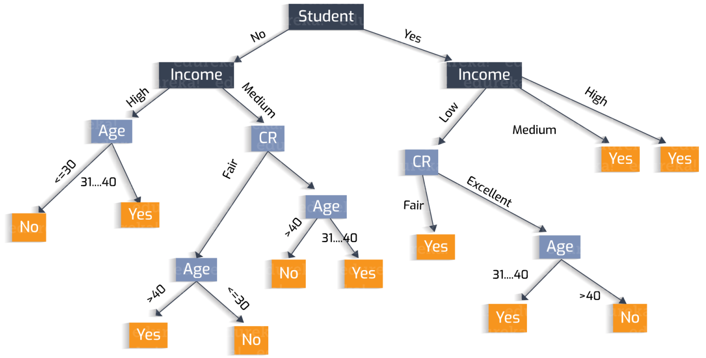
Decision Trees: Comments

- Decision trees are a widely used approach
  - Many variations
- Provides humanly comprehensible models
  - When trees not too big!
- Training is insensitive to monotone transformations of numeric features.
- Implementation can (and does) vary, performance may depend on specific choices.
  - Popular variants: ID3, C4.5, CART

Decision Trees: Learning

- **Learning Algorithm:**

$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$



MakeSubtree(set of training instances D)

C = **DetermineCandidateSplits**(D)

if **stopping criteria** is met

make a leaf node N

determine class label for N

else

make an internal node N

S = **FindBestSplit**(D, C)

for each group k of S

D_k = subset of training data in group k

k^{th} child of N = MakeSubtree(D_k)

return subtree rooted at N

Model selection in decision trees

Evaluation: Accuracy

Evaluation: Accuracy

- Can we just calculate the fraction of training instances that are correctly classified?

Evaluation: Accuracy

- Can we just calculate the fraction of training instances that are correctly classified?
- Consider a problem domain in which instances are assigned labels at random with $P(Y = 1) = 0.5$
  - How accurate would it be on its training set, if you stop when all instances are in the same class?
  - How accurate would a learned decision tree be on previously unseen instances?

Evaluation: Accuracy

- Can we just calculate the fraction of training instances that are correctly classified?
- Consider a problem domain in which instances are assigned labels at random with $P(Y = 1) = 0.5$
  - How accurate would it be on its training set, if you stop when all instances are in the same class?
  - How accurate would a learned decision tree be on previously unseen instances?

- Recall: our goal is to do well on *future data*.

Evaluation: Accuracy

Evaluation: Accuracy

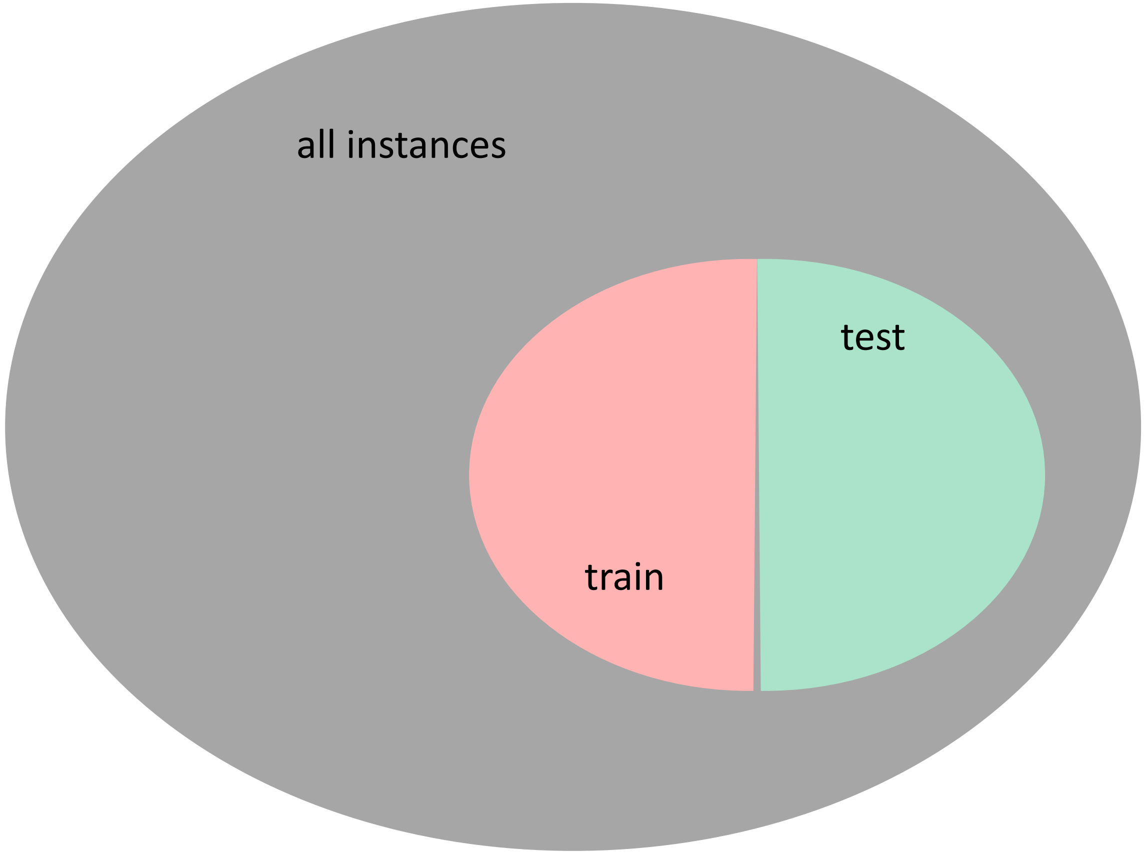
To get unbiased estimate of model accuracy, we must use a set of instances that are **held-aside** during learning

- This is called a **test set**

Evaluation: Accuracy

To get unbiased estimate of model accuracy, we must use a set of instances that are **held-aside** during learning

- This is called a **test set**



Overfitting

Notation: error of model *h* over

Overfitting

Notation: error of model *h* over

- training data: $\mathrm{error}_D(h)$

Overfitting

Notation: error of model *h* over

- training data: $\text{error}_D(h)$
- entire distribution of data: $\text{error}_P(h)$

Overfitting

Notation: error of model *h* over

- training data: $\text{error}_D(h)$
- entire distribution of data: $\text{error}_P(h)$

Model *h* **overfits** training data if it has

Overfitting

Notation: error of model *h* over

- training data: $\text{error}_D(h)$
- entire distribution of data: $\text{error}_P(h)$

Model *h* **overfits** training data if it has

- a low error on the training data (low $\text{error}_D(h)$)

Overfitting

Notation: error of model h over

- training data: $\text{error}_D(h)$
- entire distribution of data: $\text{error}_P(h)$

Model h **overfits** training data if it has

- a low error on the training data (low $\text{error}_D(h)$)
- high error on the entire distribution (high $\text{error}_P(h)$)

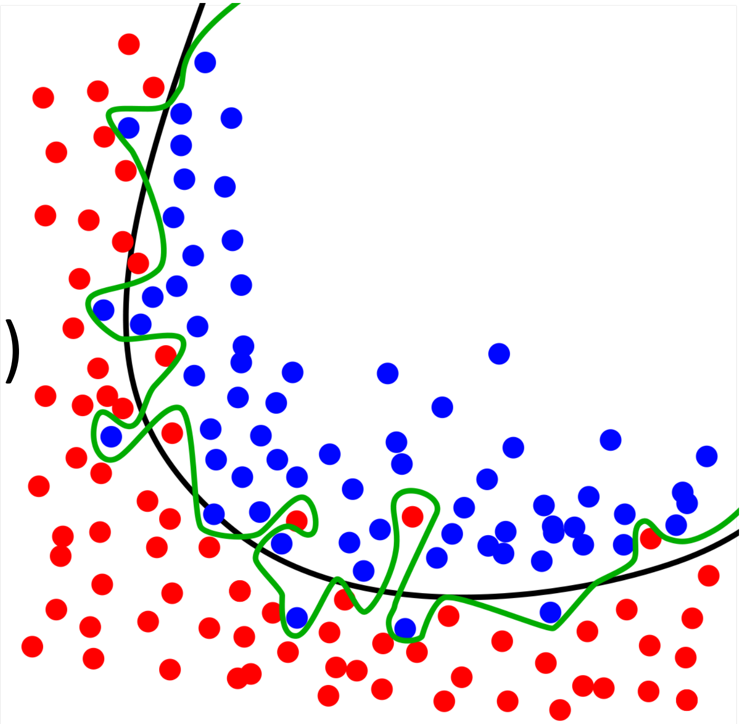
Overfitting

Notation: error of model *h* over

- training data: $\text{error}_D(h)$
- entire distribution of data: $\text{error}_P(h)$

Model *h* **overfits** training data if it has

- a low error on the training data (low $\text{error}_D(h)$)
- high error on the entire distribution (high $\text{error}_P(h)$)



Wikipedia

Overfitting Example: Noisy Data

Target function is $Y = X_1 \wedge X_2$

Overfitting Example: Noisy Data

Target function is $Y = X_1 \wedge X_2$

- There is **noise** in some feature values

Overfitting Example: Noisy Data

Target function is $Y = X_1 \wedge X_2$

- There is **noise** in some feature values
- Training set:

Overfitting Example: Noisy Data

Target function is $Y = X_1 \wedge X_2$

- There is **noise** in some feature values
- Training set:

X_1	X_2	X_3	X_4	X_5	…	Y
t	t	t	t	t	…	t
t	t	f	f	t	…	t
t	f	t	t	f	…	t
t	f	f	t	f	…	f
t	f	t	f	f	…	f
f	t	t	f	t	…	f

Overfitting Example: Noisy Data

Target function is $Y = X_1 \wedge X_2$

- There is **noise** in some feature values
- Training set:

X_1	X_2	X_3	X_4	X_5	…	Y
t	t	t	t	t	…	t
t	t	f	f	t	…	t
t	f	t	t	f	…	t
t	f	f	t	f	…	f
t	f	t	f	f	…	f
f	t	t	f	t	…	f

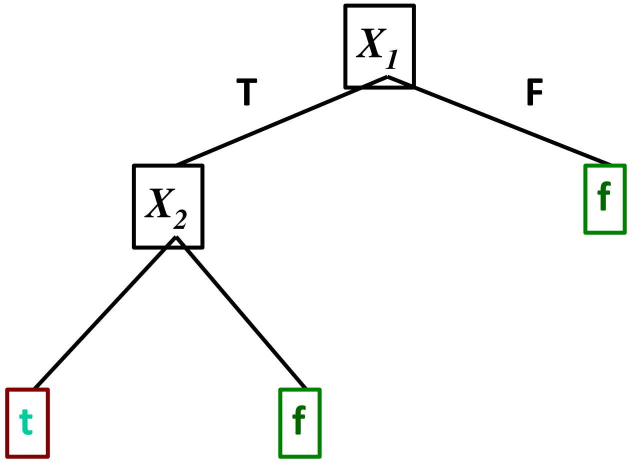
noisy value

Overfitting Example: Noisy Data

Correct tree

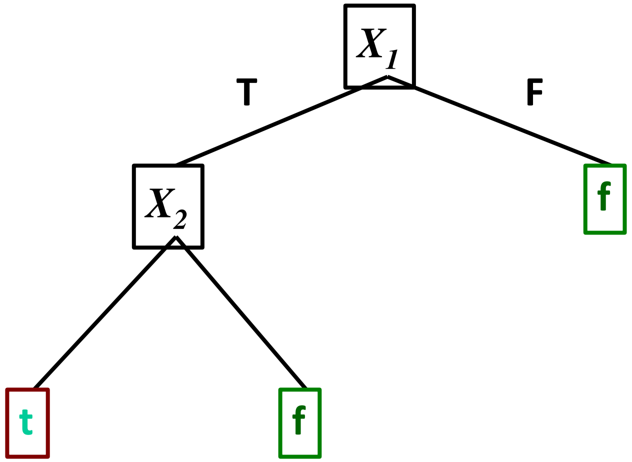
Overfitting Example: Noisy Data

Correct tree



Overfitting Example: Noisy Data

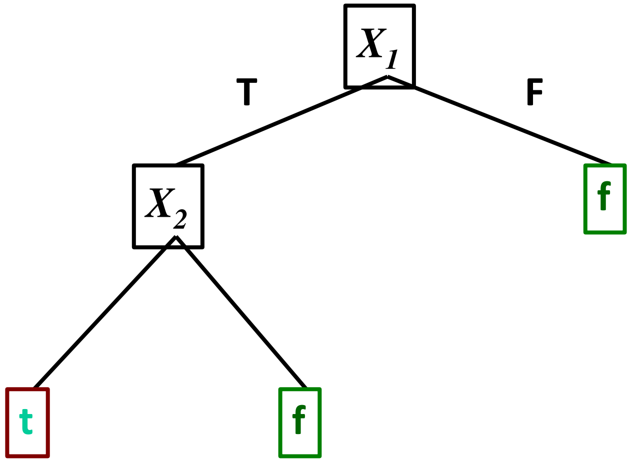
Correct tree



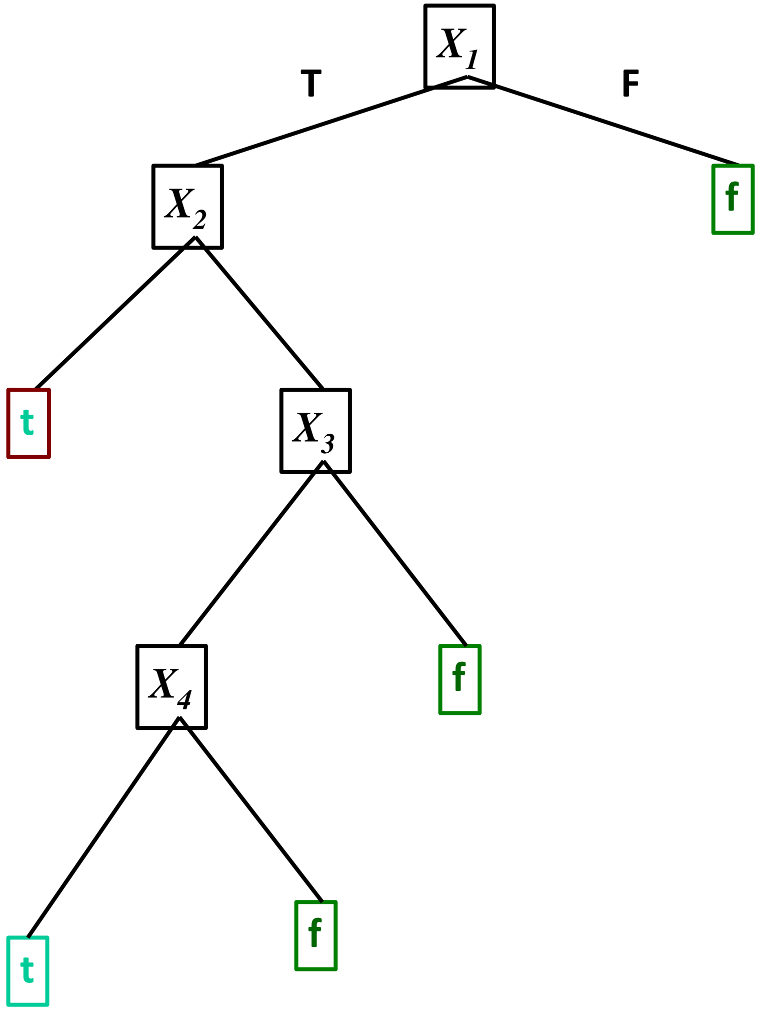
Tree that fits noisy training data

Overfitting Example: Noisy Data

Correct tree



Tree that fits noisy training data



Overfitting Example: Noise-Free Data

Target function is $Y = X_1 \wedge X_2$

Overfitting Example: Noise-Free Data

Target function is $Y = X_1 \wedge X_2$

- What about irrelevant features?

Overfitting Example: Noise-Free Data

Target function is $Y = X_1 \wedge X_2$

- What about irrelevant features?
- Training set:

Overfitting Example: Noise-Free Data

Target function is $Y = X_1 \wedge X_2$

- What about irrelevant features?
- Training set:

X_1	X_2	X_3	X_4	X_5	…	Y
t	t	t	t	t	…	t
t	t	t	f	t	…	t
t	t	t	t	f	…	t
t	f	f	t	f	…	f
f	t	f	f	t	…	f

Overfitting Example: Noise-Free Data

Overfitting Example: Noise-Free Data

- Training set is a **limited sample.** Might be (combinations of) features that are correlated with the target concept by chance

Overfitting Example: Noise-Free Data

- Training set is a **limited sample.** Might be (combinations of) features that are correlated with the target concept by chance
- Assume, P(X_3 = t) = 0.5 for both classes and P(Y = t) = 0.67

Overfitting Example: Noise-Free Data

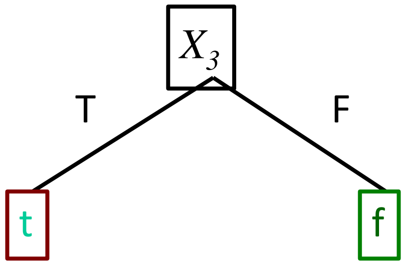
- Training set is a **limited sample.** Might be (combinations of) features that are correlated with the target concept by chance
- Assume, P(X_3 = t) = 0.5 for both classes and P(Y = t) = 0.67

X_1	X_2	X_3	X_4	X_5	…	Y
t	t	t	t	t	…	t
t	t	t	f	t	…	t
t	t	t	t	f	…	t
t	f	f	t	f	…	f
f	t	f	f	t	…	f

Overfitting Example: Noise-Free Data

- Training set is a **limited sample.** Might be (combinations of) features that are correlated with the target concept by chance
- Assume, P(X_3 = t) = 0.5 for both classes and P(Y = t) = 0.67

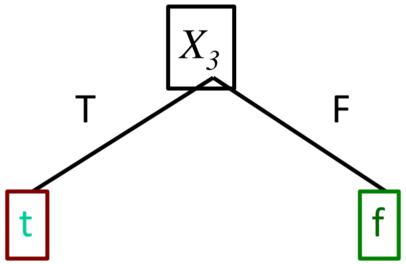
X_1	X_2	X_3	X_4	X_5	…	Y
t	t	t	t	t	…	t
t	t	t	f	t	…	t
t	t	t	t	f	…	t
t	f	f	t	f	…	f
f	t	f	f	t	…	f



Overfitting Example: Noise-Free Data

- Training set is a **limited sample.** Might be (combinations of) features that are correlated with the target concept by chance
- Assume, $P(X_3 = t) = 0.5$ for both classes and $P(Y = t) = 0.67$

X_1	X_2	X_3	X_4	X_5	…	Y
t	t	t	t	t	…	t
t	t	t	f	t	…	t
t	t	t	t	f	…	t
t	f	f	t	f	…	f
f	t	f	f	t	…	f

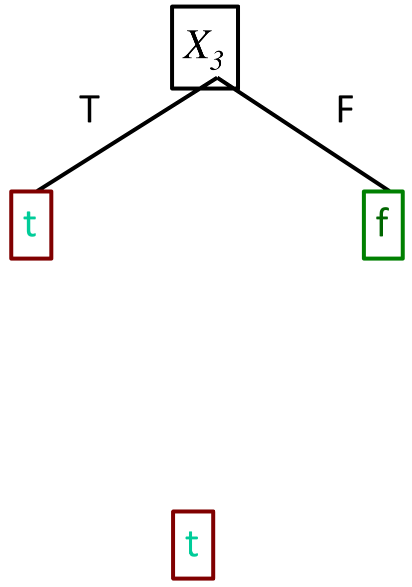


Training set accuracy	**Test set accuracy**
100%	50%

Overfitting Example: Noise-Free Data

- Training set is a **limited sample.** Might be (combinations of) features that are correlated with the target concept by chance
- Assume, P(X_3 = t) = 0.5 for both classes and P(Y = t) = 0.67

X_1	X_2	X_3	X_4	X_5	…	Y
t	t	t	t	t	…	t
t	t	t	f	t	…	t
t	t	t	t	f	…	t
t	f	f	t	f	…	f
f	t	f	f	t	…	f

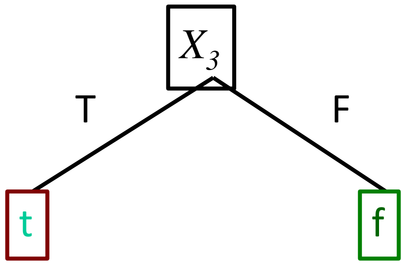
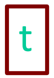


Training set accuracy	**Test set accuracy**
100%	50%

Overfitting Example: Noise-Free Data

- Training set is a **limited sample.** Might be (combinations of) features that are correlated with the target concept by chance
- Assume, $P(X_3 = t) = 0.5$ for both classes and $P(Y = t) = 0.67$

X_1	X_2	X_3	X_4	X_5	…	Y
t	t	t	t	t	…	t
t	t	t	f	t	…	t
t	t	t	t	f	…	t
t	f	f	t	f	…	f
f	t	f	f	t	…	f

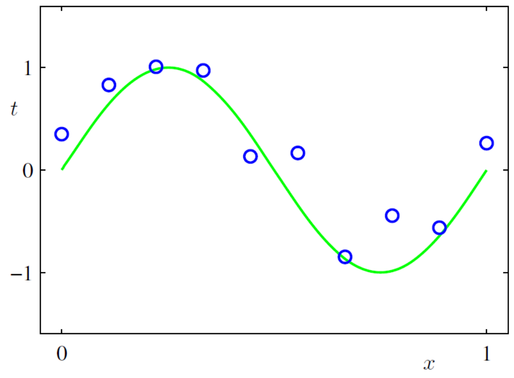
	Training set accuracy	**Test set accuracy**
	100%	50%
	66%	66%

Overfitting Example: Polynomial Regression

- Training set is a **limited sample.** Might be (combinations of) features that are correlated with the target concept by chance

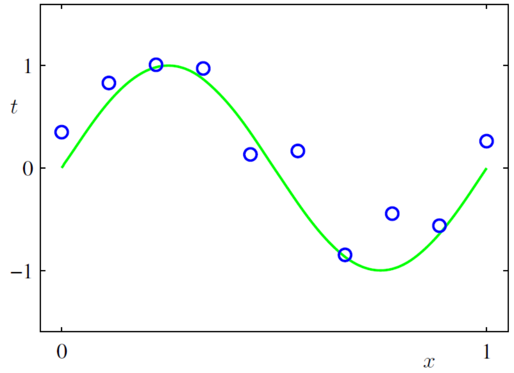
Overfitting Example: Polynomial Regression

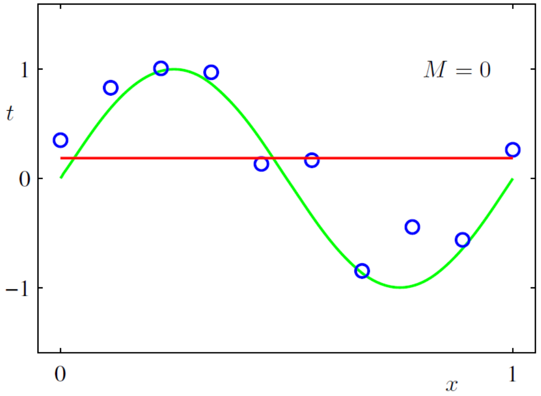
- Training set is a **limited sample.** Might be (combinations of) features that are correlated with the target concept by chance



Overfitting Example: Polynomial Regression

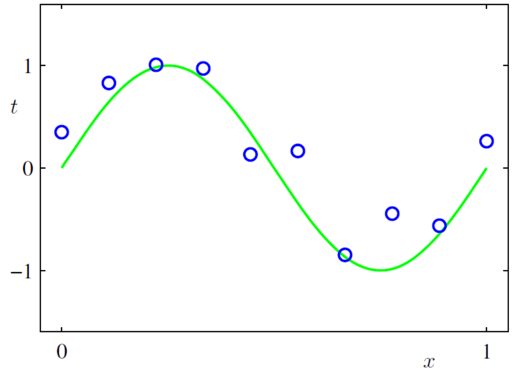
- Training set is a **limited sample.** Might be (combinations of) features that are correlated with the target concept by chance

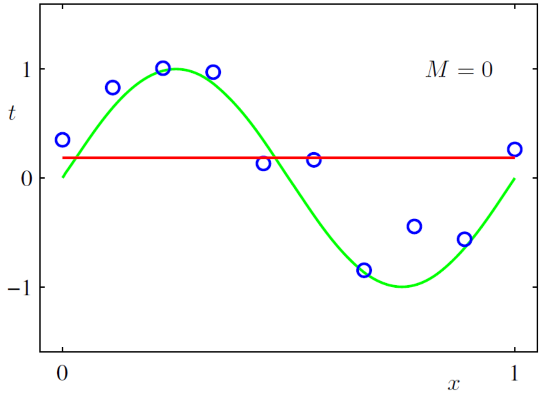


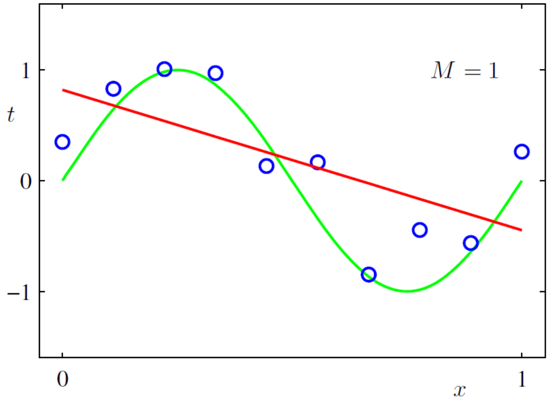


Overfitting Example: Polynomial Regression

- Training set is a **limited sample.** Might be (combinations of) features that are correlated with the target concept by chance

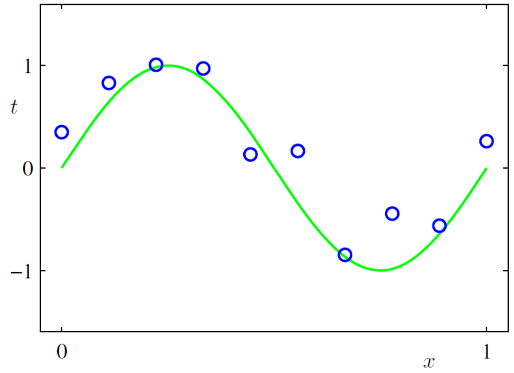


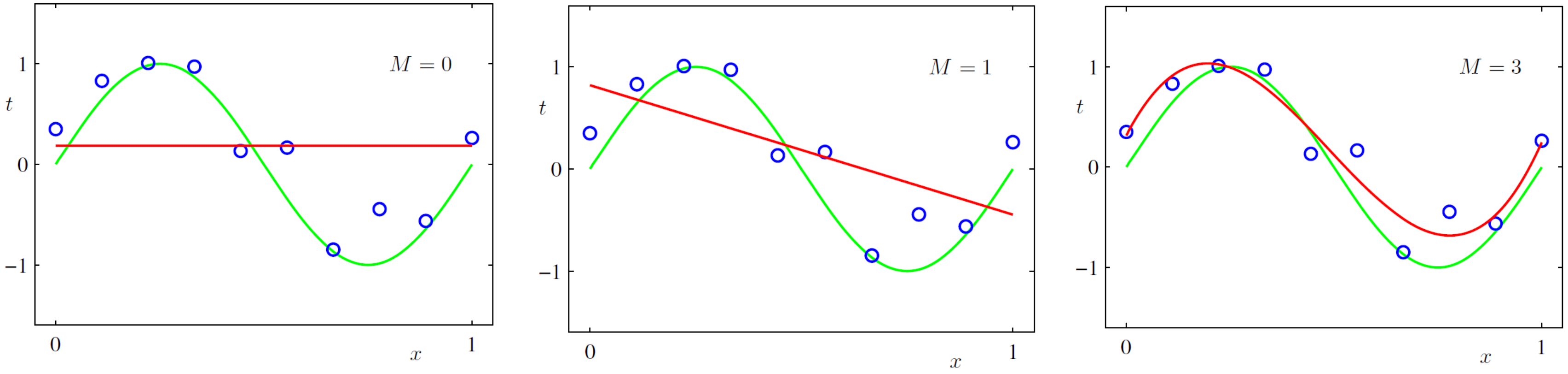




Overfitting Example: Polynomial Regression

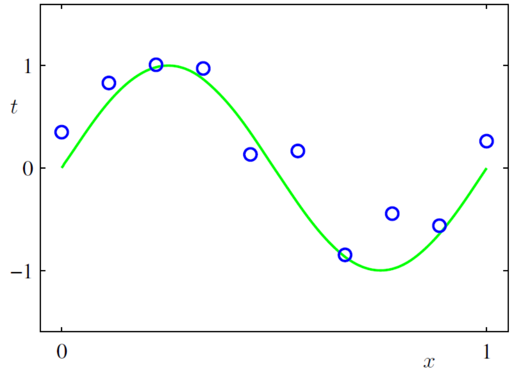
- Training set is a **limited sample.** Might be (combinations of) features that are correlated with the target concept by chance

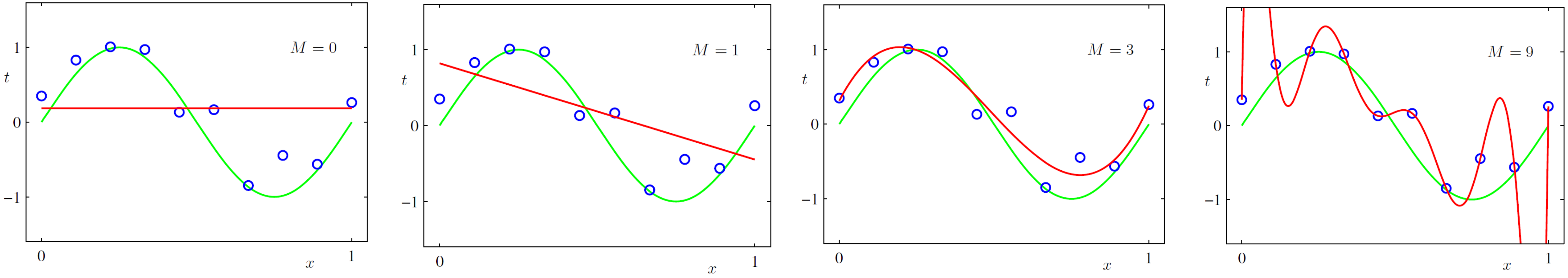




Overfitting Example: Polynomial Regression

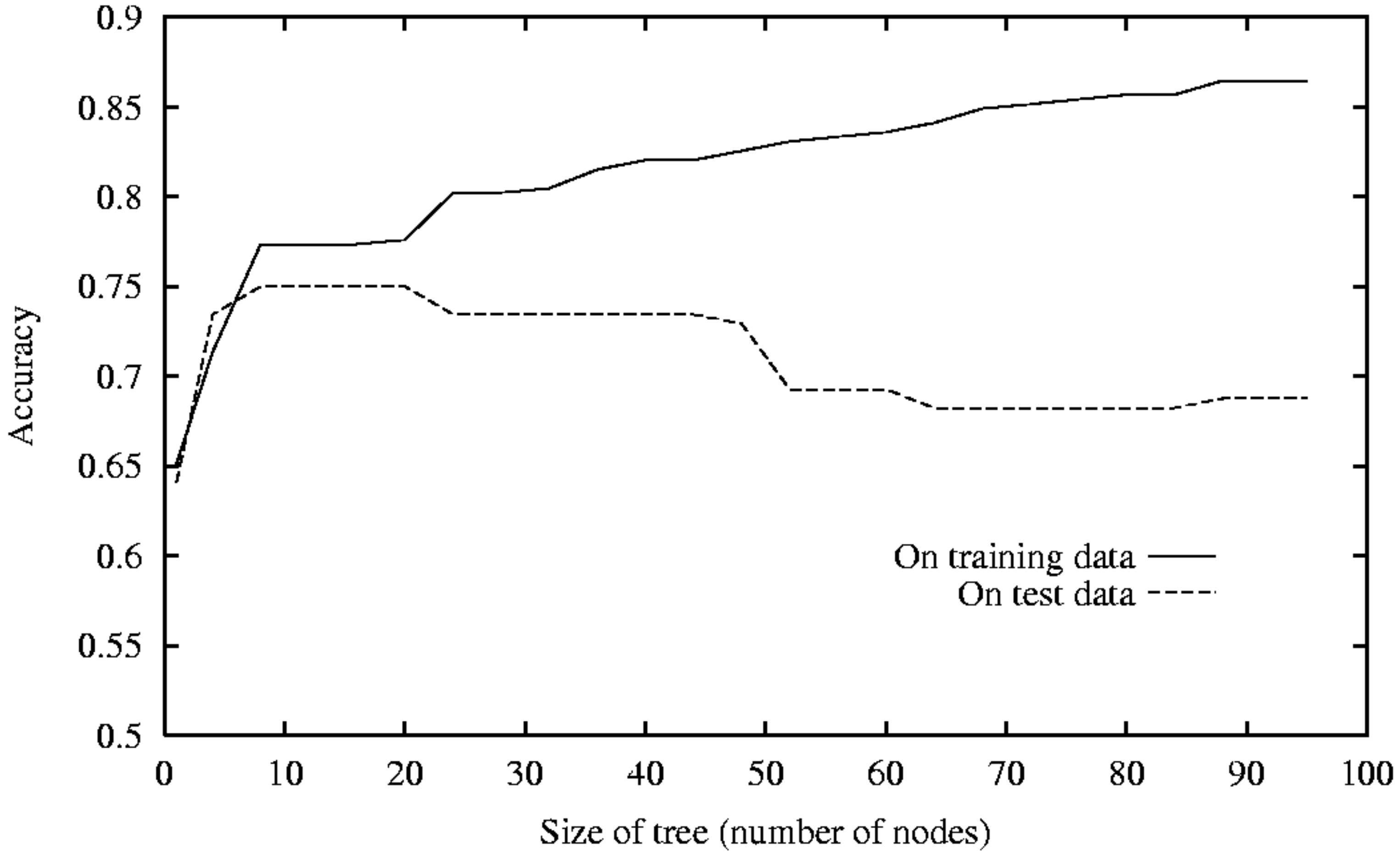
- Training set is a **limited sample.** Might be (combinations of) features that are correlated with the target concept by chance





Overfitting: Tree Size vs. Accuracy

- Tree size vs accuracy



General Phenomenon

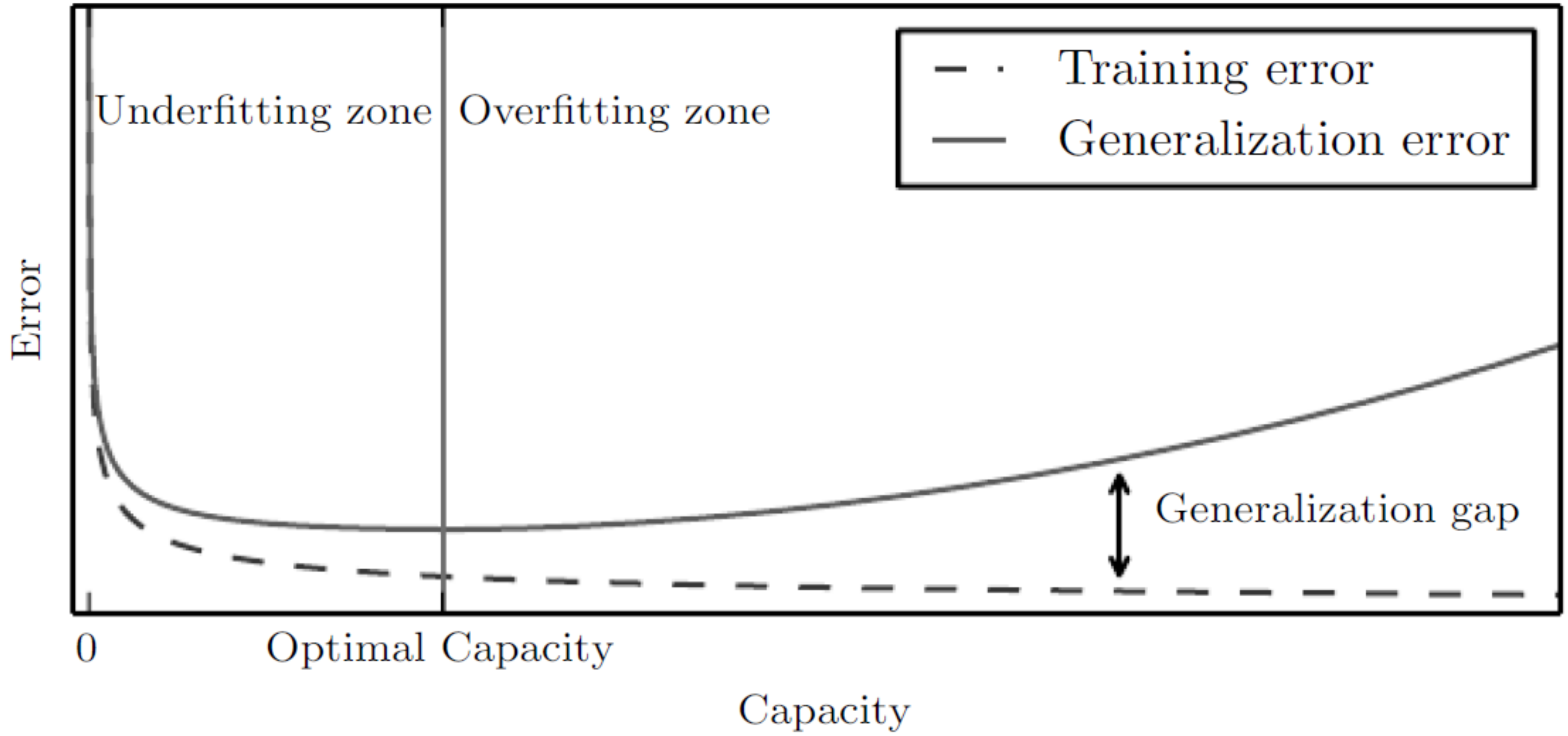


Figure from *Deep Learning*, Goodfellow, Bengio and Courville

Decision Tree Learning: Avoiding Overfitting

Decision Tree Learning: Avoiding Overfitting

Two **general strategies** to avoid overfitting

1. ***During training***: create two-way instead of multi-way splits, stop if further splitting not justified by a statistical test

2. ***Post-pruning***: grow a large tree, then prune back some nodes
   - E.g: evaluate impact on *tuning-set* accuracy of pruning each node
   - Greedily remove the one that most improves *tuning-set* accuracy

Tuning Sets

- A *tuning set* (a.k.a. *validation set*) is

Tuning Sets

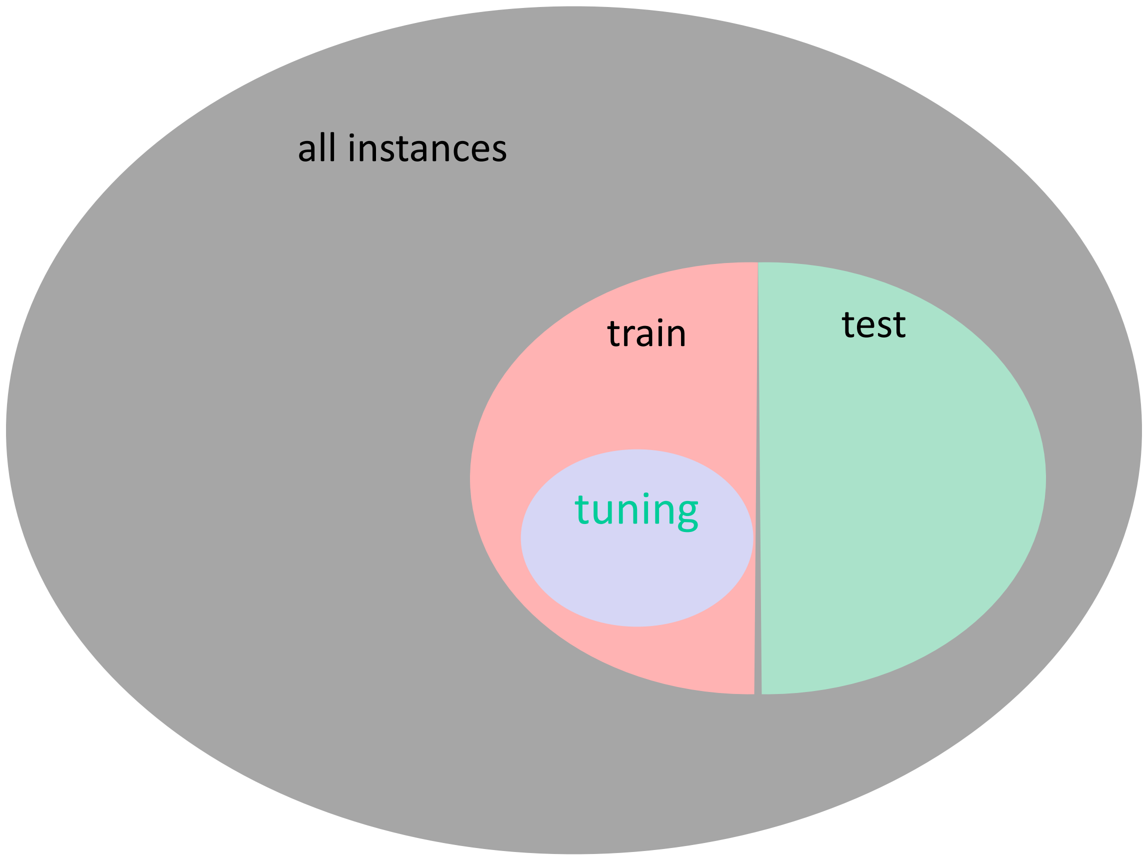
- A *tuning set* (a.k.a. *validation set*) is
  - not used for primary training process (e.g. tree growing)

Tuning Sets

- A *tuning set* (a.k.a. *validation set*) is
  - not used for primary training process (e.g. tree growing)
  - but used to select among models (e.g. trees pruned to varying degrees)

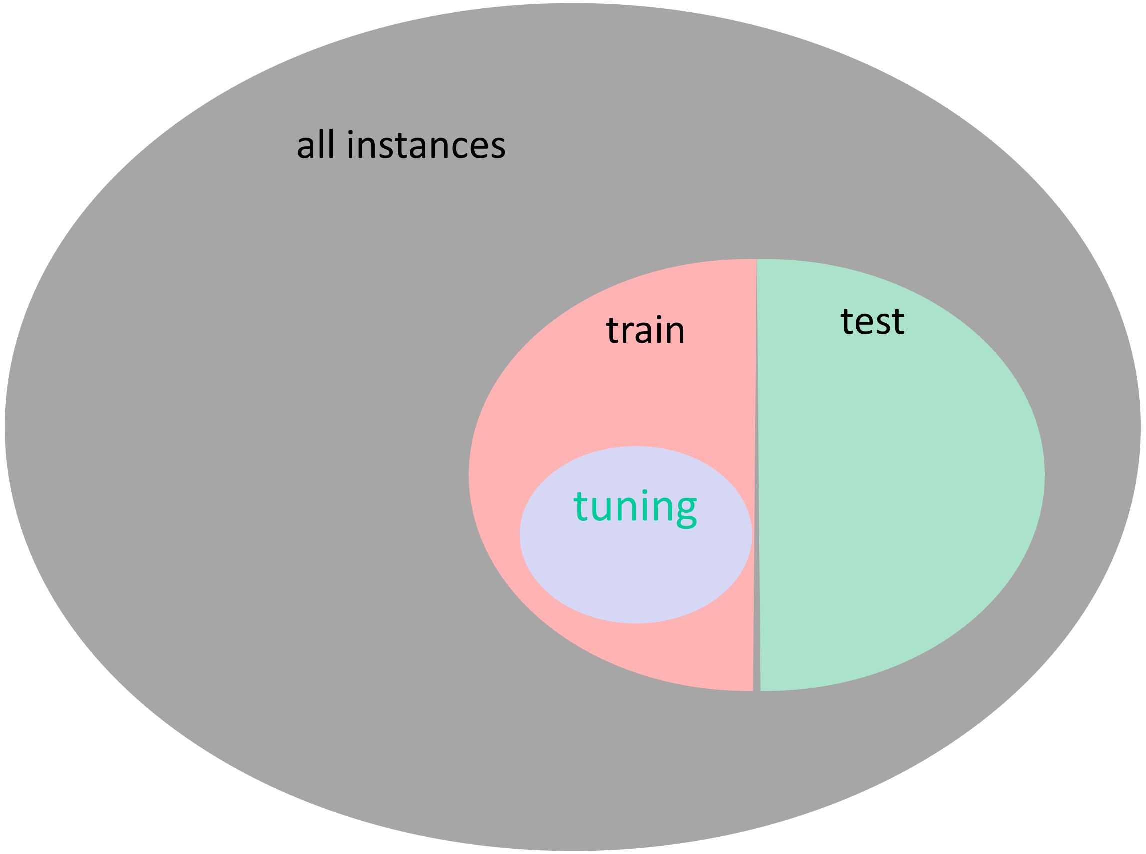
Tuning Sets

- A *tuning set* (a.k.a. *validation set*) is
  - not used for primary training process (e.g. tree growing)
  - but used to select among models (e.g. trees pruned to varying degrees)



Tuning Sets

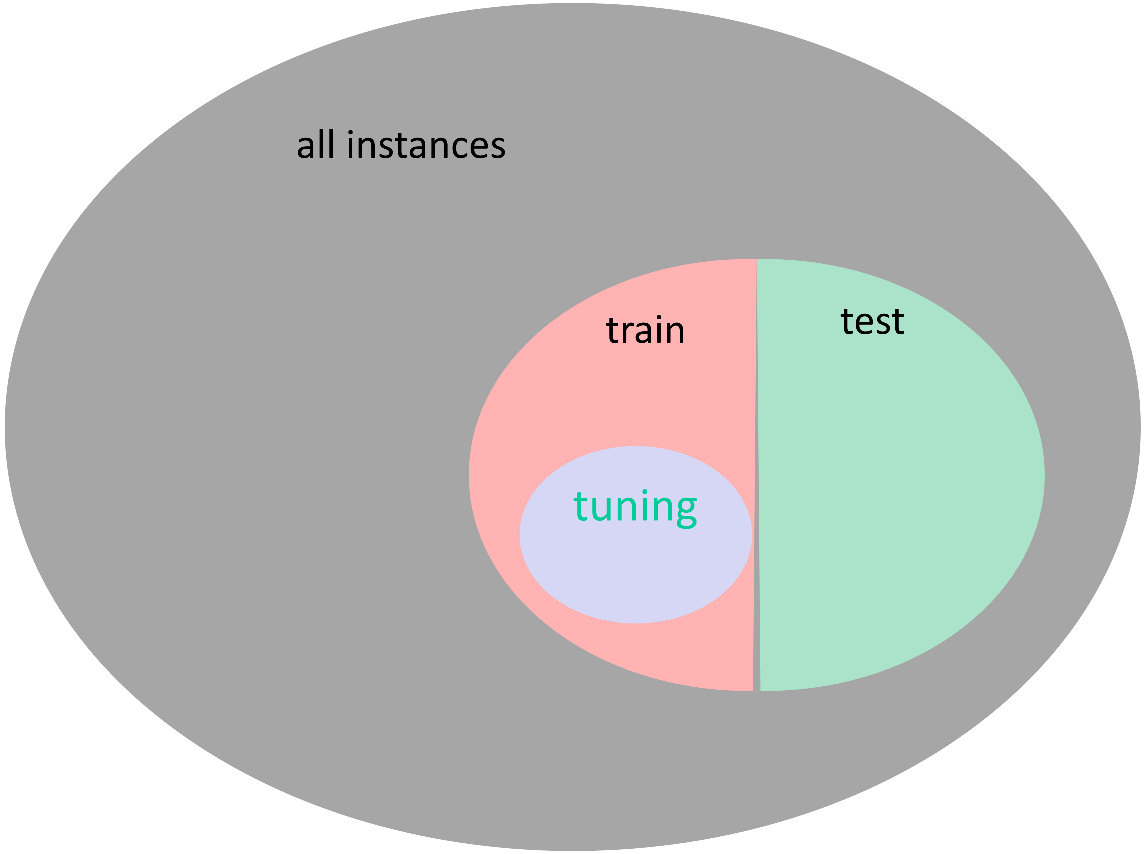
- A *tuning set* (a.k.a. *validation set*) is
  - not used for primary training process (e.g. tree growing)
  - but used to select among models (e.g. trees pruned to varying degrees)



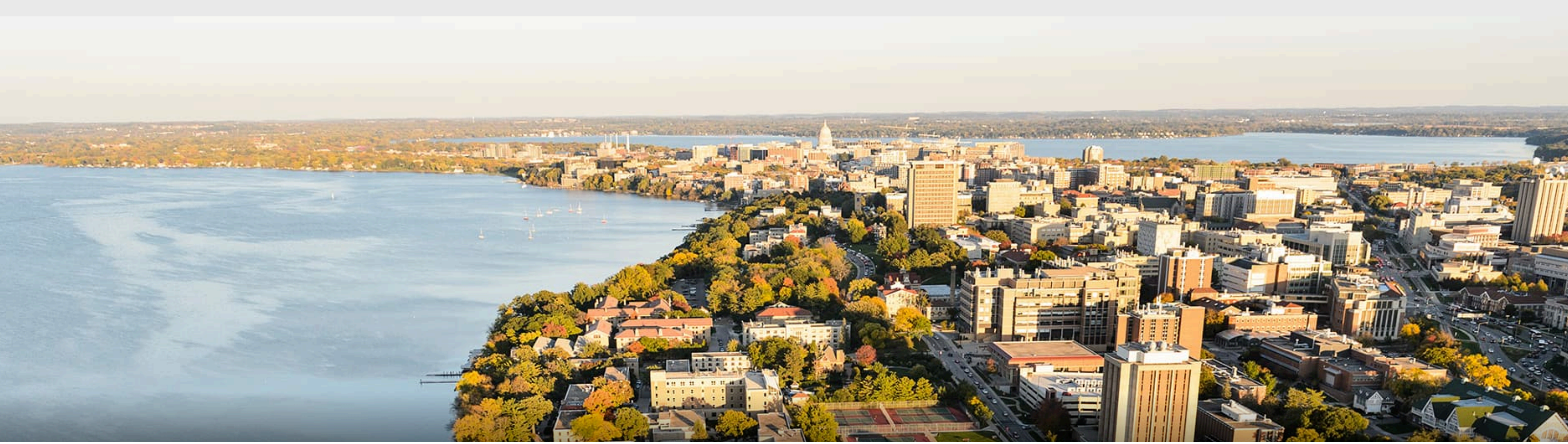
- Why can you not use the training set to prune?

Tuning Sets

- A *tuning set* (a.k.a. *validation set*) is
  - not used for primary training process (e.g. tree growing)
  - but used to select among models (e.g. trees pruned to varying degrees)



- Why can you not use the training set to prune?
- Why can you not use the test set to prune?



Break & Quiz

Which of the following statements is TRUE?

1. If there is no noise, then there is no overfitting.
2. Overfitting may improve the generalization ability of a model.
3. Generalization error is monotone with respect to the capacity/ complexity of a model.
4. More training data may help preventing overfitting.

Which of the following statements is TRUE?

1. If there is no noise, then there is no overfitting.
2. Overfitting may improve the generalization ability of a model.
3. Generalization error is monotone with respect to the capacity/complexity of a model.
4. More training data may help preventing overfitting.

1. We can still have false correlation that leads to overfitting.
2. Overfitting would undermine the generalization ability.
3. Generalization error would first decrease and then increase as the model capacity increases.
4. Increasing training data size would help better approximate the true distribution.

True or False:

In k-NN, using large k leads to over-fitting.

True or False:

In k-NN, using large k leads to over-fitting.

Ans: False!

Outline

- **Wrapping up decision trees**
  - Information gain, stopping criteria
  - Model selection in decision trees: overfitting, pruning, variations
- **Evaluation: Generalization**
  - Train/test split, random sampling, cross validation
- **Evaluation: Metrics**
  - Confusion matrices, ROC curves, precision/recall

Accuracy of a Model

Accuracy of a Model

- How can we get estimate the accuracy of a learned model?

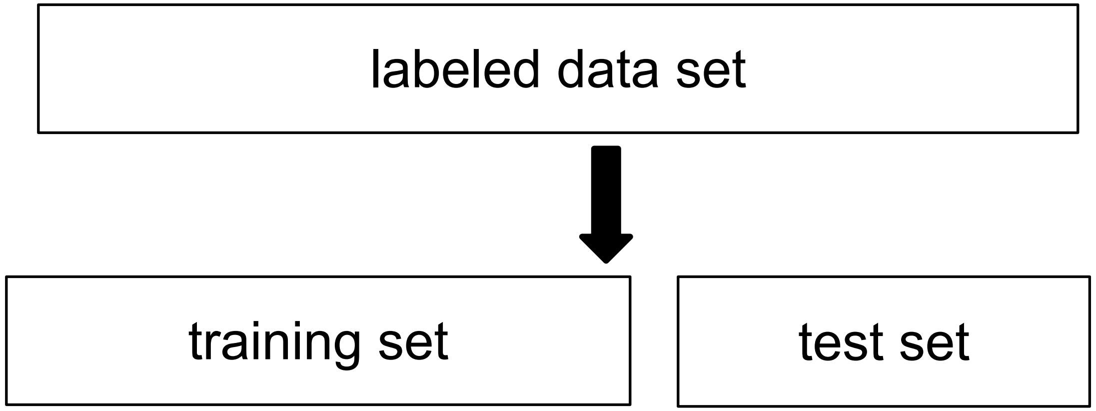
Accuracy of a Model

- How can we get estimate the accuracy of a learned model?

labeled data set

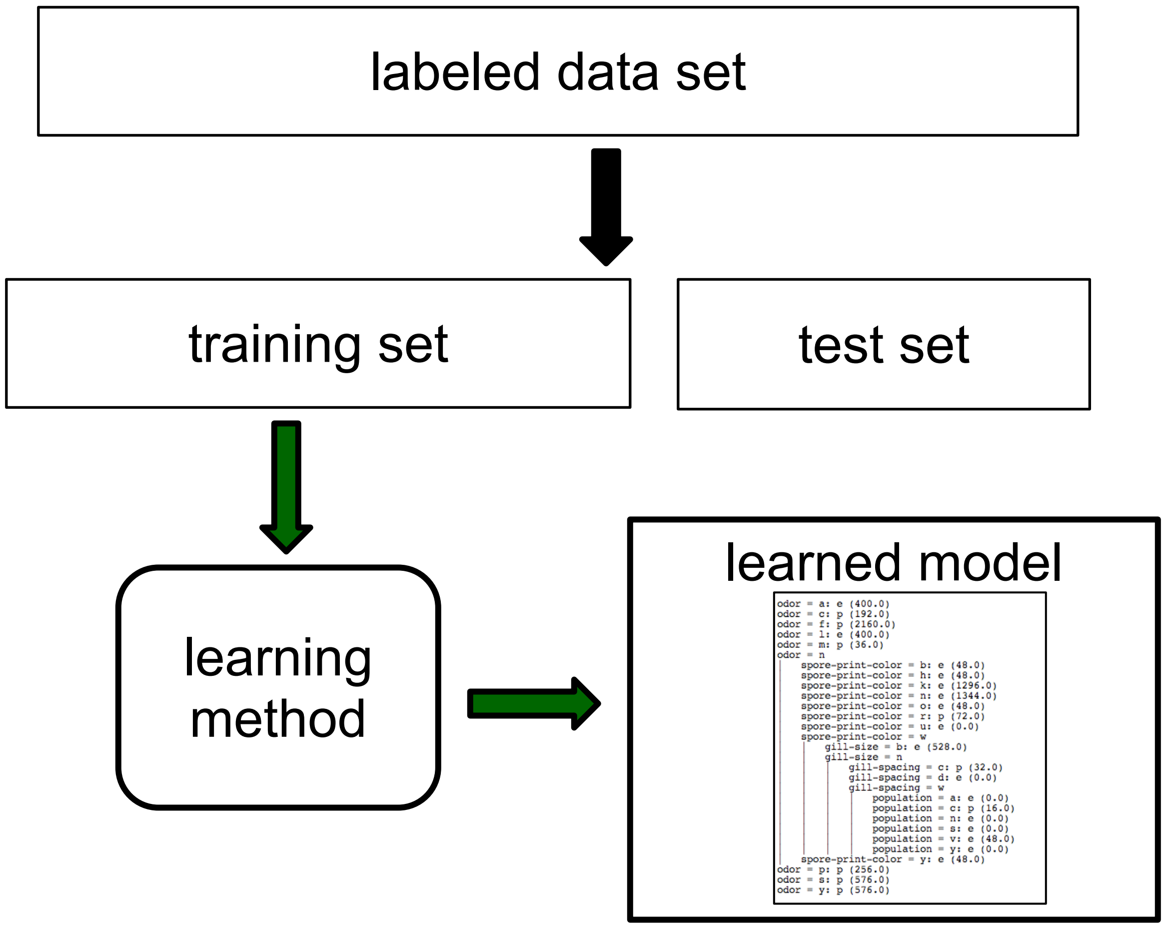
Accuracy of a Model

- How can we get estimate the accuracy of a learned model?



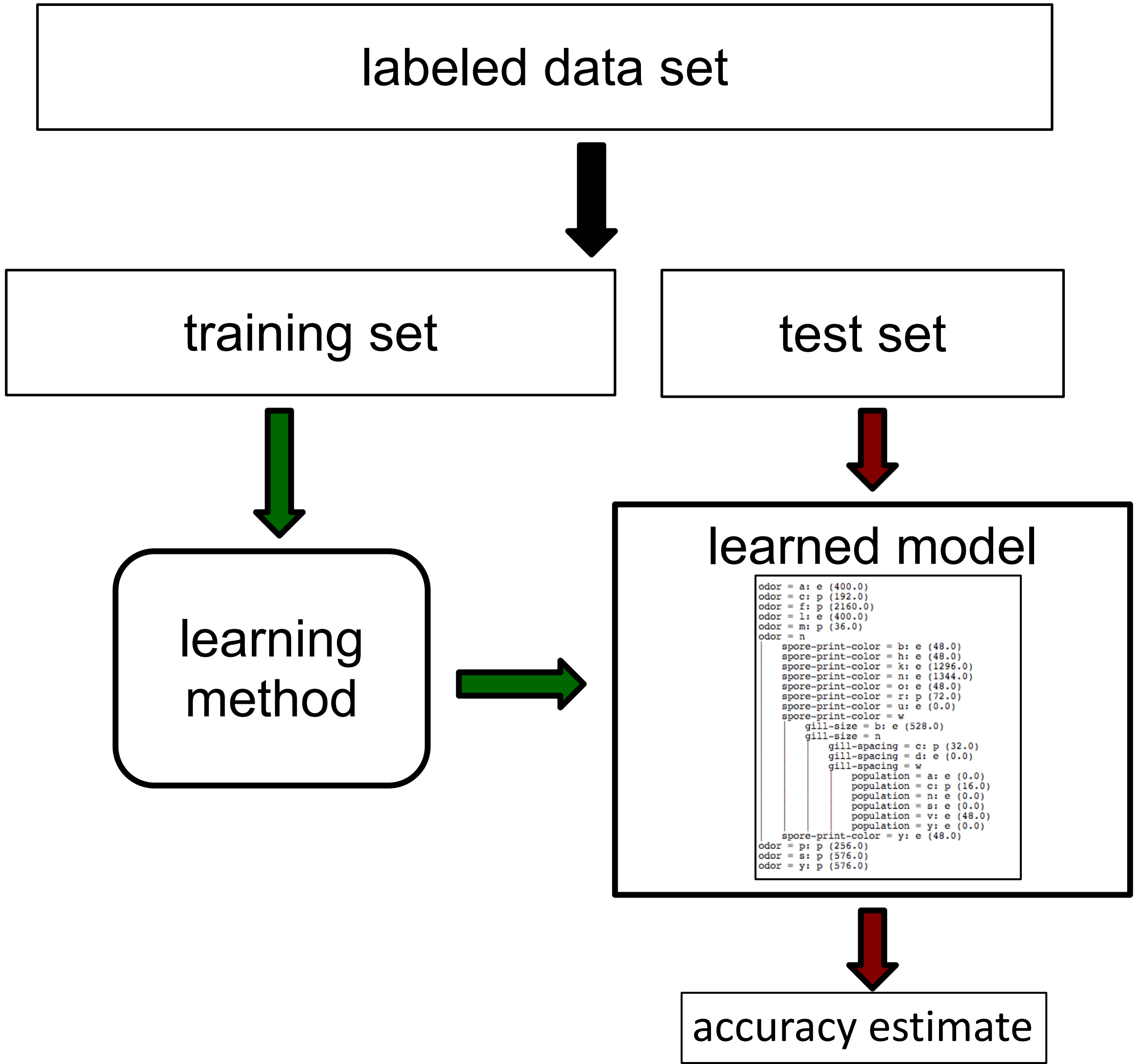
Accuracy of a Model

- How can we get estimate the accuracy of a learned model?



Accuracy of a Model

- How can we get estimate the accuracy of a learned model?



Using a Test Set

- How can we estimate the accuracy of a learned model?

Using a Test Set

- How can we estimate the accuracy of a learned model?
  - When learning a model, you should pretend that you don’t have the test data yet

Using a Test Set

- How can we estimate the accuracy of a learned model?
  - When learning a model, you should pretend that you don’t have the test data yet
  - If the test-set labels influence the learned model in any way, accuracy estimates will not be correct, as you may have fitted to your test set.

Using a Test Set

- How can we estimate the accuracy of a learned model?
  - When learning a model, you should pretend that you don’t have the test data yet
  - If the test-set labels influence the learned model in any way, accuracy estimates will not be correct, as you may have fitted to your test set.

- **Don’t train on the test set!**

Single Train/Test Split: Limitations

- May not have enough data for sufficiently large training/test sets

Single Train/Test Split: Limitations

- May not have enough data for sufficiently large training/test sets
  - A **larger test set** gives us more reliable estimate of accuracy (i.e. a lower variance estimate)

Single Train/Test Split: Limitations

- May not have enough data for sufficiently large training/test sets
  - A **larger test set** gives us more reliable estimate of accuracy (i.e. a lower variance estimate)
  - But… a **larger training set** will be more representative of how much data we actually have for learning process

Single Train/Test Split: Limitations

- May not have enough data for sufficiently large training/test sets
  - A **larger test set** gives us more reliable estimate of accuracy (i.e. a lower variance estimate)
  - But… a **larger training set** will be more representative of how much data we actually have for learning process

- A single training set does not tell us how sensitive accuracy is to a particular training sample

Strategy I: Random Resampling

Strategy I: Random Resampling

- Address the second issue by repeatedly randomly partitioning the available data into training and test sets.

Strategy I: Random Resampling

- Address the second issue by repeatedly randomly partitioning the available data into training and test sets.

labeled data set

+++++- - - - -

Strategy I: Random Resampling

- Address the second issue by repeatedly randomly partitioning the available data into training and test sets.

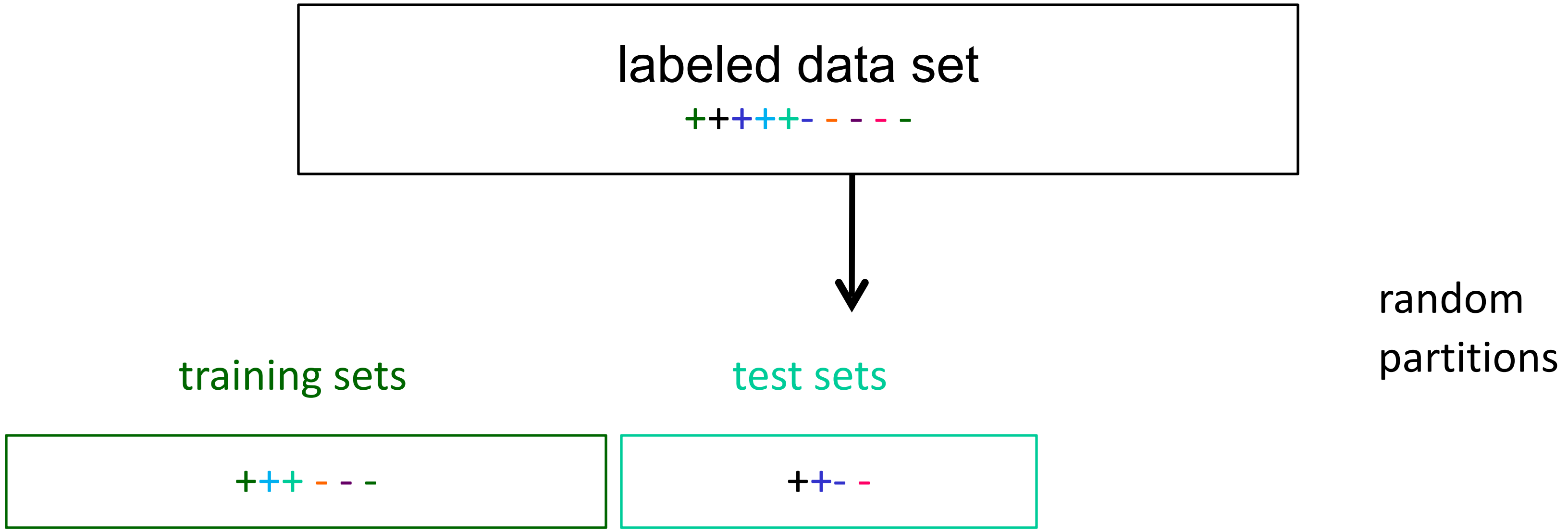
labeled data set

+++++- - - - -

random partitions

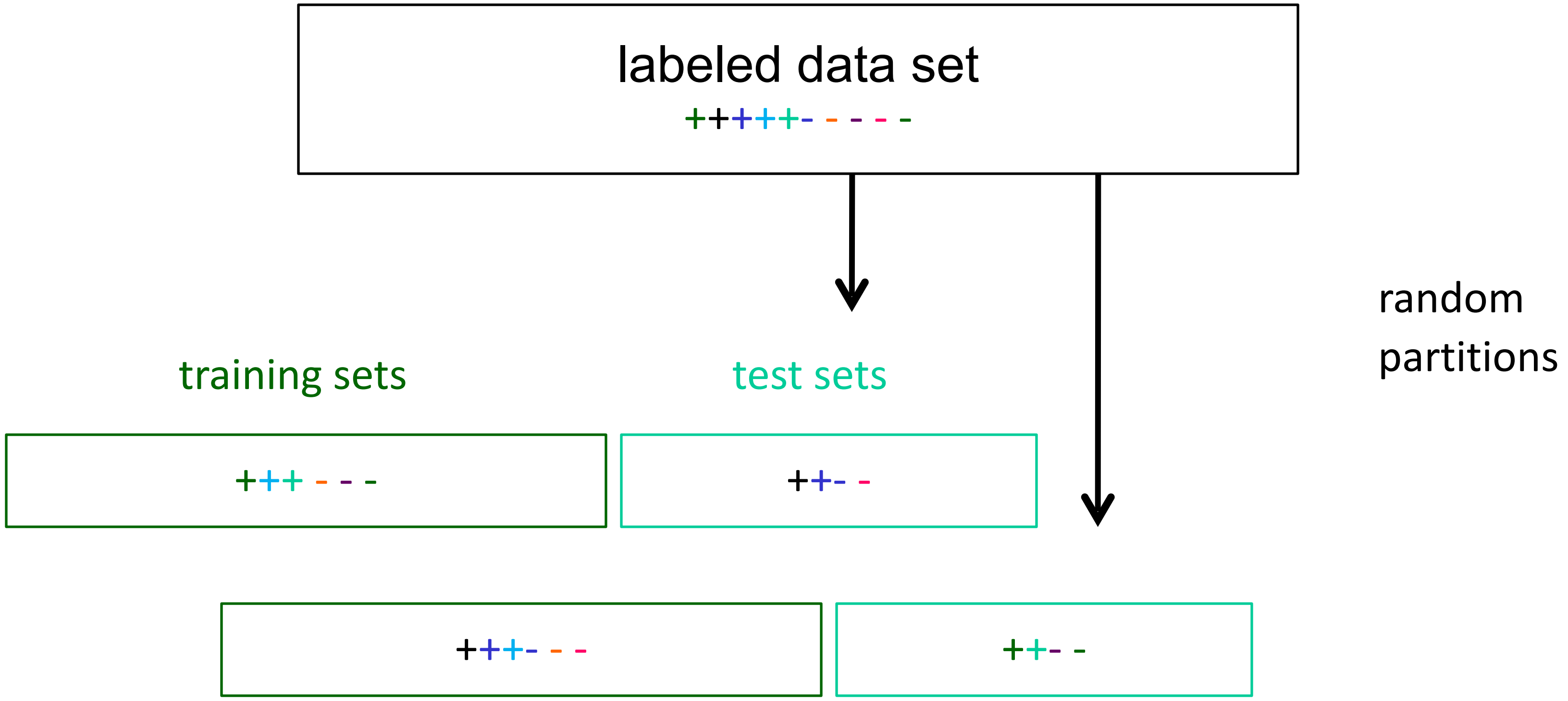
Strategy I: Random Resampling

- Address the second issue by repeatedly randomly partitioning the available data into training and test sets.



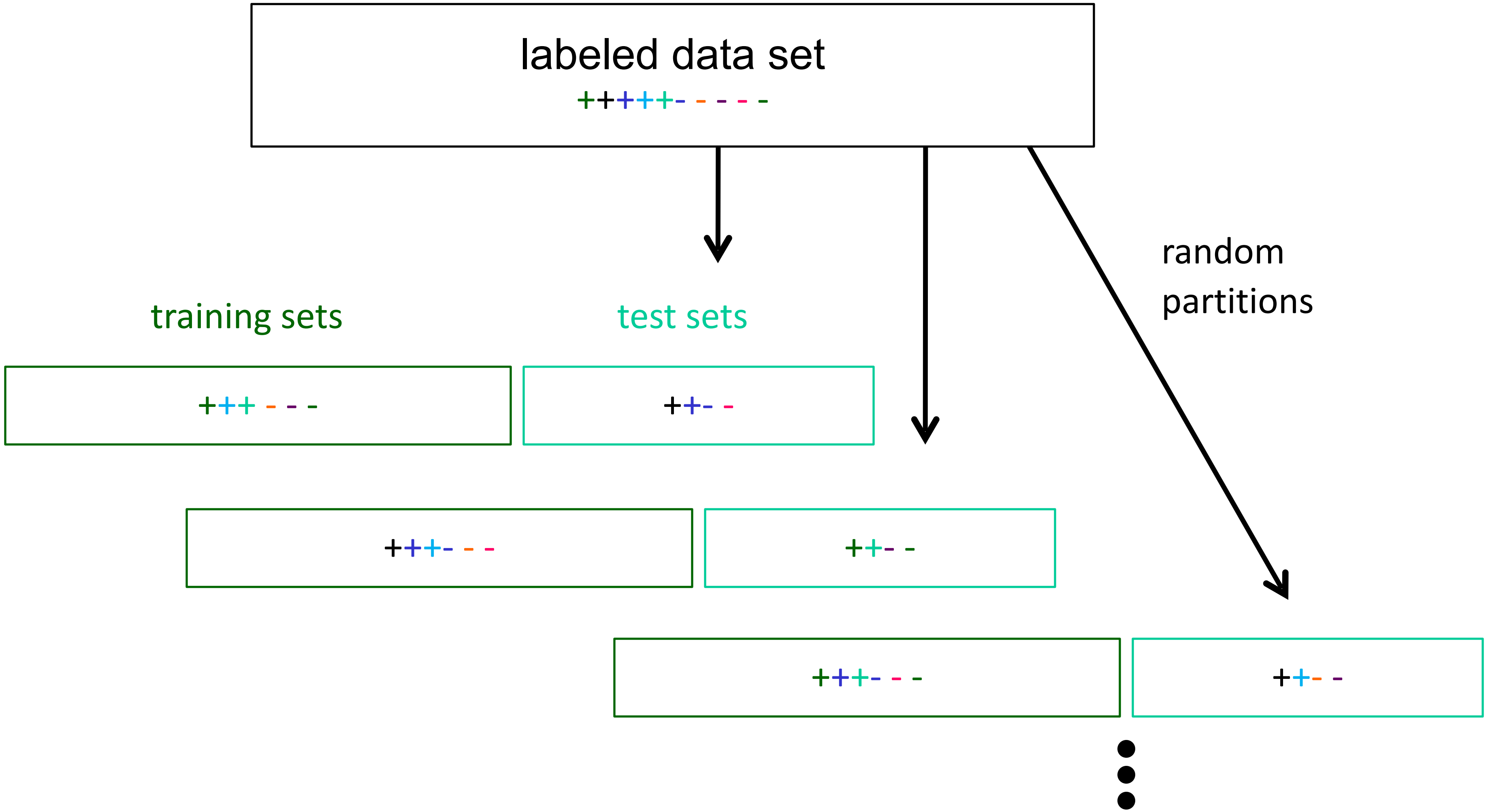
Strategy I: Random Resampling

- Address the second issue by repeatedly randomly partitioning the available data into training and test sets.



Strategy I: Random Resampling

- Address the second issue by repeatedly randomly partitioning the available data into training and test sets.



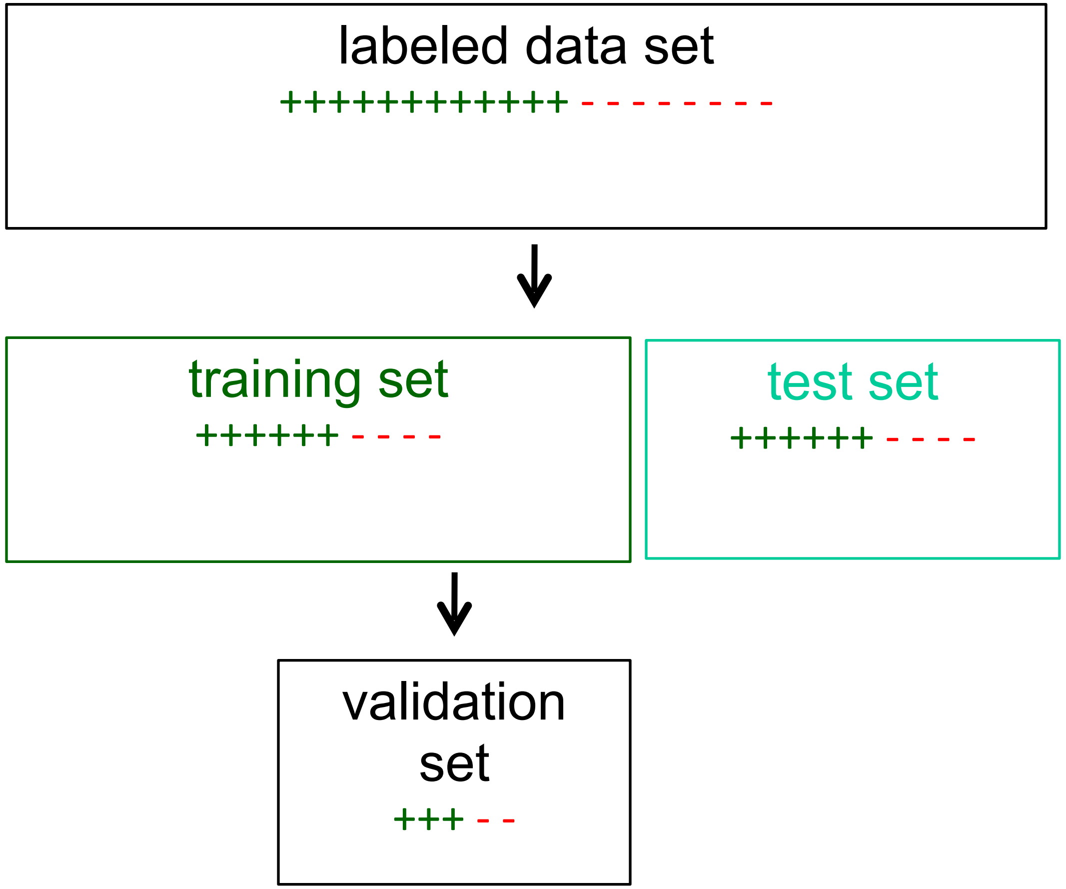
Strategy II: Stratified Sampling

Strategy II: Stratified Sampling

- When randomly selecting training or validation sets, we may want to ensure that **class proportions** are maintained in each selected set

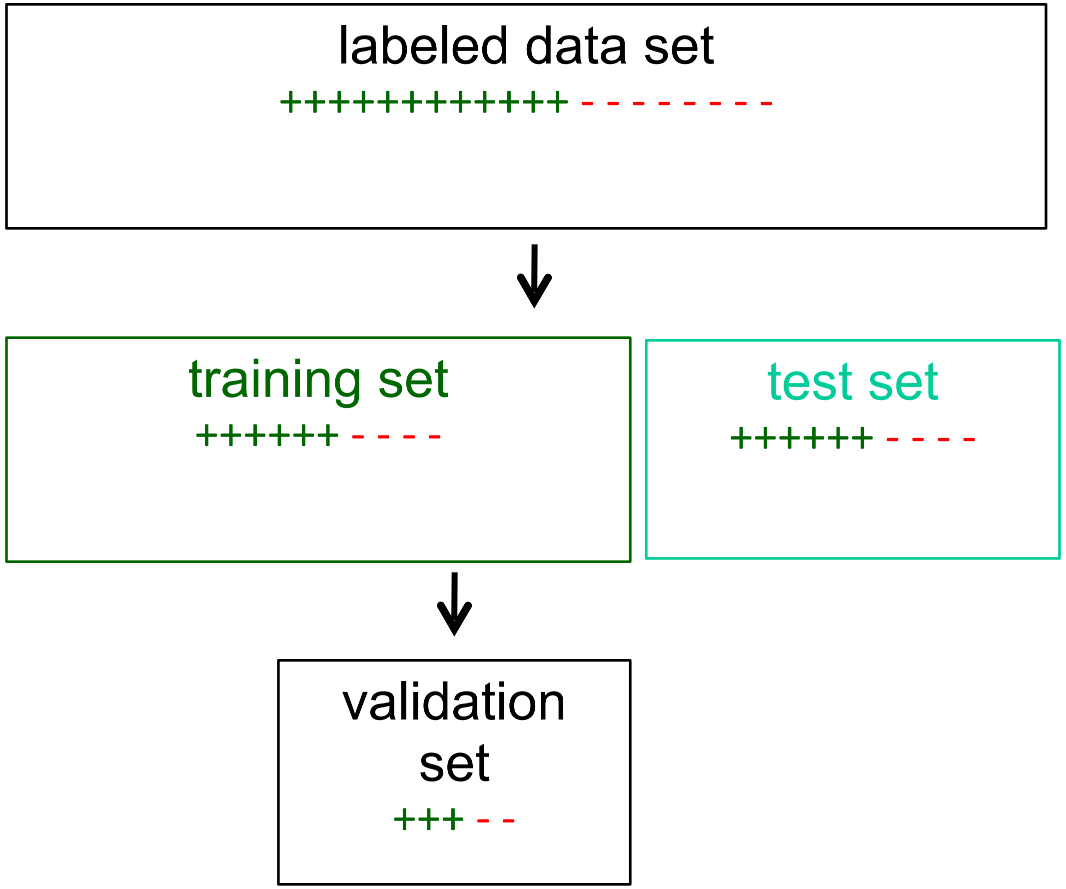
Strategy II: Stratified Sampling

- When randomly selecting training or validation sets, we may want to ensure that **class proportions** are maintained in each selected set



Strategy II: Stratified Sampling

- When randomly selecting training or validation sets, we may want to ensure that **class proportions** are maintained in each selected set



This can be done via stratified sampling: first stratify instances by class, then randomly select instances from each class proportionally.

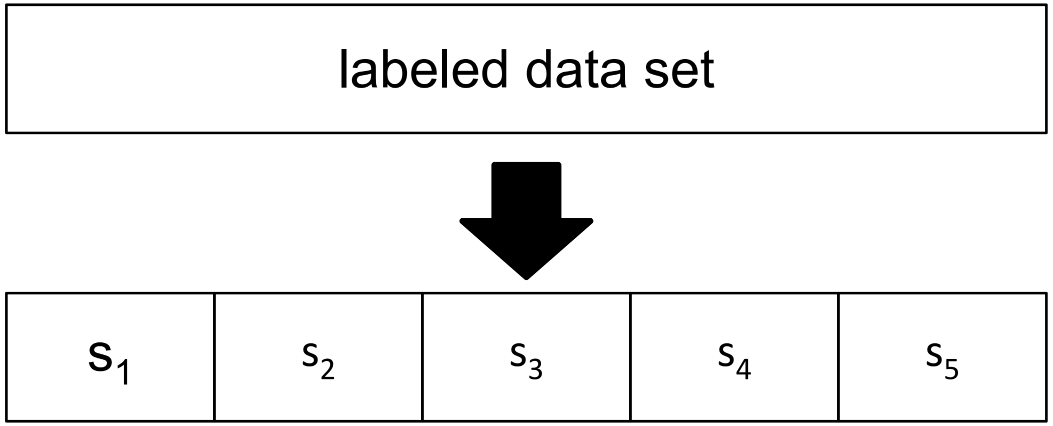
Strategy III: Cross Validation

Strategy III: Cross Validation

Partition data
into *n* subsamples

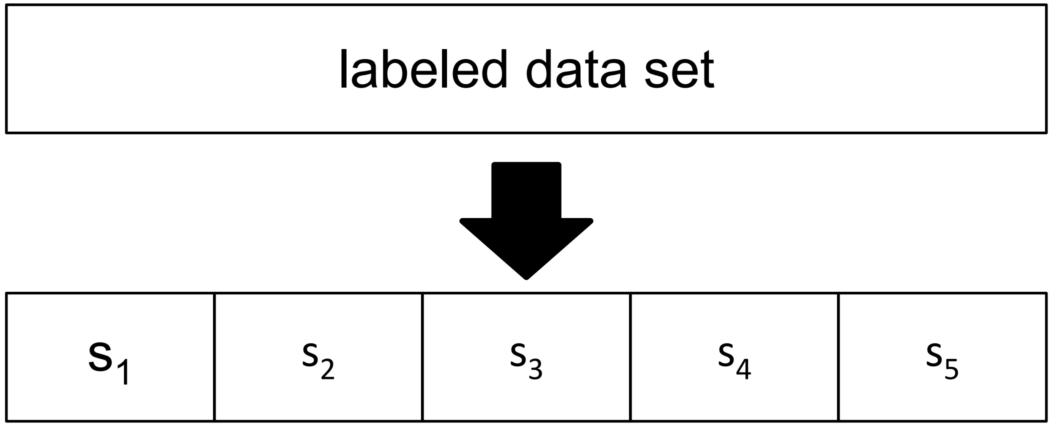
Strategy III: Cross Validation

Partition data
into *n* subsamples



Strategy III: Cross Validation

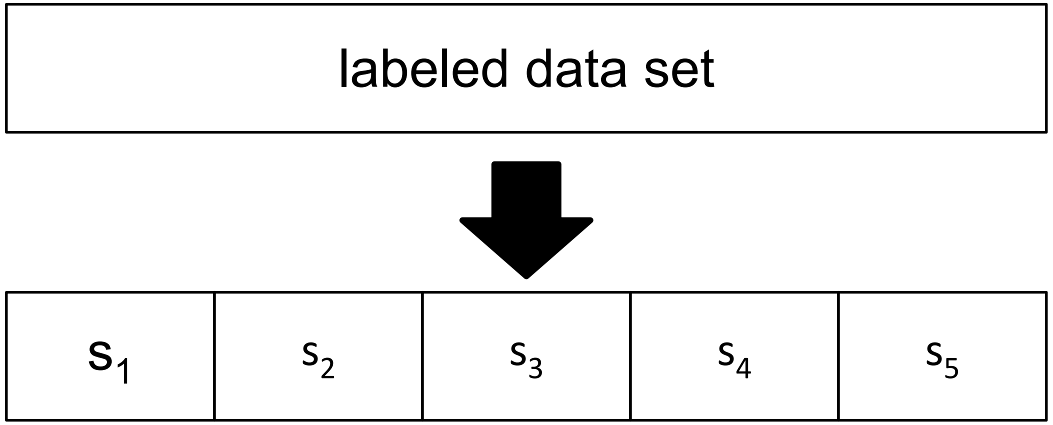
Partition data
into *n* subsamples



Iteratively leave one
subsample out for the
test set, train on the
rest

Strategy III: Cross Validation

Partition data
into *n* subsamples



Iteratively leave one subsample out for the test set, train on the rest

iteration	**train on**	**test on**
1	s_2 s_3 s_4 s_5	s_1
2	s_1 s_3 s_4 s_5	s_2
3	s_1 s_2 s_4 s_5	s_3
4	s_1 s_2 s_3 s_5	s_4
5	s_1 s_2 s_3 s_4	s_5

Strategy III: Cross Validation Example

Strategy III: Cross Validation Example

- Suppose we have 100 instances, and we want to estimate accuracy with cross validation

Strategy III: Cross Validation Example

- Suppose we have 100 instances, and we want to estimate accuracy with cross validation

iteration	**train on**	**test on**	**correct**
1	s_2 s_3 s_4 s_5	s_1	11 / 20
2	s_1 s_3 s_4 s_5	s_2	17 / 20
3	s_1 s_2 s_4 s_5	s_3	16 / 20
4	s_1 s_2 s_3 s_5	s_4	13 / 20
5	s_1 s_2 s_3 s_4	s_5	16 / 20

Strategy III: Cross Validation Example

- Suppose we have 100 instances, and we want to estimate accuracy with cross validation

iteration	**train on**	**test on**	**correct**
1	s_2 s_3 s_4 s_5	s_1	11 / 20
2	s_1 s_3 s_4 s_5	s_2	17 / 20
3	s_1 s_2 s_4 s_5	s_3	16 / 20
4	s_1 s_2 s_3 s_5	s_4	13 / 20
5	s_1 s_2 s_3 s_4	s_5	16 / 20

accuracy = 73/100 = 73%

Strategy II: Cross Validation Tips

Strategy II: Cross Validation Tips

- 10-fold cross validation is common, but smaller values folds are often used when learning takes a lot of time

Strategy II: Cross Validation Tips

- 10-fold cross validation is common, but smaller values folds are often used when learning takes a lot of time

- in *leave-one-out* cross validation, n = # instances

Strategy II: Cross Validation Tips

- 10-fold cross validation is common, but smaller values folds are often used when learning takes a lot of time

- in *leave-one-out* cross validation, *n* = # instances

- in *stratified* cross validation, stratified sampling is used when partitioning the data

Strategy II: Cross Validation Tips

- 10-fold cross validation is common, but smaller values folds are often used when learning takes a lot of time

- in *leave-one-out* cross validation, n = # instances

- in *stratified* cross validation, stratified sampling is used when partitioning the data

- CV makes efficient use of the available data for testing

Strategy II: Cross Validation Tips

- 10-fold cross validation is common, but smaller values folds are often used when learning takes a lot of time

- in *leave-one-out* cross validation, n = # instances

- in *stratified* cross validation, stratified sampling is used when partitioning the data

- CV makes efficient use of the available data for testing

- note that whenever we use multiple training sets, as in CV and random resampling, we are evaluating a <u>learning method (with specific choices)</u> as opposed to an <u>individual learned hypothesis.</u>

Strategy II: Cross Validation Tips

- 10-fold cross validation is common, but smaller values folds are often used when learning takes a lot of time

- in *leave-one-out* cross validation, *n* = # instances

- in *stratified* cross validation, stratified sampling is used when partitioning the data

- CV makes efficient use of the available data for testing

- note that whenever we use multiple training sets, as in CV and random resampling, we are evaluating a <u>learning method (with specific choices)</u> as opposed to an <u>individual learned hypothesis.</u>

- You can use CV for tuning as well!

Learning Curves

Learning Curves

- Accuracy of a method as a function of the train set size?
  - Plot ***learning curves***

Learning Curves

- Accuracy of a method as a function of the train set size?
  - Plot ***learning curves***

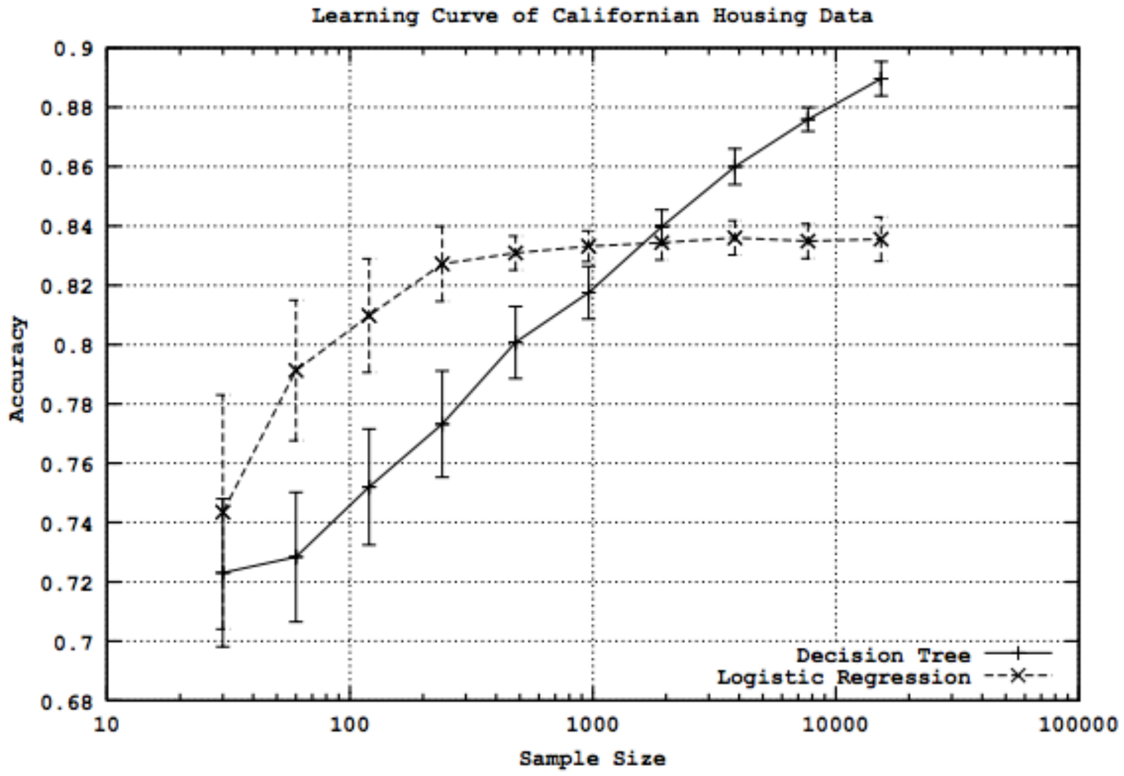


Figure from Perlich et al. *Journal of Machine Learning Research*, 2003

Learning Curves

- Accuracy of a method as a function of the train set size?
  - Plot ***learning curves***

Training/test set partition

- for each sample size s on learning curve
  - (optionally) repeat n times
    - randomly select s instances from training set
    - learn model
    - evaluate model on test set to determine accuracy a
    - plot (s, a) or (s, avg. accuracy and error bars)

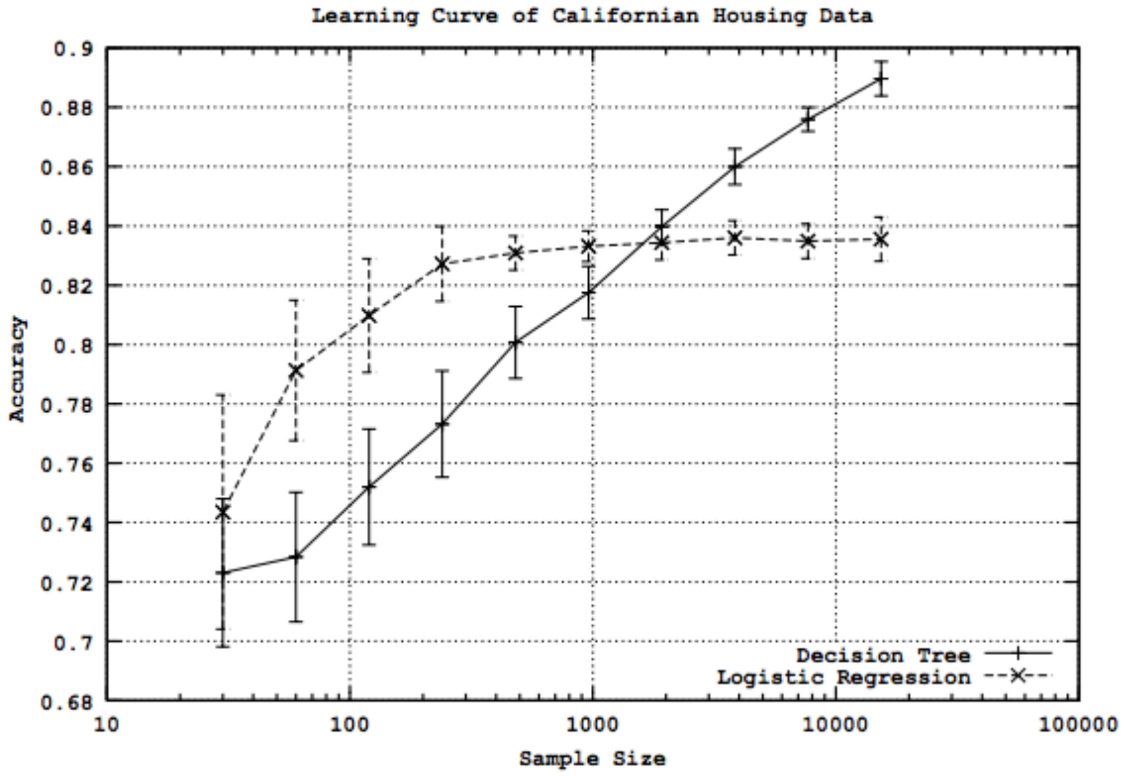


Figure from Perlich et al. *Journal of Machine Learning Research*, 2003

Learning Curves

- Accuracy of a method as a function of the train set size?
  - Plot ***learning curves***

Training/test set partition

- for each sample size s on learning curve
  - (optionally) repeat n times
    - randomly select s instances from training set
    - learn model
    - evaluate model on test set to determine accuracy a
    - plot (s, a) or (s, avg. accuracy and error bars)

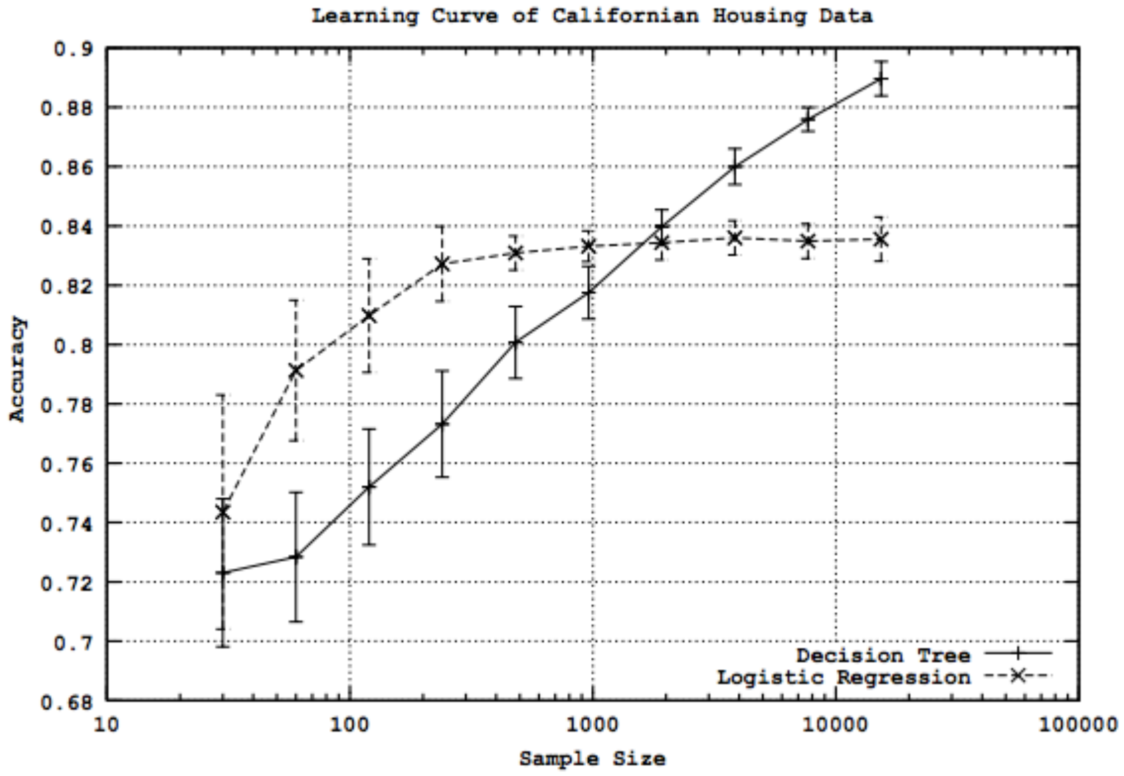


Figure from Perlich et al. *Journal of Machine Learning Research*, 2003

- Why are learning curves useful?

Break & Quiz

Q: Are these statements true or not?
(A) The sample size on the learning curve is the size of test set.
(B) A larger training set would provide a lower variance estimate of the accuracy of a learned model.

1. True, True
2. True, False
3. False, True
4. False, False

Q: Are these statements true or not?
(A) The sample size on the learning curve is the size of test set.
(B) A larger training set would provide a lower variance estimate of the accuracy of a learned model.

1. True, True
2. True, False
3. False, True
4. False, False

(A) The sample size on the learning curve is for training set.
(B) A larger test set rather than a larger training set does so.

Q: Which of the following is NOT true?

1. Class proportions are maintained the same in stratified sampling.
2. In leave-one-out cross validation, the number of partition equals to the number of instances.
3. In cross validation, we are evaluating the performance of an individual learned hypothesis.

Q: Which of the following is NOT true?

1. Class proportions are maintained the same in stratified sampling.
2. In leave-one-out cross validation, the number of partition equals to the number of instances.
3. In cross validation, we are evaluating the performance of an individual learned hypothesis.

In cross validation, we are evaluating a learning method as opposed to a specific individual learned hypothesis.

Outline

- **Wrapping up decision trees**
  - Information gain, stopping criteria
  - Evaluation in decision trees: overfitting, pruning, variations
- **Evaluation: Generalization**
  - Train/test split, random sampling, cross validation
- **Evaluation: Metrics**
  - Confusion matrices, ROC curves, precision/recall

Beyond Accuracy: Confusion Matrices

Beyond Accuracy: Confusion Matrices

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

Beyond Accuracy: Confusion Matrices

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

task: activity recognition from video

actual class

	bend	jack	jump	pjump	run	side	skip	walk	wave1	wave2
bend	100	0	0	0	0	0	0	0	0	0
jack	0	100	0	0	0	0	0	0	0	0
jump	0	0	89	0	0	0	11	0	0	0
pjump	0	0	0	100	0	0	0	0	0	0
run	0	0	0	0	89	0	11	0	0	0
side	0	0	0	0	0	100	0	0	0	0
skip	0	0	0	0	0	0	100	0	0	0
walk	0	0	0	0	0	0	0	100	0	0
wave1	0	0	0	0	0	0	0	0	67	33
wave2	0	0	0	0	0	0	0	0	0	100

predicted class

Confusion Matrices: 2-Class Version

		actual class: positive	actual class: negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

Confusion Matrices: 2-Class Version

		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Confusion Matrices: 2-Class Version

		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

$$\text{error} = 1 - \text{accuracy} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Accuracy: Sufficient?

Accuracy: Sufficient?

Accuracy may not be useful measure in cases where

- There is a large class skew
  - Is 98% accuracy good when 97% of the instances are negative?

- 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

Other Metrics

		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

Other Metrics

		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{true positive rate (recall)} = \frac{\mathrm{TP}}{\text{actual pos}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Other Metrics

		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{true positive rate (recall)} = \frac{\text{TP}}{\text{actual pos}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{false positive rate} = \frac{\text{FP}}{\text{actual neg}} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

If you have probabilities for binary classification, how do you decide on class?

If you have probabilities for binary classification, how do you decide on class?

I.e., classifier returns $\Pr(Y=1|x)$ instead of Y.

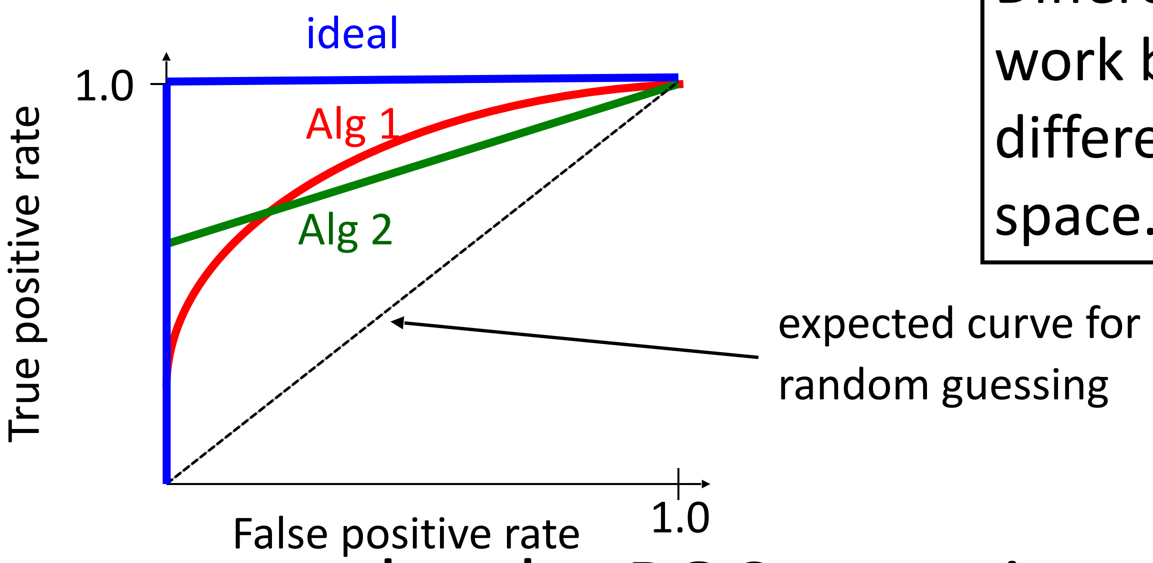
If you have probabilities for binary classification, how do you decide on class?

I.e., classifier returns $Pr(Y=1|x)$ instead of Y.

One solution: choose threshold c and output Y=1 if $Pr(Y=1|x) > c$ else Y=0.

Other Metrics: ROC Curves

- A *Receiver Operating Characteristic* (*ROC*) curve plots the TP-rate vs. the FP-rate as the thresholding value is varied.



Different methods can work better in different parts of ROC space.

Sometimes, area under the ROC curve is used to evaluate a model.

ROC Curves: Algorithm

let $\left(\left(y^{(1)}, c^{(1)}\right) \ldots \left(y^{(m)}, c^{(m)}\right)\right)$ be the test-set instances sorted according to predicted confidence $c^{(i)}$ that each instance is positive

let *num_neg, num_pos* be the number of negative/positive instances in the test set

TP = 0, *FP* = 0

last_TP = 0

for *i* = 1 to *m*

    // find thresholds where there is a pos instance on high side, neg instance on low side

    if (*i* > 1) and ($c^{(i)} \neq c^{(i-1)}$) and ($y^{(i)}$ == neg) and (*TP* > *last_TP*)

        FPR = FP / num_neg, TPR = TP / num_pos

        output (*FPR, TPR*) coordinate

        last_TP = TP

    if $y^{(i)}$ == pos

        ++TP

    else

        ++FP

FPR = FP / num_neg, TPR = TP / num_pos

output (*FPR, TPR*) coordinate

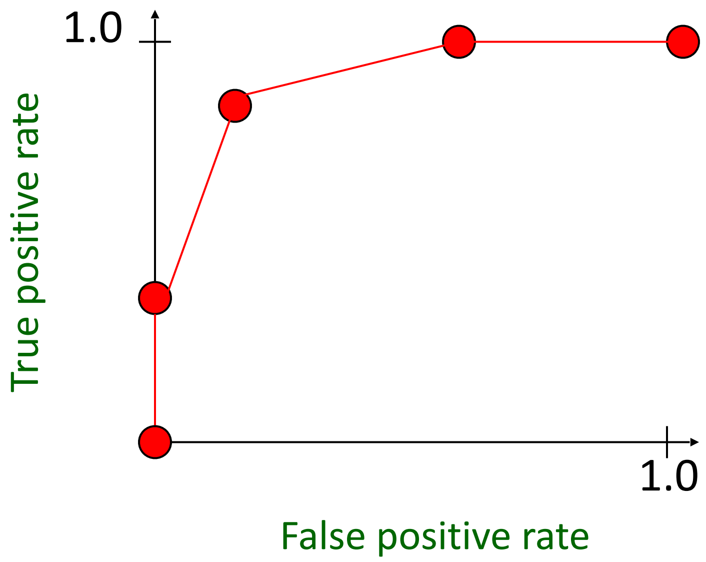
ROC Curves: Plotting

ROC Curves: Plotting

instance	threshold		correct class
Ex 9	.99		+
Ex 7	.98	TPR= 2/5, FPR= 0/5	+
Ex 1	.72		-
Ex 2	.70		+
Ex 6	.65	TPR= 4/5, FPR= 1/5	+
Ex 10	.51		-
Ex 3	.39		-
Ex 5	.24	TPR= 5/5, FPR= 3/5	+
Ex 4	.11		-
Ex 8	.01	TPR= 5/5, FPR= 5/5	-

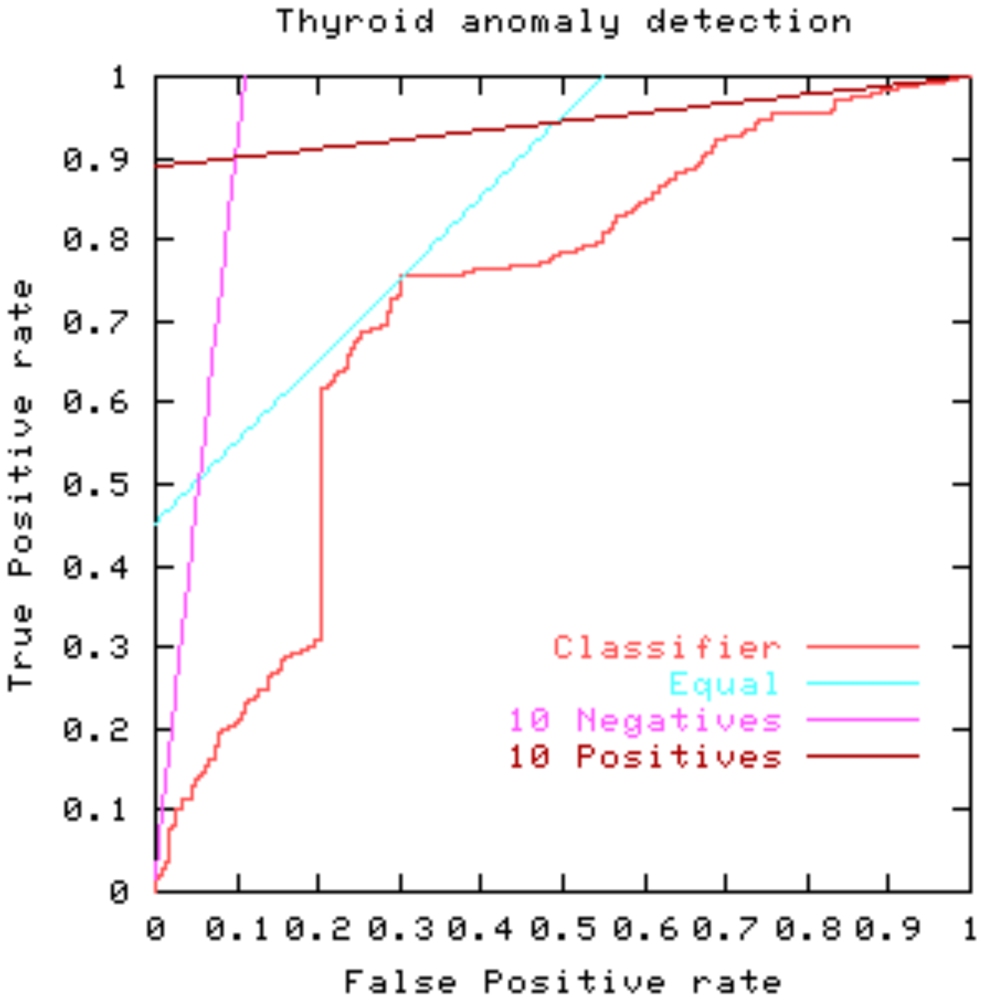
ROC Curves: Plotting

instance	threshold		correct class
Ex 9	.99		+
Ex 7	.98	TPR= 2/5, FPR= 0/5	+
Ex 1	.72		-
Ex 2	.70		+
Ex 6	.65	TPR= 4/5, FPR= 1/5	+
Ex 10	.51		-
Ex 3	.39		-
Ex 5	.24	TPR= 5/5, FPR= 3/5	+
Ex 4	.11		-
Ex 8	.01	TPR= 5/5, FPR= 5/5	-



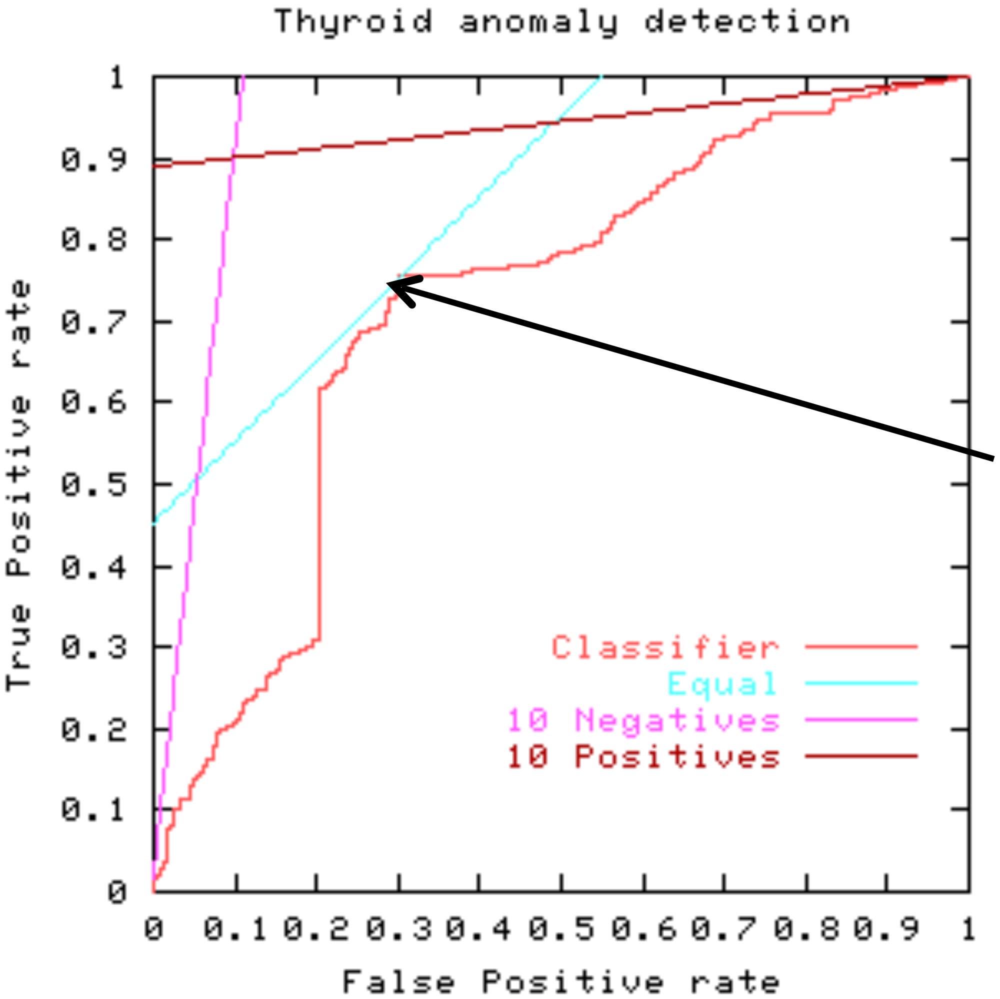
ROC Curves: Misclassification Cost

- The best operating point depends on relative cost of FN and FP misclassifications



ROC Curves: Misclassification Cost

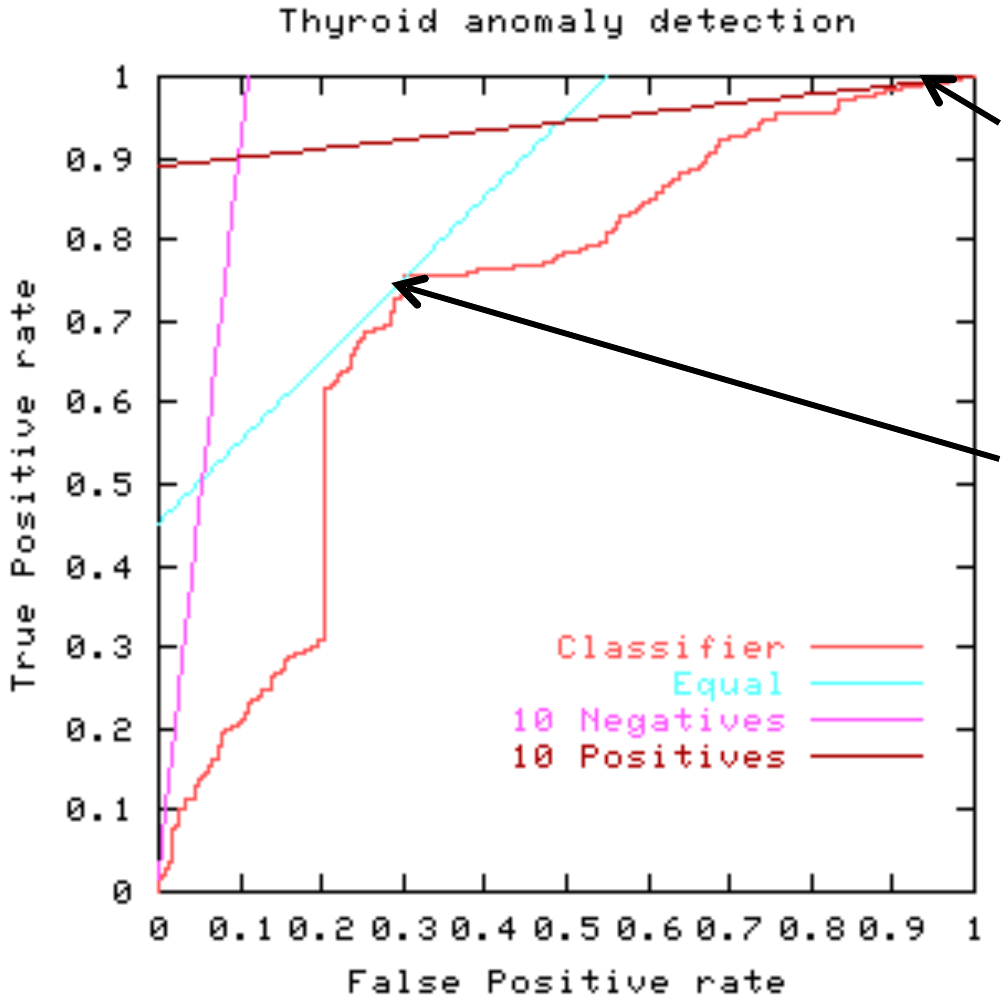
- The best operating point depends on relative cost of FN and FP misclassifications



best operating point when
cost of misclassifying positives and
negatives is equal

ROC Curves: Misclassification Cost

- The best operating point depends on relative cost of FN and FP misclassifications

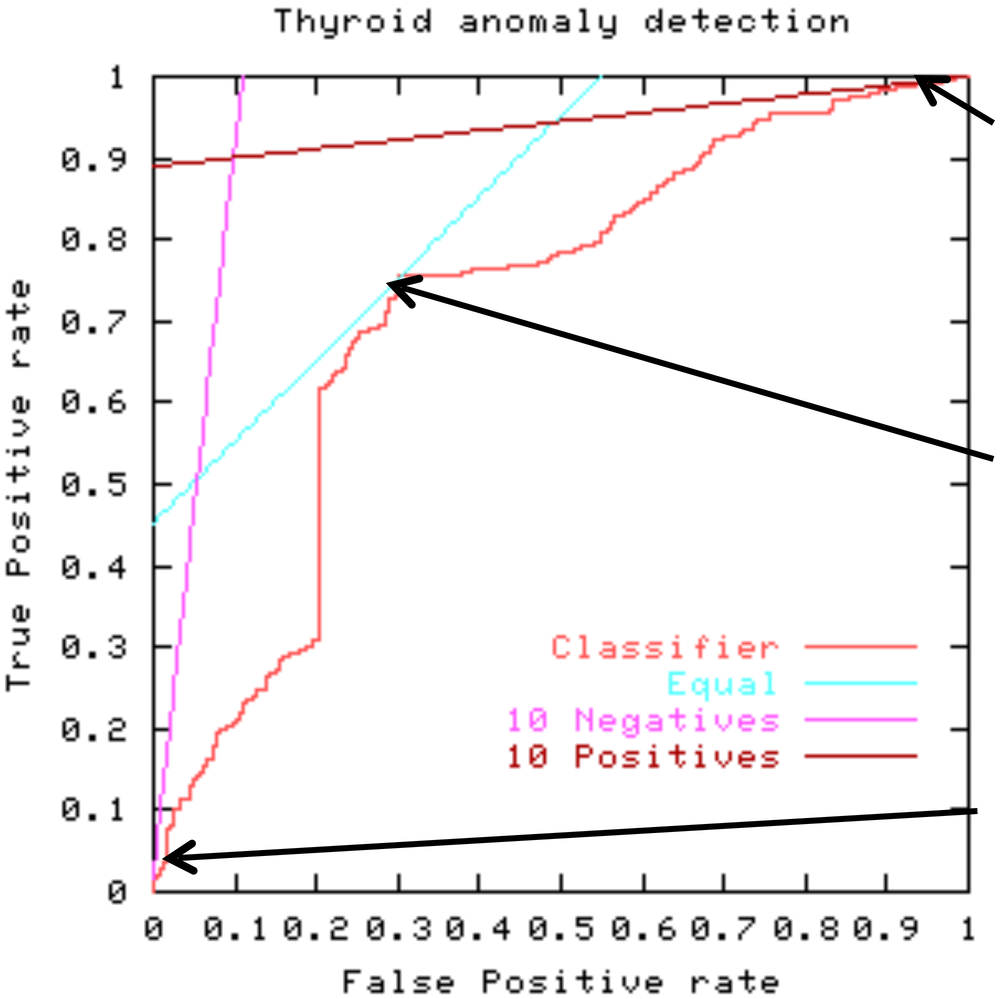


best operating point when
FN costs 10× FP

best operating point when
cost of misclassifying positives and
negatives is equal

ROC Curves: Misclassification Cost

- The best operating point depends on relative cost of FN and FP misclassifications



best operating point when
FN costs 10× FP

best operating point when
cost of misclassifying positives and negatives is equal

best operating point when
FP costs 10× FN

Other Metrics: Precision

		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

Other Metrics: Precision

		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

Other Metrics: Precision

		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{recall (TP rate)} = \frac{\text{TP}}{\text{actual pos}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Other Metrics: Precision

predicted class \ actual class	positive	negative
positive	true positives (TP)	false positives (FP)
negative	false negatives (FN)	true negatives (TN)

$$\text{recall (TP rate)} = \frac{\text{TP}}{\text{actual pos}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{precision (positive predictive value)} = \frac{\text{TP}}{\text{predicted pos}} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

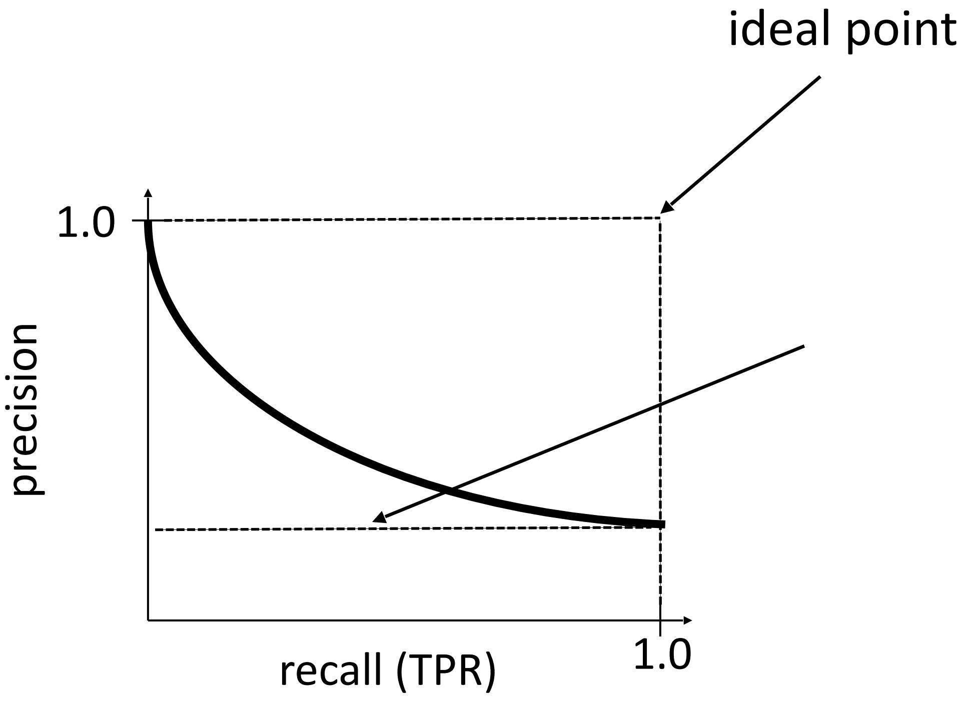
Other Metrics: Precision/Recall Curve

Other Metrics: Precision/Recall Curve

- A *precision/recall curve* (TP-rate): threshold on the confidence of an instance being positive is varied

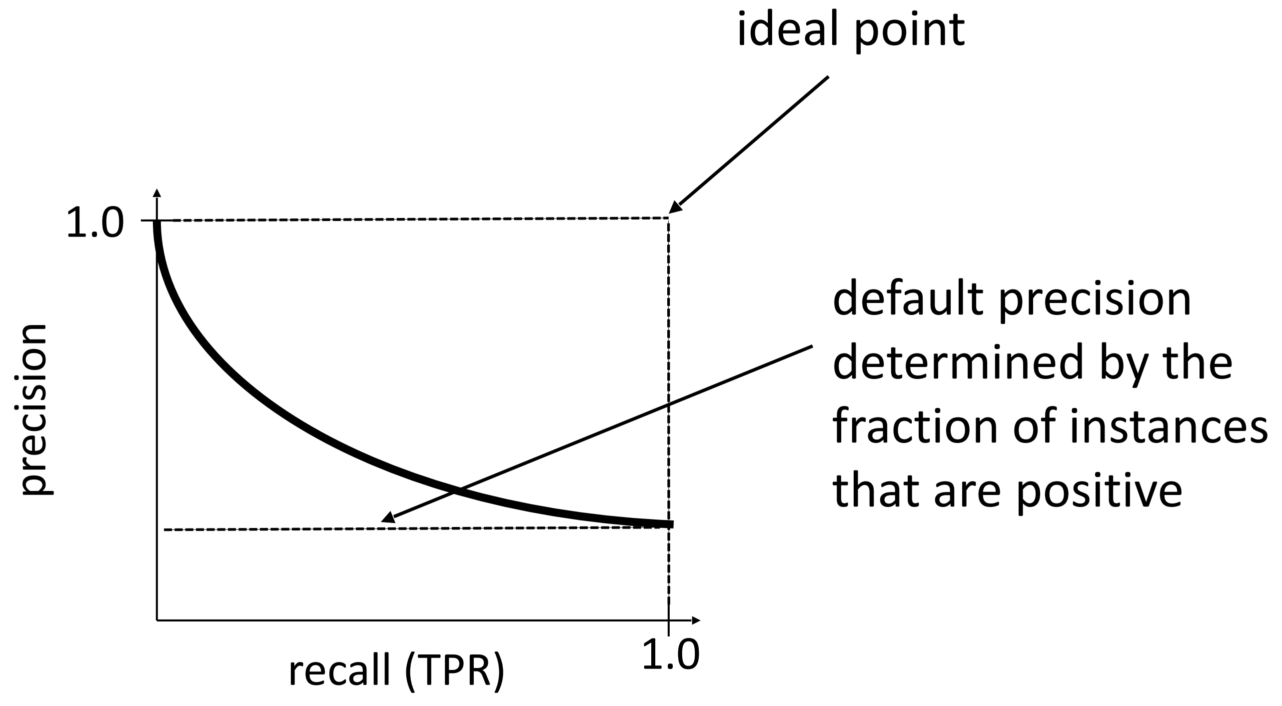
Other Metrics: Precision/Recall Curve

- A *precision/recall curve* (TP-rate): threshold on the confidence of an instance being positive is varied



Other Metrics: Precision/Recall Curve

- A *precision/recall curve* (TP-rate): threshold on the confidence of an instance being positive is varied



ROC vs. PR curves

ROC vs. PR curves

Both

ROC vs. PR curves

Both

- Allow predictive performance to be assessed at various levels of thresholding

ROC vs. PR curves

Both

- Allow predictive performance to be assessed at various levels of thresholding
- Assume binary classification tasks

ROC vs. PR curves

Both

- Allow predictive performance to be assessed at various levels of thresholding
- Assume binary classification tasks
- Sometimes summarized by calculating *area under the curve*

ROC vs. PR curves

Both

- Allow predictive performance to be assessed at various levels of thresholding
- Assume binary classification tasks
- Sometimes summarized by calculating *area under the curve*

ROC curves

ROC vs. PR curves

Both

- Allow predictive performance to be assessed at various levels of thresholding
- Assume binary classification tasks
- Sometimes summarized by calculating *area under the curve*

ROC curves

- Insensitive to changes in class distribution (ROC curve does not change if the proportion of positive and negative instances in the test set are varied)

ROC vs. PR curves

Both

- Allow predictive performance to be assessed at various levels of thresholding
- Assume binary classification tasks
- Sometimes summarized by calculating *area under the curve*

ROC curves

- Insensitive to changes in class distribution (ROC curve does not change if the proportion of positive and negative instances in the test set are varied)
- Can identify optimal classification thresholds for tasks with differential misclassification costs

ROC vs. PR curves

Both

- Allow predictive performance to be assessed at various levels of thresholding
- Assume binary classification tasks
- Sometimes summarized by calculating *area under the curve*

ROC curves

- Insensitive to changes in class distribution (ROC curve does not change if the proportion of positive and negative instances in the test set are varied)
- Can identify optimal classification thresholds for tasks with differential misclassification costs

Precision/recall curves

ROC vs. PR curves

Both

- Allow predictive performance to be assessed at various levels of thresholding
- Assume binary classification tasks
- Sometimes summarized by calculating *area under the curve*

ROC curves

- Insensitive to changes in class distribution (ROC curve does not change if the proportion of positive and negative instances in the test set are varied)
- Can identify optimal classification thresholds for tasks with differential misclassification costs

Precision/recall curves

- Show <u>the fraction of predictions</u> that are false positives

ROC vs. PR curves

Both

- Allow predictive performance to be assessed at various levels of thresholding
- Assume binary classification tasks
- Sometimes summarized by calculating *area under the curve*

ROC curves

- Insensitive to changes in class distribution (ROC curve does not change if the proportion of positive and negative instances in the test set are varied)
- Can identify optimal classification thresholds for tasks with differential misclassification costs

Precision/recall curves

- Show the fraction of predictions that are false positives
- Well suited for tasks with lots of negative instances

Confidence Intervals

Confidence Intervals

- Back to looking at accuracy on new data.

Confidence Intervals

- Back to looking at accuracy on new data.
- **Scenario**:
  - For some model h, a test set S with n samples
  - We have h producing r errors out of n.
  - Our estimate of the error rate: $\text{error}_S(h) = r/n$

Confidence Intervals

- Back to looking at accuracy on new data.
- **Scenario**:
  - For some model *h*, a test set S with *n* samples
  - We have *h* producing *r* errors out of *n*.
  - Our estimate of the error rate: $\text{error}_S(h) = r/n$

- With C% probability, true error is in interval

Confidence Intervals

- Back to looking at accuracy on new data.
- **Scenario**:
  - For some model *h*, a test set S with *n* samples
  - We have *h* producing *r* errors out of *n*.
  - Our estimate of the error rate: $\text{error}_S(h) = r/n$

- With C% probability, true error is in interval

Confidence Intervals

- Back to looking at accuracy on new data.
- **Scenario**:
  - For some model *h*, a test set S with *n* samples
  - We have *h* producing *r* errors out of *n*.
  - Our estimate of the error rate: $\mathrm{error}_S(h) = r/n$

- With C% probability, true error is in interval

$$error_S(h) \pm z_C \sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

  - z_C depends on C.

Thanks Everyone!

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, Pedro Domingos, Jerry Zhu, Yingyu Liang, Volodymyr Kuleshov