CMSC 471: Machine Learning

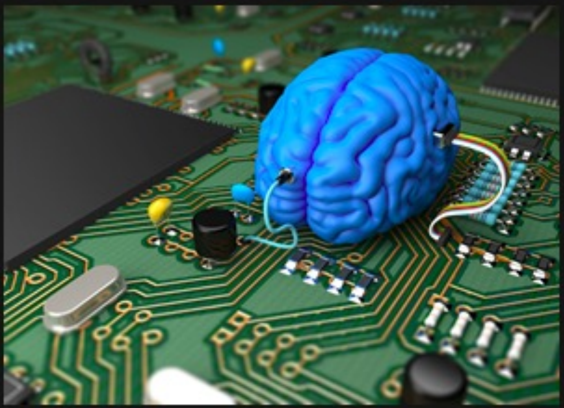
KMA Solaiman – ksolaima@umbc.edu

Some slides courtesy Tim Finin and Frank Ferraro

ML FOR USERS

Deep Learning

What society thinks I do



What my friends think I do

What other computer scientists think I do

What mathematicians think I do

What I think I do

```
from [redacted] import *
```

keras
torch

What I actually do

Toolkit Basics

- Machine learning involves working with data
  - analyzing, manipulating, transforming, …
- More often than not, it’s numeric or has a natural numeric representation
- Natural language text is an exception, but this too can have a numeric representation
- A common data model is as a N-dimensional matrix or tensor
- These are supported in Python via libraries

Typical Python Libraries

numpy, scipy

- Basic mathematical libraries for dealing with matrices and scientific/mathematical functions

pandas, matplotlib

- Libraries for data science & plotting

sklearn (scikit-learn)

- A whole bunch of implemented classifiers

torch (pytorch) and tensorflow

- Frameworks for building neural networks

Lots of documentation available for all of these online!

What is Numpy?

- NumPy supports features needed for ML
  - Typed N-dimensional arrays (matrices/tensors)
  - Fast numerical computations (matrix math)
  - High-level math functions
- Python does numerical computations slowly and lacks an efficient matrix representation
- 1000 x 1000 matrix multiply
  - **Python triple loop takes > 10 minutes!**
  - **Numpy takes ~0.03 seconds**

NumPy Arrays Can Represent ..

Structured lists of numbers

- **Vectors**
- **Matrices**
- Images
- Tensors
- Convolutional Neural Networks

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

NumPy Arrays Can Represent ..

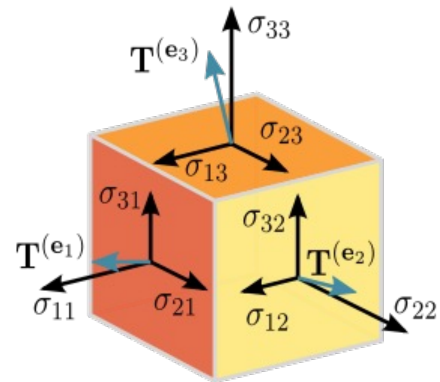
Structured lists of numbers

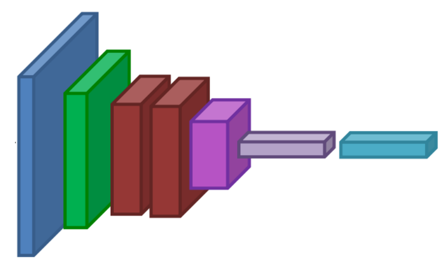
- Vectors
- Matrices
- **Images**
- Tensors
- Convolutional Neural Networks

NumPy Arrays Can Represent ..

Structured lists of numbers

- Vectors
- Matrices
- Images
- **Tensors**
- **Convolutional Neural Networks**





NumPy Arrays, Basic Properties

```
>>> import numpy as np
>>> a= np.array([[1,2,3],[4,5,6]],dtype=np.float32)
>>> print(a.ndim, a.shape, a.dtype)
2  (2, 3)  float32
>> print(a)
[[1. 2. 3.]
 [4. 5. 6.]]
```

Arrays:

1. Can have any number of dimensions, including zero (a scalar)
2. Are **typed**: np.uint8, np.int64, np.float32, np.float64
3. Are **dense:** each element of array exists and has the same type

NumPy Array Indexing, Slicing

```
a[0,0]  # top-left element
a[0,-1] # first row, last column
a[0,:]  # first row, all columns
a[:,0]  # first column, all rows
a[0:2,0:2]  # 1st 2 rows, 1st 2 columns
```

Notes:

- Zero-indexing
- Multi-dimensional indices are comma-separated)
- Python notation for slicing

SciPy

- SciPy builds on the NumPy array object
- Adds additional mathematical functions and *sparse arrays*
- **Sparse array:** one where most elements = 0
- An efficient representation only implicitly encodes the non-zero values
- Access to a missing element returns 0

SciPy sparse array use case

- NumPy and SciPy arrays are numeric
- We can represent a document's content by a vector of features
- Each feature is a possible word
- A feature's value might be any of:
  - TF: number of times it occurs in the document;
  - TF-IDF: … normalized by how common the word is
  - and maybe normalized by document length …

SciPy sparse array use case

- Maybe only model 50k most frequent words found in a document collection, ignoring others
- Assign each unique word an index (e.g., dog:137)
  - Build python dict **w** from vocabulary, so w['dog']=137
- The sentence "the dog chased the cat"
  - Would be a *numPy vector* of length 50,000
  - Or a *sciPy sparse vector* of length 4
- An 800-word news article may only have 100 unique words; The Hobbit has about 8,000

More on SciPy

SciPy.org

SciPy.org | Docs | SciPy v1.4.1 Reference Guide

SciPy Tutorial

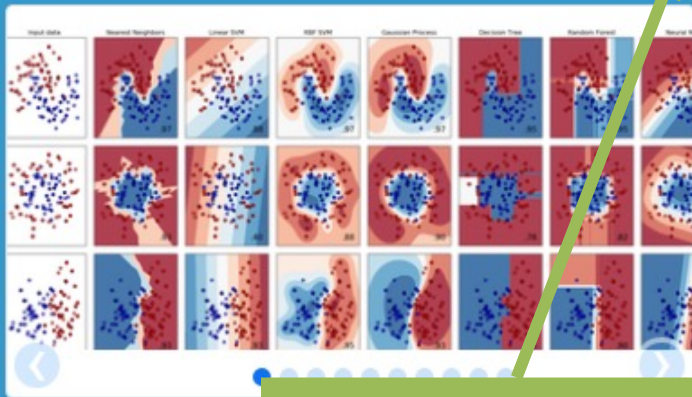
- Introduction
- Basic functions
- Special functions (scipy.special)
- Integration (scipy.integrate)
- Optimization (scipy.optimize)
- Interpolation (scipy.interpolate)
- Fourier Transforms (scipy.fft)
- Signal Processing (scipy.signal)
- Linear Algebra (scipy.linalg)
- Sparse eigenvalue problems with ARPACK
- Compressed Sparse Graph Routines (scipy.sparse.csgraph)
- Spatial data structures and algorithms (scipy.spatial)
- Statistics (scipy.stats)
- Multidimensional image processing (scipy.ndimage)
- File IO (scipy.io)

See the SciPy tutorial Web pages

scikit learn

Home Installation Documentation Examples

https://sklearn.org/



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Documentation online

Many tutorials

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, ... — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, ... — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, ... — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics. — Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction. — Examples

How easy is this?

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = load_iris(return_X_y=True)
```

features on data

labels

```
>>> clf = LogisticRegression(random_state=0).fit(X, y)
```

DATA & EVALUATION

http://archive.ics.uci.edu/ml

UCI Machine Learning Repository

archive.ics.uci.edu/ml/

About Citation Policy Donate a Data Set Contact

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

Search

View ALL Data Sets

Welcome to the UC Irvine Machine Learning Repository!

We currently maintain 233 data sets as a service to the machine learning community. You may **view all data sets** through our searchable interface. Our old web site is still available, for those who prefer the old format. For a general overview of the Repository, please visit our About page. For information about citing data sets in publications, please read our citation policy. If you wish to donate a data set, please consult our donation policy. For any other questions, feel free to contact the Repository librarians. We have also set up a mirror site for the Repository.

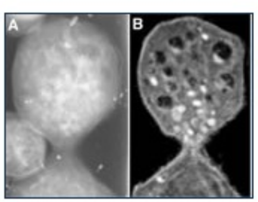
Supported By: In Collaboration With: Rexa.info

233 data sets

Latest News:

- **2010-03-01:** Note from donor regarding Netflix data
- **2009-10-16:** Two new data sets have been added.
- **2009-09-14:** Several data sets have been added.
- **2008-07-23:** Repository mirror has been set up.
- **2008-03-24:** New data sets have been added!
- **2007-06-25:** Two new data sets have been added: UJI Pen Characters, MAGIC Gamma Telescope
- **2007-04-13:** Research papers that cite the repository have been associated to specific data sets.

Featured Data Set: Yeast



Task: Classification
Data Type: Multivariate
 # Attributes: 8
 # Instances: 1484

Predicting the Cellular Localization Sites of Proteins

Newest Data Sets:

- **2012-10-21:** QtyT40I10D100K
- **2012-10-19:** Legal Case Reports
- **2012-09-29:** seeds
- **2012-08-30:** Individual household electric power consumption
- **2012-08-15:** Northix
- **2012-08-06:** PAMAP2 Physical Activity Monitoring
- **2012-08-04:** Restaurant & consumer data
- **2012-08-03:** CNAE-9

Most Popular Data Sets (hits since 2007):

- **386214:** Iris
- **272233:** Adult
- **237503:** Wine
- **195947:** Breast Cancer Wisconsin (Diagnostic)
- **182423:** Car Evaluation
- **151635:** Abalone
- **135419:** Poker Hand
- **113024:** Forest Fires

UCI Machine Learning Repository: Zoo Data Set

http://archive.ics.uci.edu/ml/datasets/Zoo

About Citation Policy Donate a Data Set Contact

Search

Repository Web

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

View ALL Data Sets

Zoo Data Set

Download: Data Folder, Data Set Description

Abstract: Artificial, 7 classes of animals

http://archive.ics.uci.edu/ml/datasets/Zoo

Data Set Characteristics:	Multivariate	**Number of Instances:**	101	**Area:**	Life
Attribute Characteristics:	Categorical, Integer	**Number of Attributes:**	17	**Date Donated**	1990-05-15
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	18038

Zoo data

animal name: string
hair: Boolean
feathers: Boolean
eggs: Boolean
milk: Boolean
airborne: Boolean
aquatic: Boolean
predator: Boolean
toothed: Boolean
backbone: Boolean
breathes: Boolean
venomous: Boolean
fins: Boolean
legs: {0,2,4,5,6,8}
tail: Boolean
domestic: Boolean
catsize: Boolean
type: {mammal, fish, bird, shellfish, insect, reptile, amphibian}

101 examples

aardvark,1,0,0,1,0,0,1,1,1,1,0,0,4,0,0,1,mammal
antelope,1,0,0,1,0,0,0,1,1,1,0,0,4,1,0,1,mammal
bass,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,fish
bear,1,0,0,1,0,0,1,1,1,1,0,0,4,0,0,1,mammal
boar,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1,mammal
buffalo,1,0,0,1,0,0,0,1,1,1,0,0,4,1,0,1,mammal
calf,1,0,0,1,0,0,0,1,1,1,0,0,4,1,1,1,mammal
carp,0,0,1,0,0,1,0,1,1,0,0,1,0,1,1,0,fish
catfish,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,fish
cavy,1,0,0,1,0,0,0,1,1,1,0,0,4,0,1,0,mammal
cheetah,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1,mammal
chicken,0,1,1,0,1,0,0,0,1,1,0,0,2,1,1,0,bird
chub,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,fish
clam,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,shellfish
crab,0,0,1,0,0,1,1,0,0,0,0,0,4,0,0,0,shellfish
…

Defining Appropriate Features

Feature functions help extract useful features (characteristics) of the data

They turn *data* into *numbers*

Features that are not 0 are said to have **fired**

Defining Appropriate Features

Feature functions help extract useful features (characteristics) of the data

They turn *data* into *numbers*

Features that are not 0 are said to have fired

Often binary-valued (0 or 1), but can be real-valued

sklearn example
(in-class, live coding)

Zoo example

AIMA Code Example

```
aima-python> python
>>> from learning import *
>>> zoo
<DataSet(zoo): 101 examples, 18 attributes>
>>> dt = DecisionTreeLearner()
>>> dt.train(zoo)
>>> dt.predict(['shark',0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0])
'fish'
>>> dt.predict(['shark',0,0,0,0,0,1,1,1,1,0,0,1,0,1,0,0])
'mammal'
```

Central Question: How Well Are We Doing?

Classification

- Precision, Recall, F1
- Accuracy
- Log-loss
- ROC-AUC
- …

Regression

- (Root) Mean Square Error
- Mean Absolute Error
- …

Clustering

- Mutual Information
- V-score
- …

the ***task****: what* kind *of problem are you solving?*

Evaluation Metrics

Classification

- Precision, Recall, F1
- Accuracy
- Log-loss
- ROC-AUC
- …

Regression

- (Root) Mean Square Error
- Mean Absolute Error
- …

Clustering

- Mutual Information
- V-score
- …

the ***task****: what* kind *of problem are you solving?*

This does not have to be the same thing as the loss function you optimize

Evaluation methodology (1)

Standard methodology:

1. Collect large set of examples with correct classifications (aka ground truth data)
2. Randomly divide collection into two disjoint sets: ***training*** and ***test*** *(e.g., via a 90-10% split)*
3. Apply learning algorithm to **training** set giving hypothesis H
4. Measure performance of H on the held-out **test** set

Evaluation methodology (2)

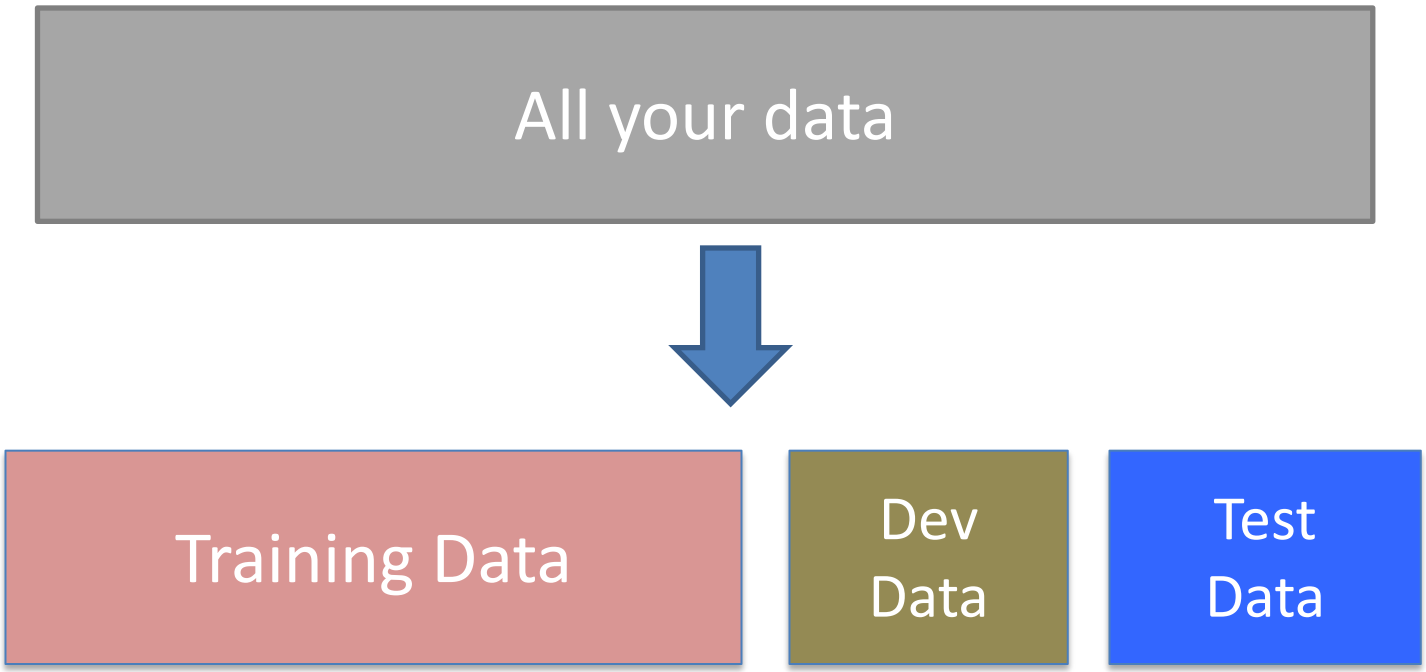
- **Important: keep the training and test sets disjoint!**
- Study efficiency & robustness of algorithm: repeat steps 2-4 for different training sets & training set sizes
- On modifying algorithm, restart with step 1 to avoid evolving algorithm to work well on just this collection

Evaluation methodology (2)

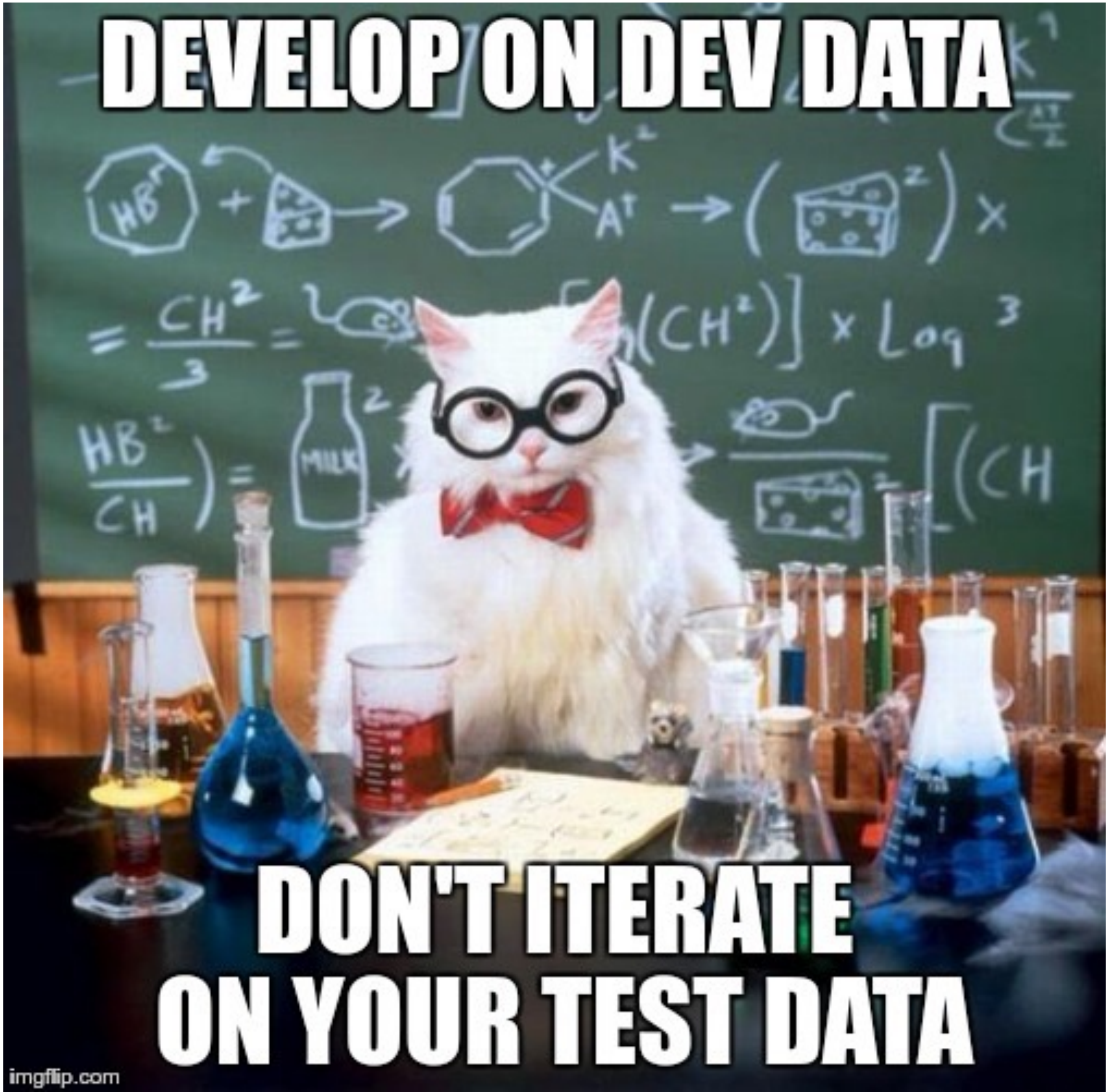
- **Important: keep the training and test sets disjoint!**
- Study efficien ... f algorithm: repeat steps ... ining sets & training set ...
- On modifying ... with step 1 to avoid evolving a ... work well on just this collection

But Not supposed to see test data to improve model

Experimenting with Machine Learning Models



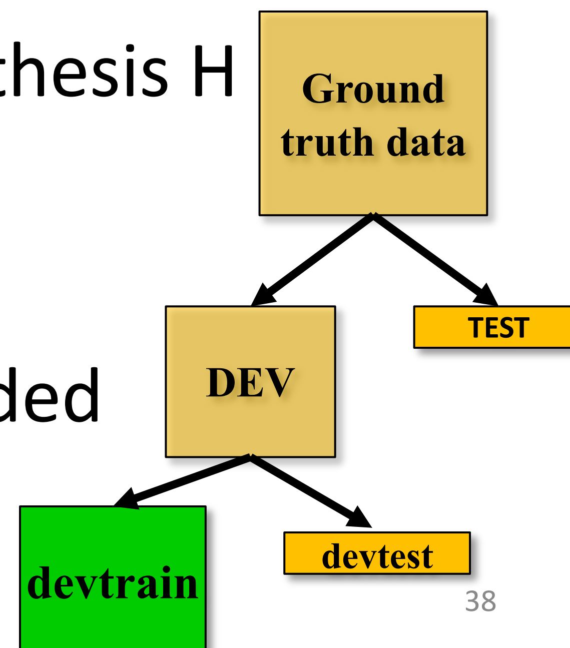
Rule #1


DEVELOP ON DEV DATA
DON'T ITERATE
ON YOUR TEST DATA
imgflip.com

Evaluation methodology (3)

Common variation on methodology:

1. Collect set of examples with correct classifications
2. Randomly divide it into two disjoint sets: ***development*** & ***test****;* further divide development into ***devtrain*** & ***devtest***
3. Apply ML to *devtrain,* creating hypothesis H
4. Measure performance of H w.r.t. *devtest* data
5. Modify approach, repeat 3-4 as needed
6. Final test on *test* data

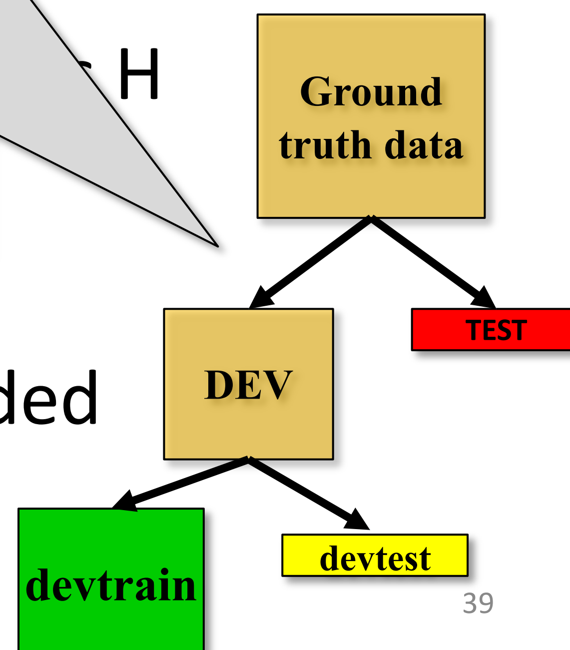


Evaluation methodology (4)

- Only **devtest** data used for evaluation during system **development**
- When all development has ended, **test** data used for final evaluation
- Ensures final system not influenced by test data
- If more development needed, get new dataset!

Co…

1. … classifications
2. … sets:
   … development
3. … H
4. …
   devtest data
5. Modify approach, repeat 3-4 as needed
6. Final test on *test* data



Zoo evaluation

train_and_test(learner, data, start, end) uses data[start:end] for test and rest for train

```
>>> dtl = DecisionTreeLearner
>>> train_and_test(dtl(), zoo, 0, 10)
1.0
>>> train_and_test(dtl(), zoo, 90, 100)
0.80000000000000004
>>> train_and_test(dtl(), zoo, 90, 101)
0.81818181818181823
>>> train_and_test(dtl(), zoo, 80, 90)
0.90000000000000002
```

Zoo evaluation

train_and_test(learner, data, start, end) uses data[start:end] for test and rest for train

- We hold out 10 data items for test; train on the other 91; show the accuracy on the test data
- Doing this **four times for different test subsets** shows accuracy from 80% to 100%
- **What's the true accuracy of our approach?**

K-fold Cross Validation

- **Problems:**
  - getting *ground truth* data expensive
  - need different test data for each test
  - experiments needed to find right *feature space* & parameters for ML algorithms
- **Goal:** minimize training+test data needed
- **Idea:** split training data into K subsets; use K-1 for *training* and one for *development testing*
- Repeat K times and average performance
- Common K values are 5 and 10

Zoo evaluation

- AIMA code has a cross_validation function that runs K-fold cross validation
- cross_validation(learner, data, K, N) does N iterations, each time randomly selecting 1/K data points for test, leaving rest for train

```
>>> cross_validation(dtl(), zoo, 10, 20)
0.9550000000000007
```

- This is a very common approach to evaluating the accuracy of a model during development
- Best practice is still to hold out a final test data set

Leave one out Cross Validation

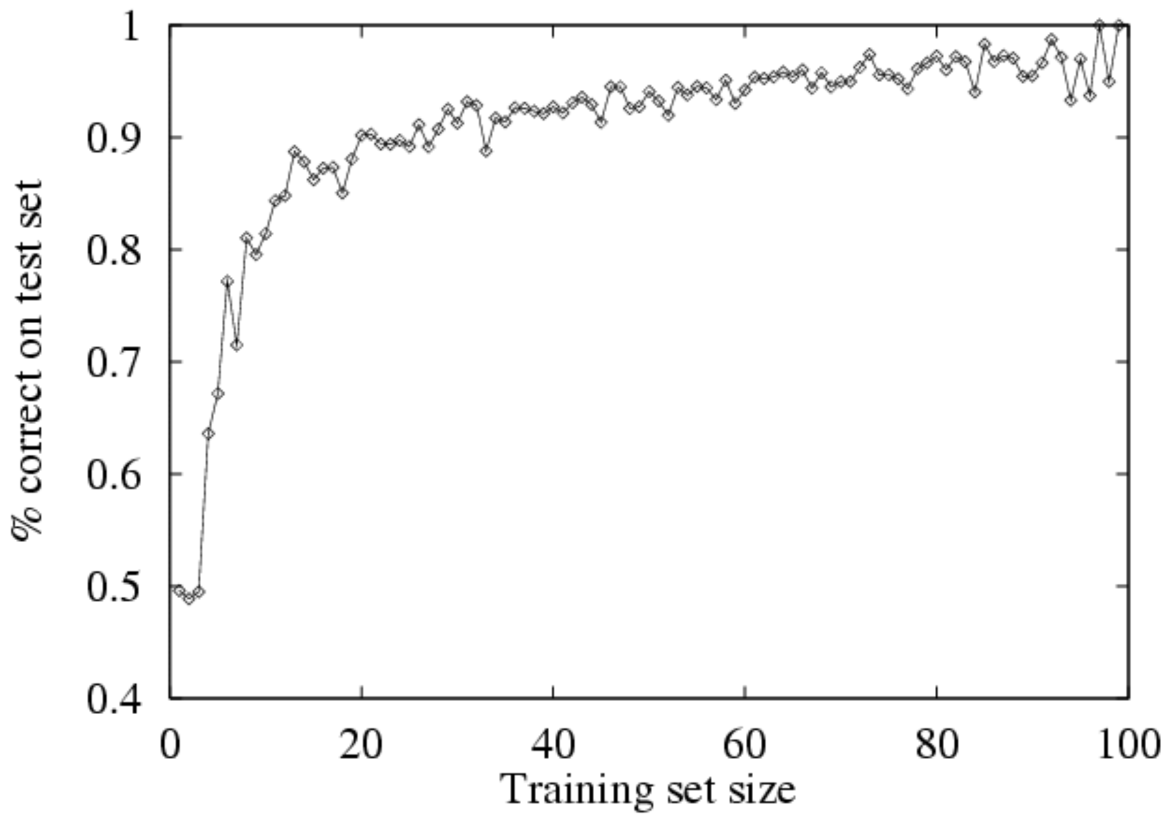
- AIMA code also has a *leave1out* function that runs a different set of experiments to estimate accuracy of the model
- *leave1out(learner, data)* does len(data) trials, each using one element for test, rest for train

```
>>> leave1out(dtl(), zoo)
0.97029702970297027
```

- K-fold cross validation can be too pessimistic, since it only trains with 80% or 90% of the data
- The leave one out evaluation is an alternative

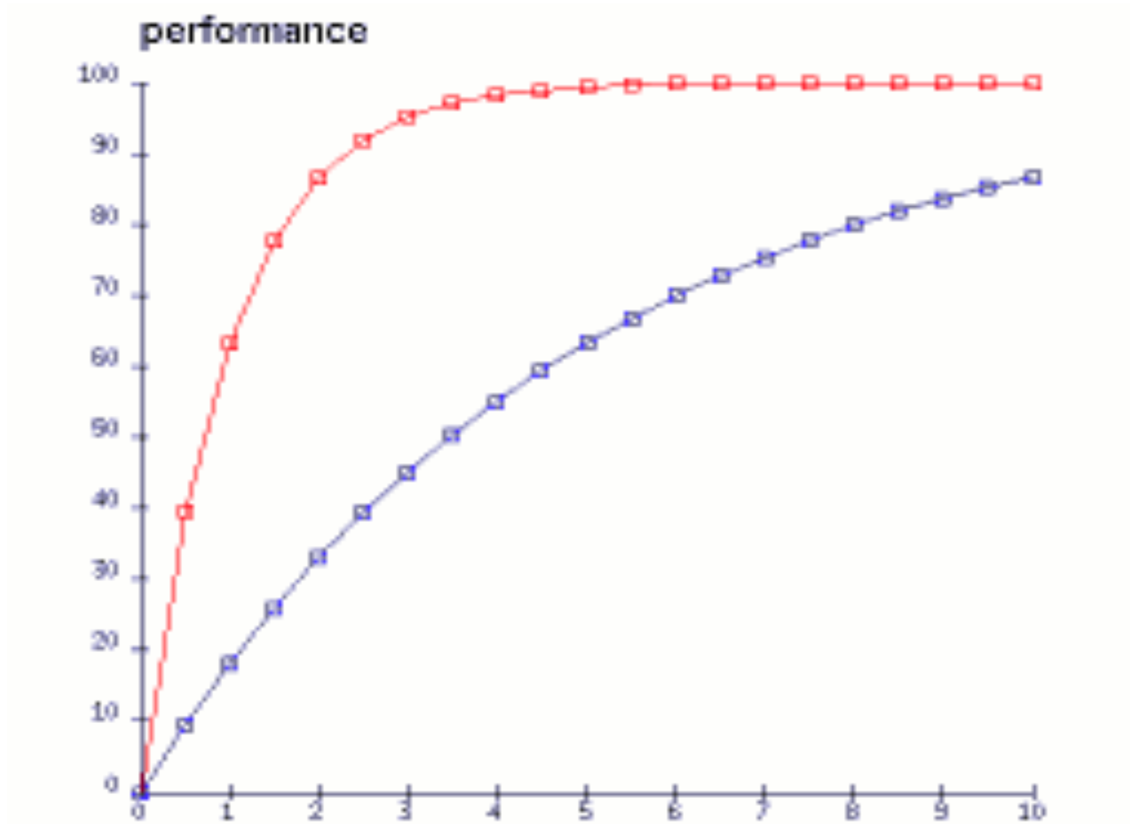
Learning curve (1)

A learning curve shows accuracy on test set as a function of training set size or (for neural networks) running time



Learning curve

- When evaluating ML algorithms, steeper learning curves are better
- They represents faster learning with less data



Training set size

Here the system with the red curve is better since it requires less data to achieve desired accuracy

EVALUATION METRICS

Classification Evaluation: the 2-by-2 contingency table

Let’s assume there are two classes/labels

● ○

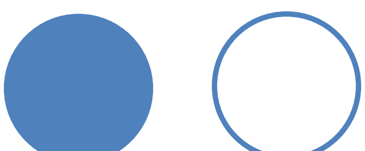
Assume ● is the “positive” label

Given X, our classifier predicts either label

p(● |X) vs. p(○ |X)

Classification Evaluation: the 2-by-2 contingency table

	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed		
Not selected/ not guessed		



Classes/Choices

Classification Evaluation: the 2-by-2 contingency table

	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed	True Positive (TP) *Actual* *Guessed*	
Not selected/ not guessed		

Classes/Choices

Classification Evaluation: the 2-by-2 contingency table

	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed	True Positive (TP) *Actual* *Guessed*	False Positive (FP) *Actual* *Guessed*
Not selected/ not guessed		

Classes/Choices

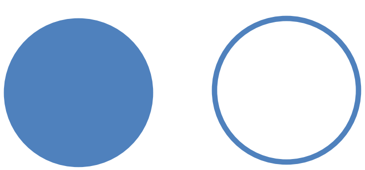
Classification Evaluation: the 2-by-2 contingency table

	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed	True Positive *Actual* (TP) *Guessed*	False Positive *Actual* (FP) *Guessed*
Not selected/ not guessed	False Negative *Actual* (FN) *Guessed*	

Classes/Choices

Classification Evaluation: the 2-by-2 contingency table

What label does our system predict? (↓)	*What is the actual label?* Actually Correct	Actually Incorrect
Selected/ Guessed	True Positive (TP) *Actual* *Guessed*	False Positive (FP) *Actual* *Guessed*
Not selected/ not guessed	False Negative (FN) *Actual* *Guessed*	True Negative (TN) *Actual* *Guessed*



Classes/Choices

Classification Evaluation: the 2-by-2 contingency table

	What is the actual label?	
What label does our system predict? (↓)	Actually Correct	Actually Incorrect
Selected/ Guessed	True Positive (TP) *Actual* *Guessed*	False Positive (FP) *Actual* *Guessed*
Not selected/ not guessed	False Negative (FN) *Actual* *Guessed*	True Negative (TN) *Actual* *Guessed*

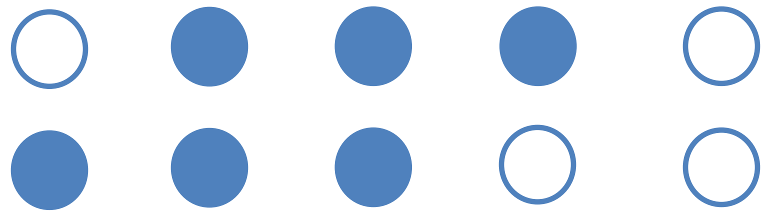
Classes/Choices

Construct this table by *counting* the number of TPs, FPs, FNs, TNs

Contingency Table Example

Predicted:

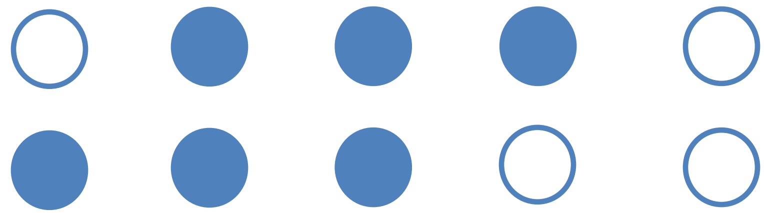
Actual:



Contingency Table Example

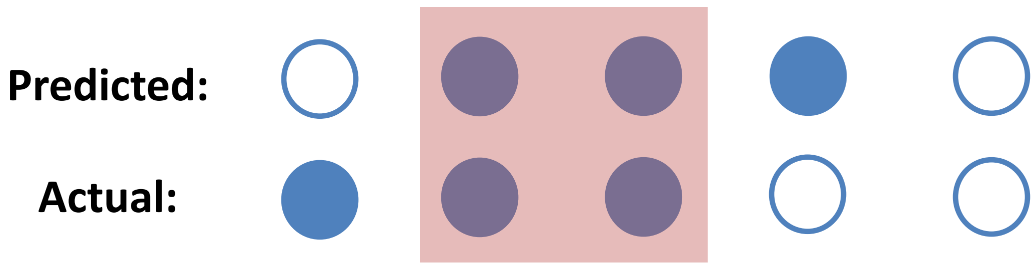
Predicted:

Actual:



	What is the actual label?	
What label does our system predict? (↓)	Actually Correct	Actually Incorrect
Selected/ Guessed	True Positive (TP)	False Positive (FP)
Not selected/ not guessed	False Negative (FN)	True Negative (TN)

Contingency Table Example

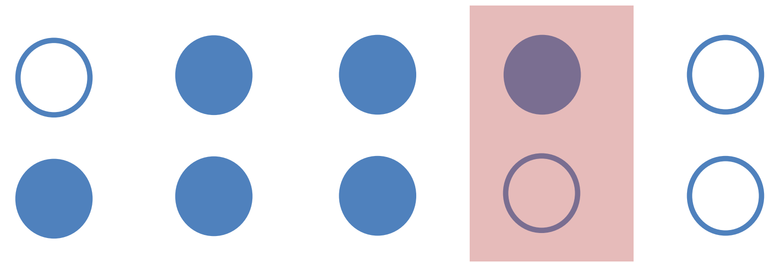


	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed	True Positive (TP) = 2	False Positive (FP)
Not selected/ not guessed	False Negative (FN)	True Negative (TN)

Contingency Table Example

Predicted:

Actual:

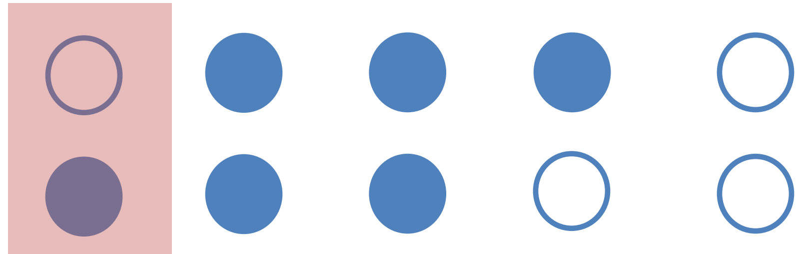


	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed	True Positive (TP) = 2	False Positive (FP) = 1
Not selected/ not guessed	False Negative (FN)	True Negative (TN)

Contingency Table Example

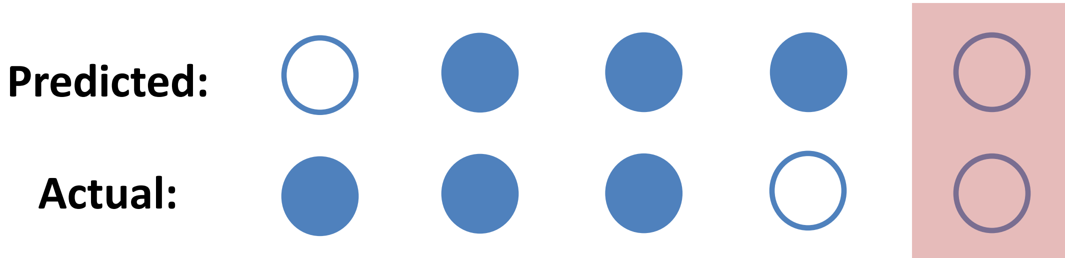
Predicted:

Actual:



	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed	True Positive (TP) = 2	False Positive (FP) = 1
Not selected/ not guessed	False Negative (FN) = 1	True Negative (TN)

Contingency Table Example

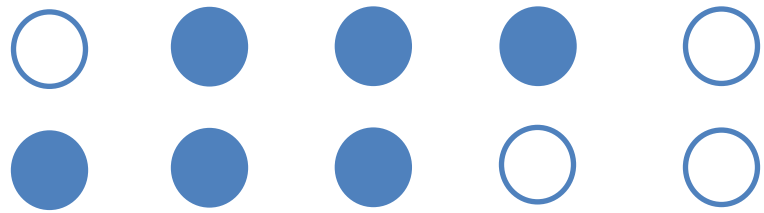


	What is the actual label?	
What label does our system predict? (↓)	Actually Correct	Actually Incorrect
Selected/ Guessed	True Positive (TP) = 2	False Positive (FP) = 1
Not selected/ not guessed	False Negative (FN) = 1	True Negative (TN) = 1

Contingency Table Example

Predicted:

Actual:



	What is the actual label?	
What label does our system predict? (↓)	Actually Correct	Actually Incorrect
Selected/ Guessed	True Positive (TP) = 2	False Positive (FP) = 1
Not selected/ not guessed	False Negative (FN) = 1	True Negative (TN) = 1

Classification Evaluation: Accuracy, Precision, and Recall

Accuracy: % of items correct

$$\frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}}$$

	Actually Correct	Actually Incorrect
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

Classification Evaluation: Accuracy, Precision, and Recall

Accuracy: % of items correct

$$\frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}}$$

Precision: % of selected items that are correct

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

	Actually Correct	Actually Incorrect
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

Classification Evaluation: Accuracy, Precision, and Recall

Accuracy: % of items correct

$$\frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}}$$

Precision: % of selected items that are correct

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Recall: % of correct items that are selected

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

	Actually Correct	Actually Incorrect
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

Classification Evaluation: Accuracy, Precision, and Recall

Accuracy: % of items correct

$$\frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}}$$

Precision: % of selected items that are correct

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Min: 0 ☹
Max: 1 😀

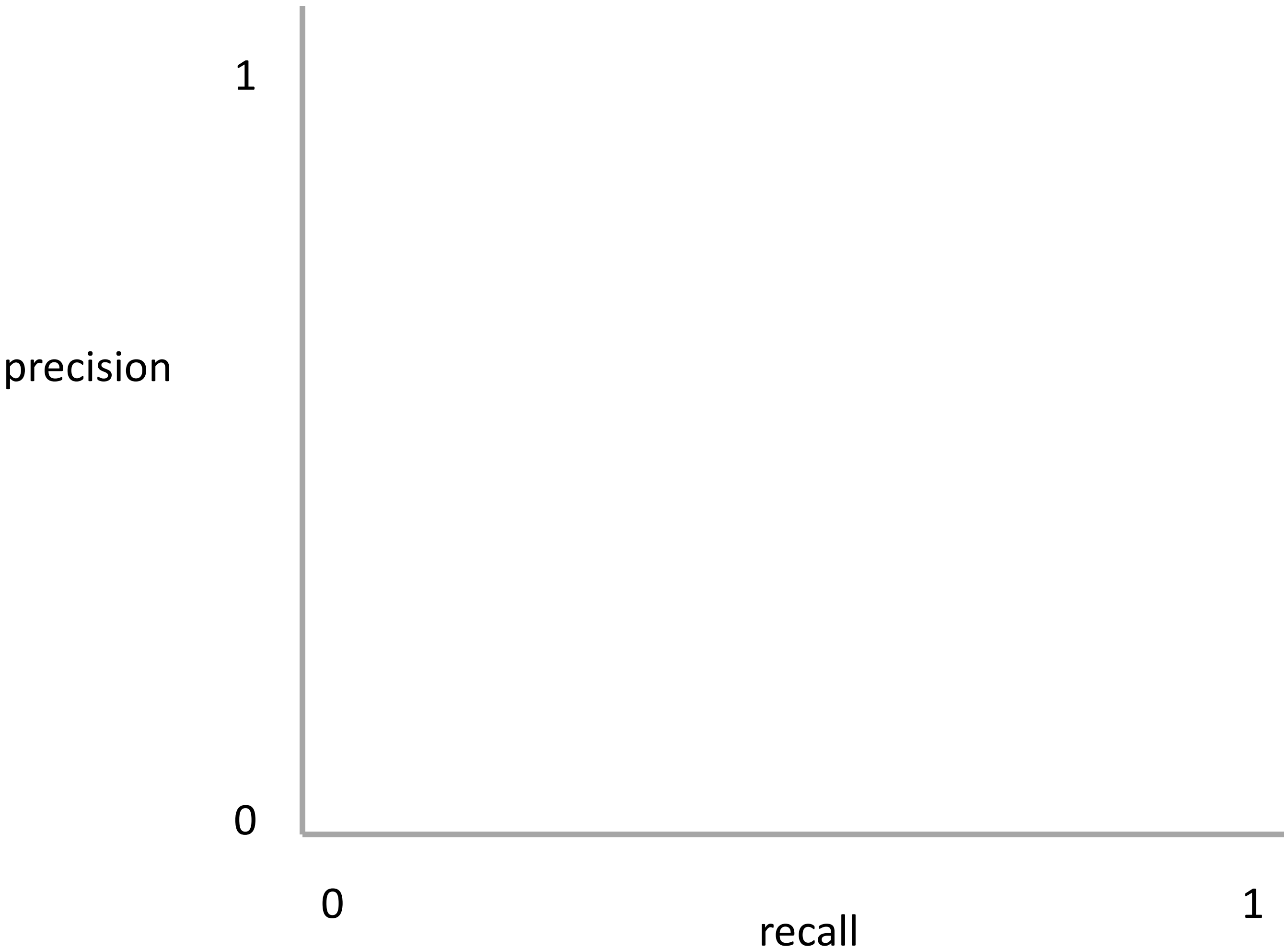
Recall: % of correct items that are selected

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

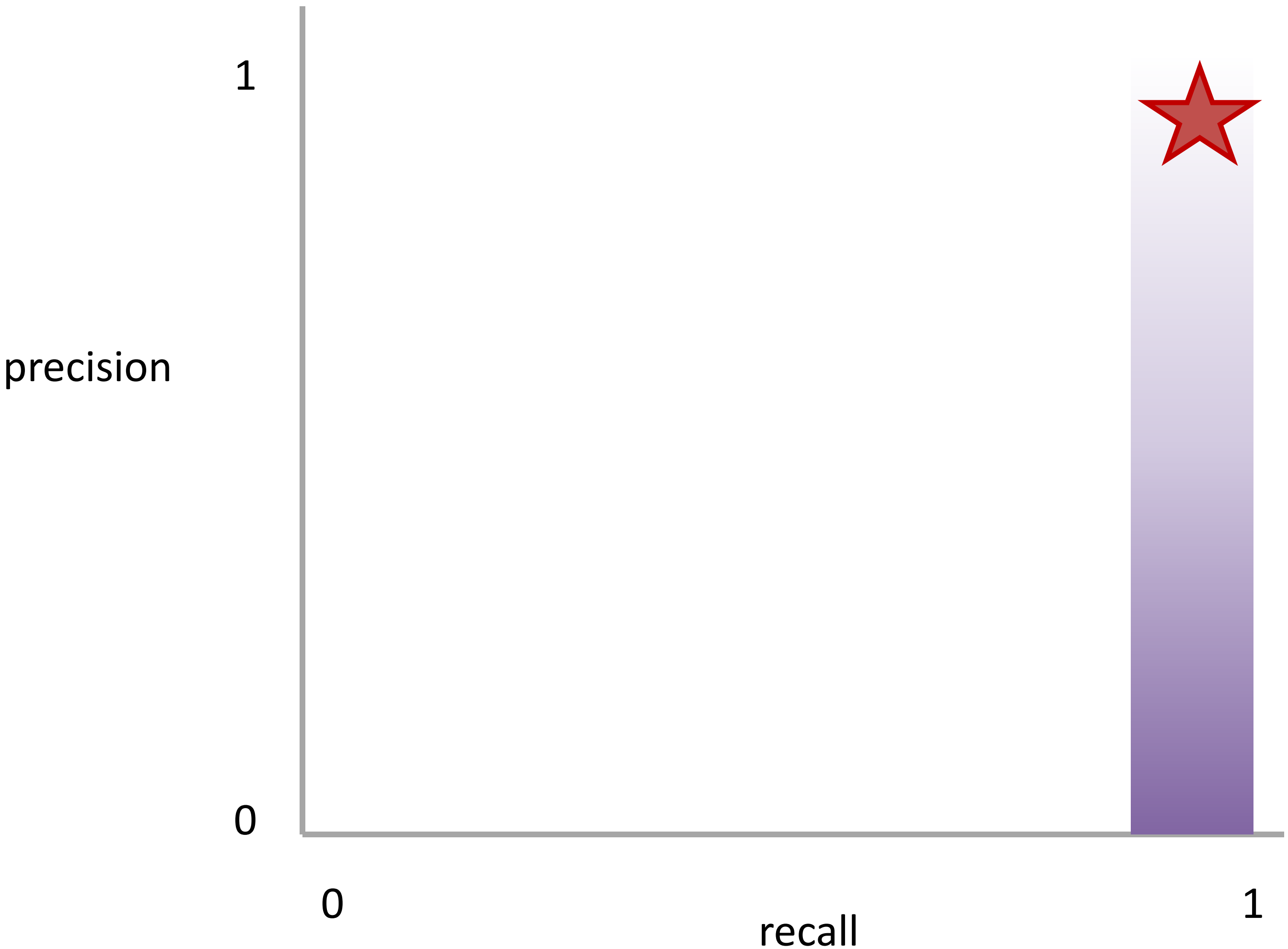
	Actually Correct	Actually Incorrect
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

Precision and Recall Present a Tradeoff

Q: Where do you want your ideal model ?



Precision and Recall Present a Tradeoff

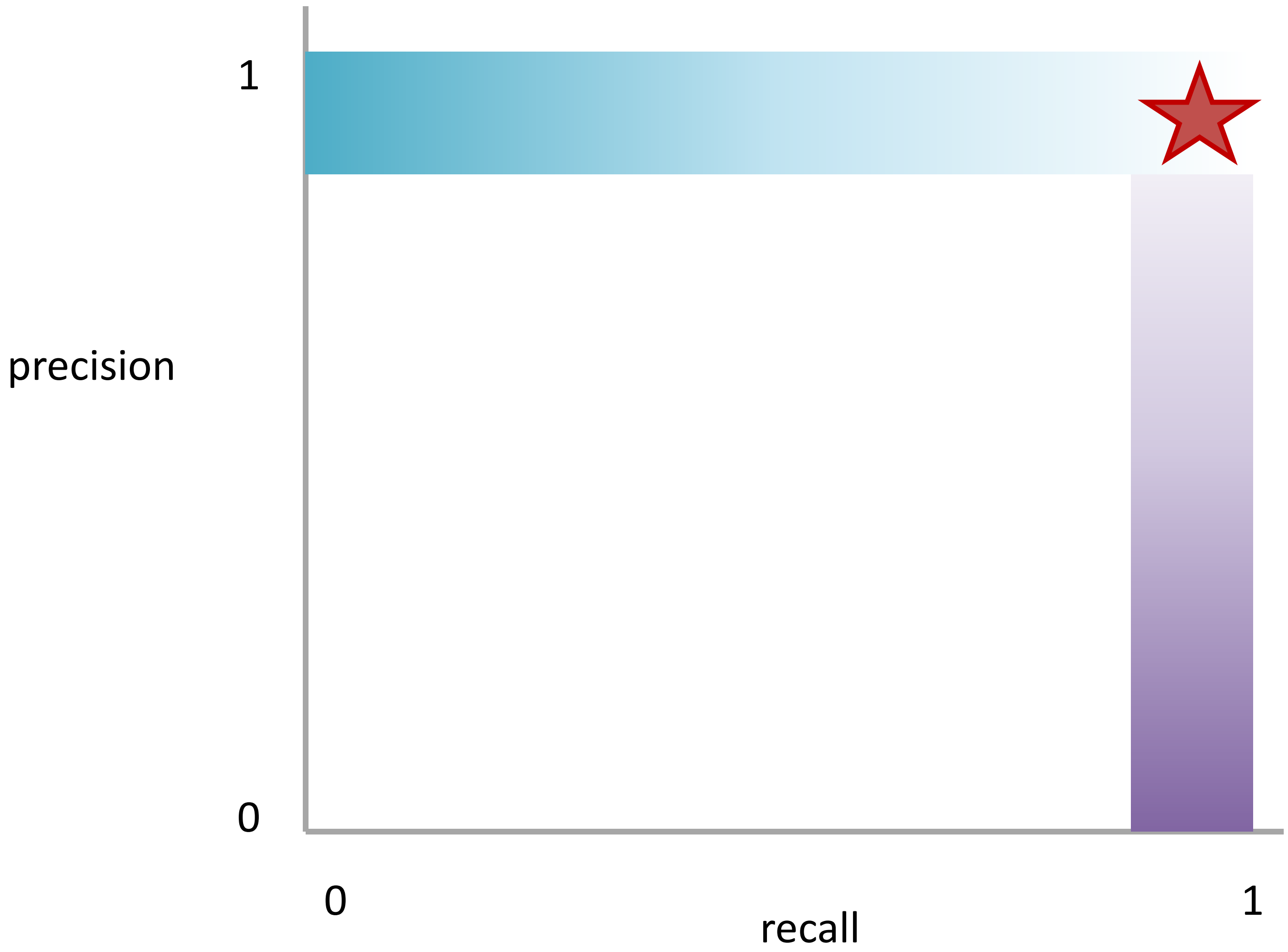


Q: Where do you want your ideal model?

Q: You have a model that always identifies correct instances. Where on this graph is it?

Q: You have a model that only make correct predictions. Where on this graph is it?

Precision and Recall Present a Tradeoff

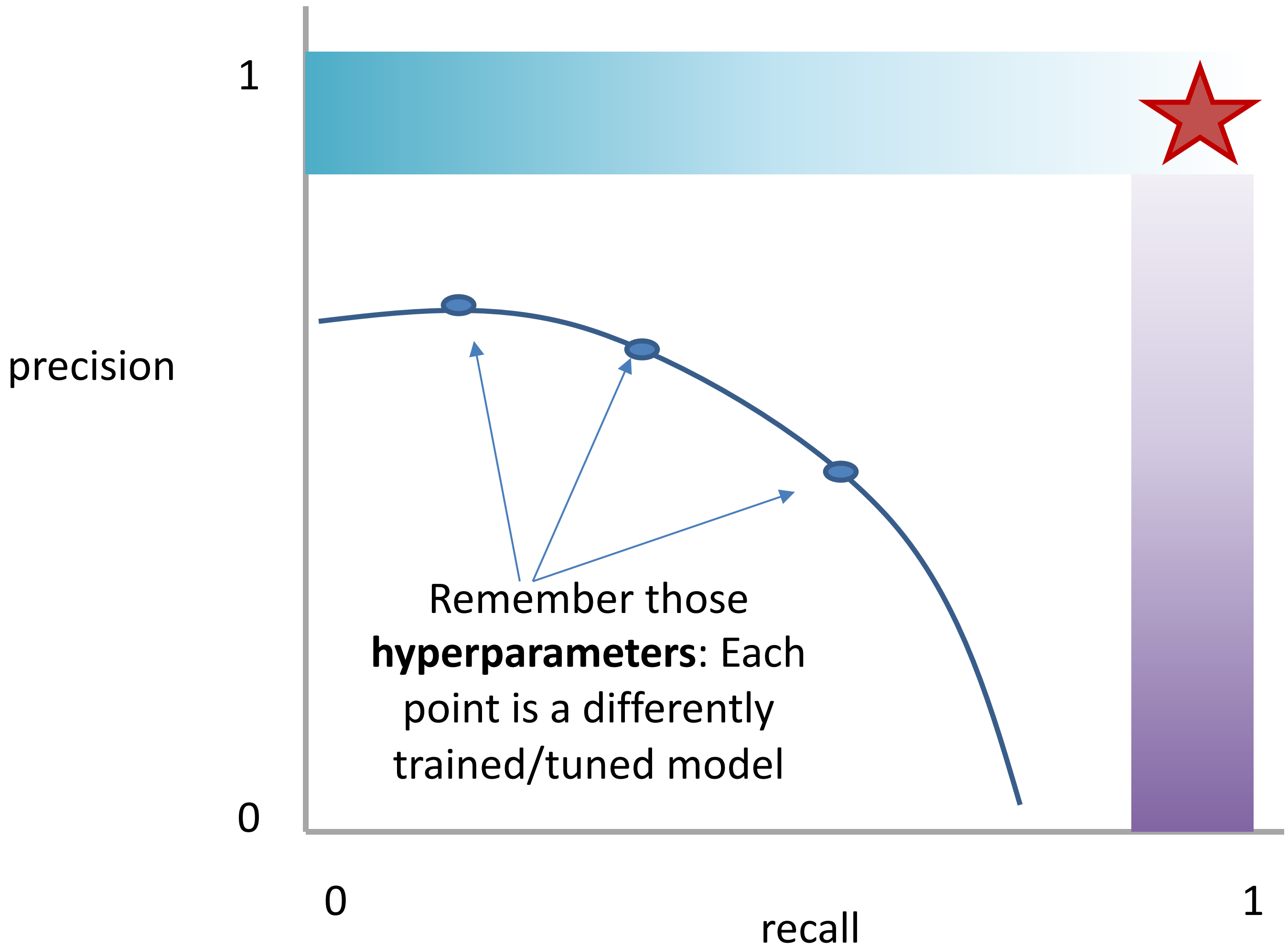


Q: Where do you want your ideal model ?

Q: You have a model that always identifies correct instances. Where on this graph is it?

Q: You have a model that only make correct predictions. Where on this graph is it?

Precision and Recall Present a Tradeoff



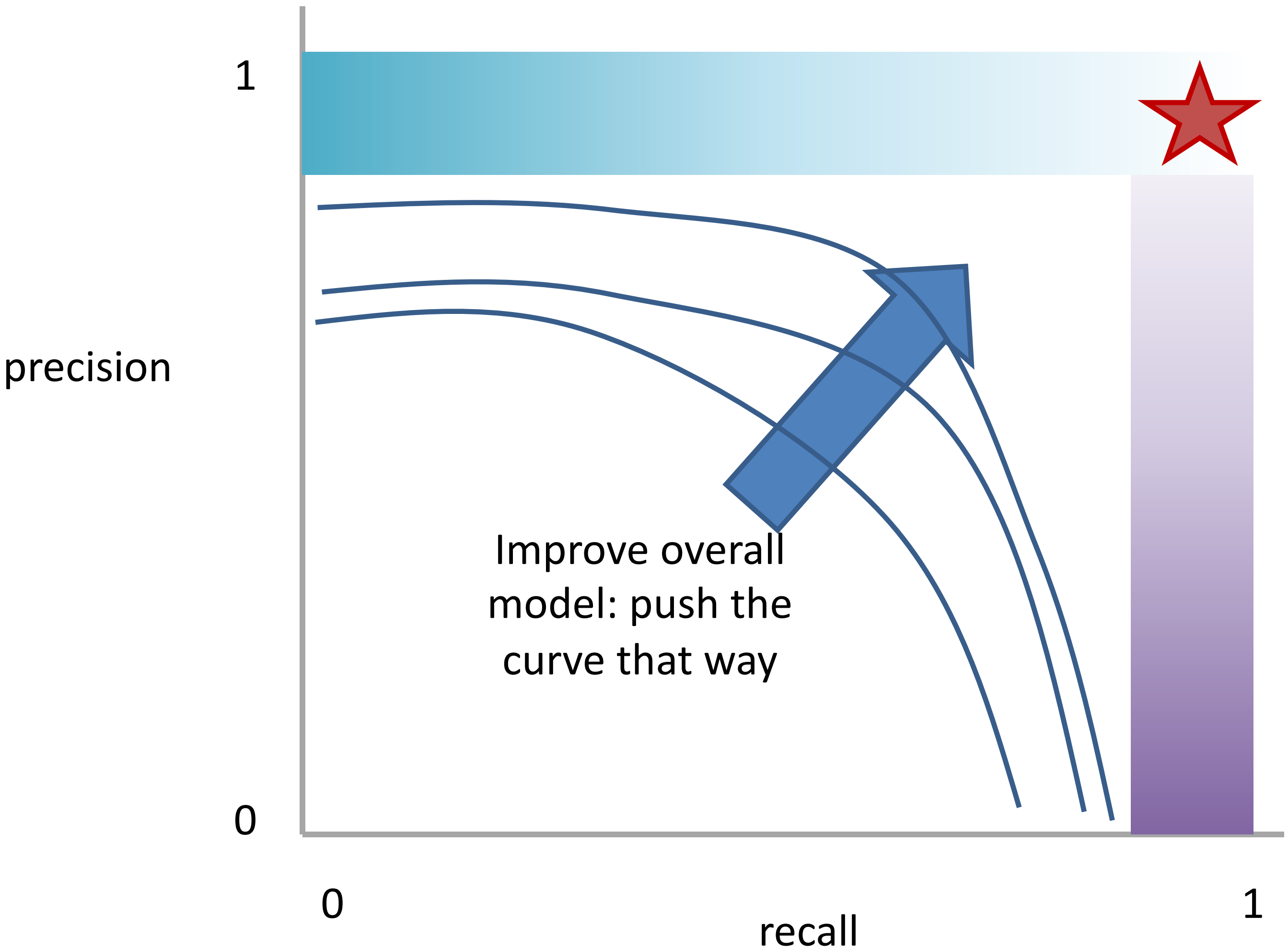
Q: Where do you want your ideal model ?

Q: You have a model that always identifies correct instances. Where on this graph is it?

Q: You have a model that only make correct predictions. Where on this graph is it?

Idea: measure the tradeoff between precision and recall

Precision and Recall Present a Tradeoff



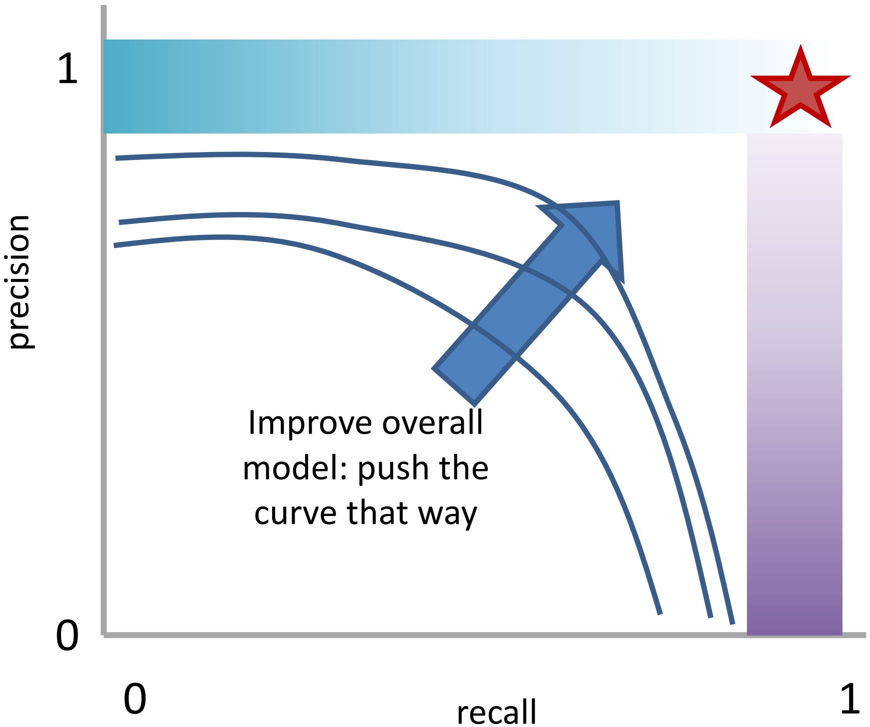
Q: Where do you want your ideal model?

Q: You have a model that always identifies correct instances. Where on this graph is it?

Q: You have a model that only make correct predictions. Where on this graph is it?

Idea: measure the tradeoff between precision and recall

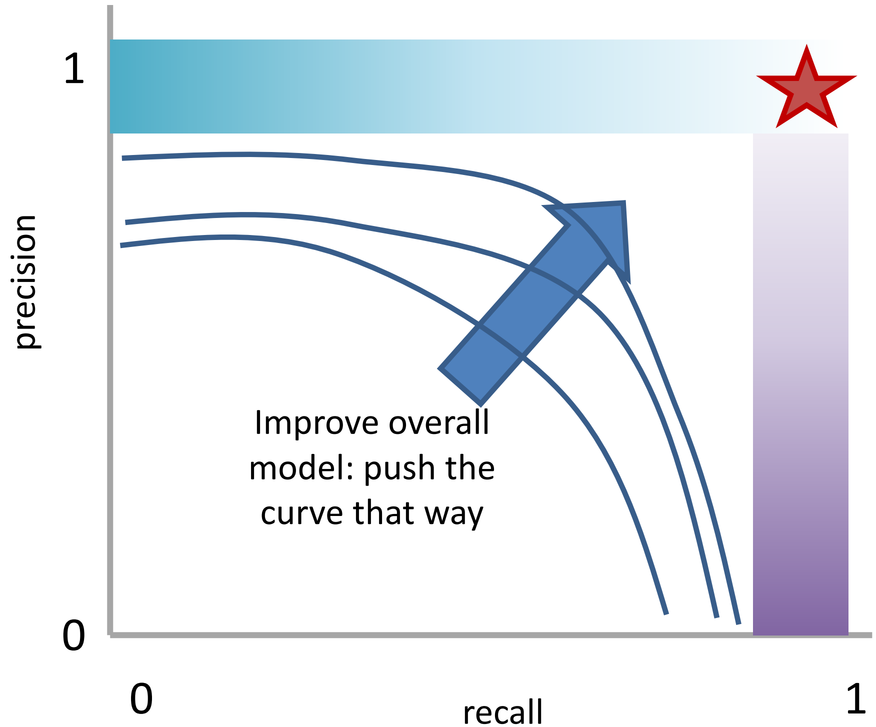
Measure this Tradeoff: Area Under the Curve (AUC)



AUC measures the area under this tradeoff curve

Min AUC: 0 ☹
Max AUC: 1 😀

Measure this Tradeoff: Area Under the Curve (AUC)



Min AUC: 0 ☹️
Max AUC: 1 😀

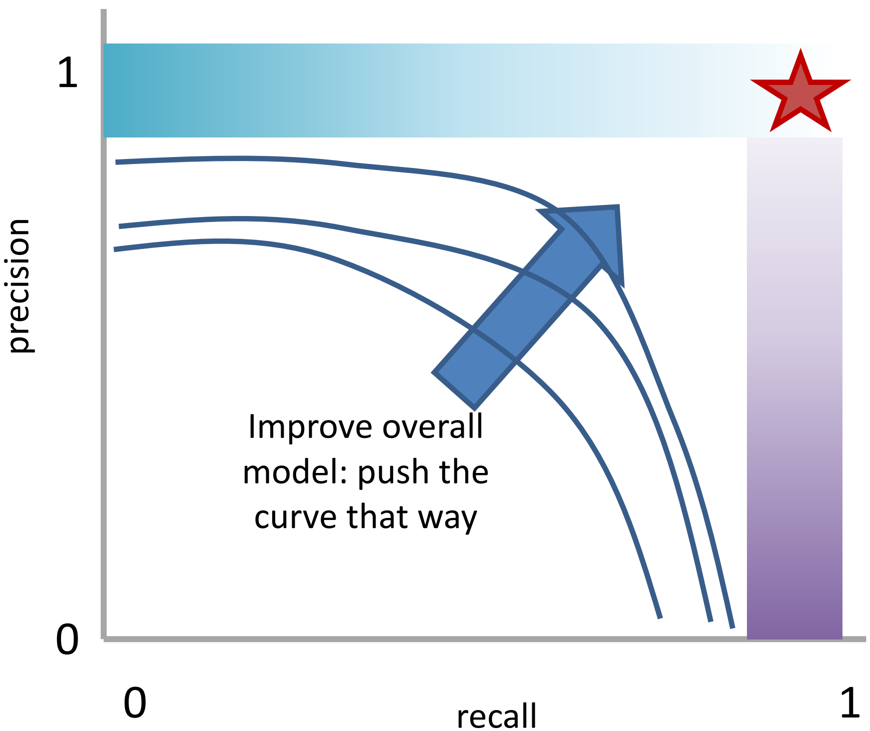
AUC measures the area under this tradeoff curve

1. Computing the curve

   You need true labels & predicted labels with some score/confidence estimate

   Threshold the scores and for each threshold compute precision and recall

Measure this Tradeoff: Area Under the Curve (AUC)



Min AUC: 0 ☹️
Max AUC: 1 😀

AUC measures the area under this tradeoff curve

1. Computing the curve

   You need true labels & predicted labels with some score/confidence estimate

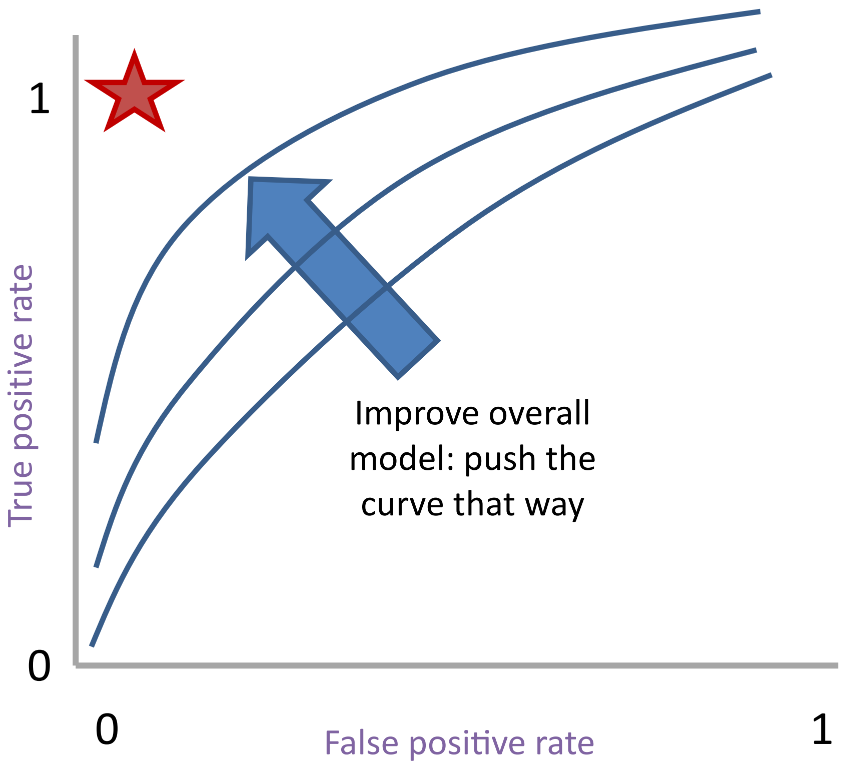
   Threshold the scores and for each threshold compute precision and recall

2. Finding the area

   How to implement: trapezoidal rule (& others)

In practice: external library like the sklearn.metrics module

Measure A Slightly Different Tradeoff: ROC-AUC



Min ROC-AUC: 0.5 ☹️
Max ROC-AUC: 1 😀

AUC measures the area under this tradeoff curve

1. Computing the curve
   You need true labels & predicted labels with some score/confidence estimate
   Threshold the scores and for each threshold compute metrics
2. Finding the area
   How to implement: trapezoidal rule (& others)

In practice: external library like the sklearn.metrics module

Main variant: ROC-AUC

Same idea as before but with some flipped metrics

A combined measure: F

Weighted (harmonic) average of **P**recision & **R**ecall

$$F = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha) \frac{1}{R}}$$

A combined measure: F

Weighted (harmonic) average of **P**recision & **R**ecall

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(1+\beta^2) * P * R}{(\beta^2 * P) + R}$$

algebra
(not important)

A combined measure: F

Weighted (harmonic) average of **P**recision & **R**ecall

$$F = \frac{(1+\beta^2) \ * P \ * R}{(\beta^2 * P) + R}$$

Balanced F1 measure: β=1

$$F_1 = \frac{2 \ * P \ * R}{P + R}$$

P/R/F in a Multi-class Setting: Micro- vs. Macro-Averaging

If we have more than one class, how do we combine multiple performance measures into one quantity?

Macroaveraging: Compute performance for each class, then average.

Microaveraging: Collect decisions for all classes, compute contingency table, evaluate.

P/R/F in a Multi-class Setting: Micro- vs. Macro-Averaging

Macroaveraging: Compute performance for each class, then average.

$$\text{macroprecision} = \sum_{c} \frac{\mathrm{TP_c}}{\mathrm{TP_c} + \mathrm{FP_c}} = \sum_{c} \text{precision}_c$$

Microaveraging: Collect decisions for all classes, compute contingency table, evaluate.

$$\text{microprecision} = \frac{\sum_\mathrm{c} \mathrm{TP_c}}{\sum_\mathrm{c} \mathrm{TP_c} + \sum_\mathrm{c} \mathrm{FP_c}}$$

P/R/F in a Multi-class Setting: Micro- vs. Macro-Averaging

Macroaveraging: Compute performance for each class, then average.

when to prefer the macroaverage?

$$\text{macroprecision} = \sum_c \frac{\mathrm{TP_c}}{\mathrm{TP_c} + \mathrm{FP_c}} = \sum_c \text{precision}_c$$

Microaveraging: Collect decisions for all classes, compute contingency table, evaluate.

when to prefer the microaverage?

$$\text{microprecision} = \frac{\sum_\mathrm{c} \mathrm{TP_c}}{\sum_\mathrm{c} \mathrm{TP_c} + \sum_\mathrm{c} \mathrm{FP_c}}$$

Micro- vs. Macro-Averaging: Example

Class 1

	Truth : yes	Truth : no
Classifier: yes	10	10
Classifier: no	10	970

Class 2

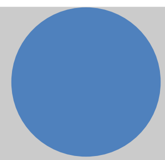
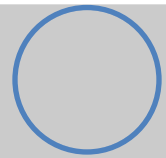
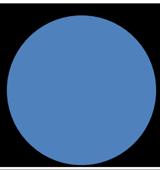
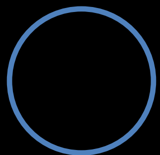
	Truth : yes	Truth : no
Classifier: yes	90	10
Classifier: no	10	890

Micro Ave. Table

	Truth : yes	Truth : no
Classifier: yes	100	20
Classifier: no	20	1860

- Macroaveraged precision: (0.5 + 0.9)/2 = 0.7
- Microaveraged precision: 100/120 = .83
- Microaveraged score is dominated by score on frequent classes

Confusion Matrix: Generalizing the 2-by-2 contingency table

		Correct Value		
				
Guessed Value		#	#	#
		#	#	#
		#	#	#

Confusion Matrix: Generalizing the 2-by-2 contingency table

		Correct Value		
		●	○	▭
Guessed Value	●	80	9	11
	○	7	86	7
	▭	2	8	9

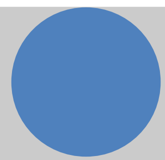
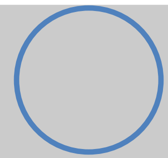
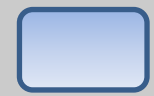
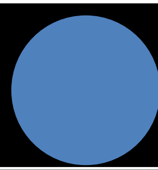
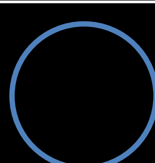
Q: Is this a good result?

Confusion Matrix: Generalizing the 2-by-2 contingency table

		Correct Value		
		●	○	▭
Guessed Value	●	30	40	30
	○	25	30	50
	▭	30	35	35

Q: Is this a good result?

Confusion Matrix: Generalizing the 2-by-2 contingency table

		Correct Value		
				
Guessed Value		7	3	90
		4	8	88
		3	7	90

Q: Is this a good result?