Ensemble Methods

Sameer Singh and Conal Sathi

BANA 290: ADVANCED DATA ANALYTICS

MACHINE LEARNING FOR TEXT

SPRING 2018

May 1, 2018

Upcoming…

Homework

- Homework 2 is out!
- Due: May 11, 2017
- Start early as this is a bit more involved than HW1
- Conal's office hours 9:30-10:30AM tomorrow (Wed May 2^{nd})
- HW1 Grades and Highlights Out

Project

- Instructions will be out this week
- Proposal due: May 15th

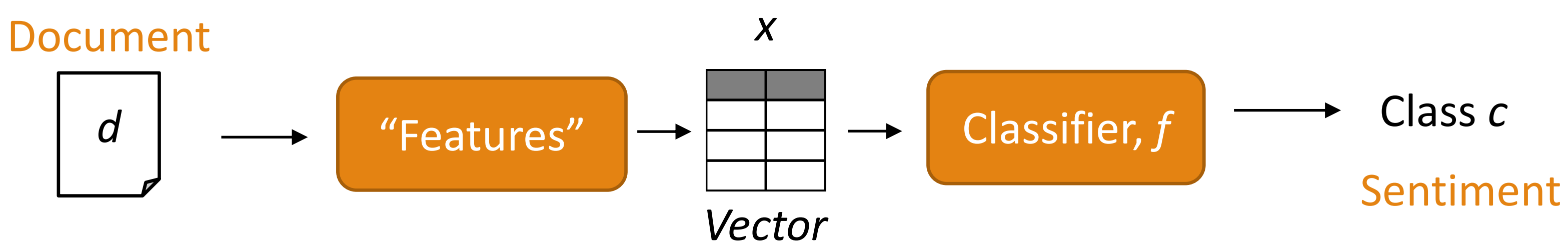
Participation

- Midterm Evaluations out (see email)
- Due: May 8^{th}
- Please give us feedback!
- Reminder: questions/answers/upvotes also count for participation on Piazza

Output of Classifiers

BEYOND THE CLASS LABELS: CONFIDENCE SCORES

Classifier Scores



Usually, they don't do it directly

$f(x, c_1)$

$f(x, c_2)$

$f(x, c_3)$

$f(x, c_4)$

NaiveBayes?

KNN?

This score is the confidence!

Recap: Naïve Bayes

Classification

d → Features → Classifier, *f* → Class *c*

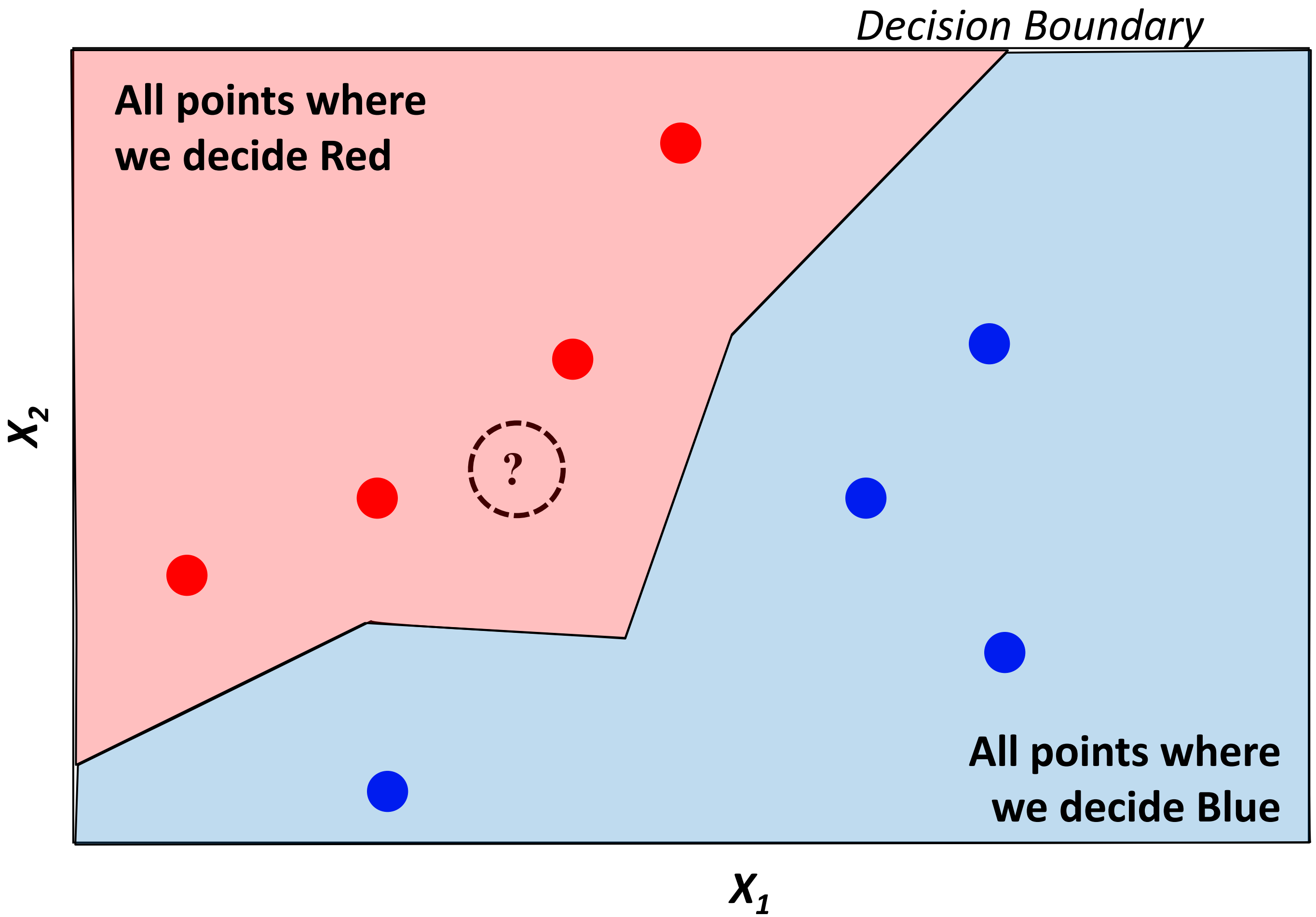
$$P(c_i|d) = P(c_i)\prod_{f_d} P(fd|ci)$$

Prob. of class i for document

How common is class c?

How common is feature f_d for class c_i?

Recap: Nearest Neighbor Classification

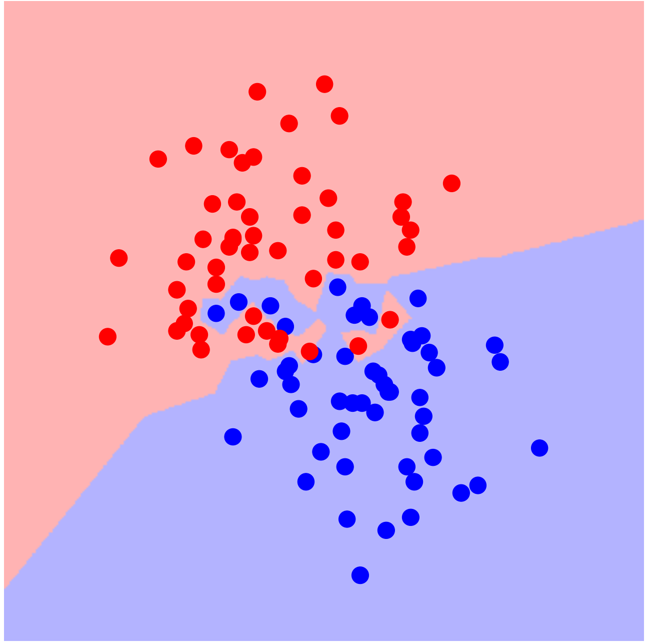


kNN Decision Boundary

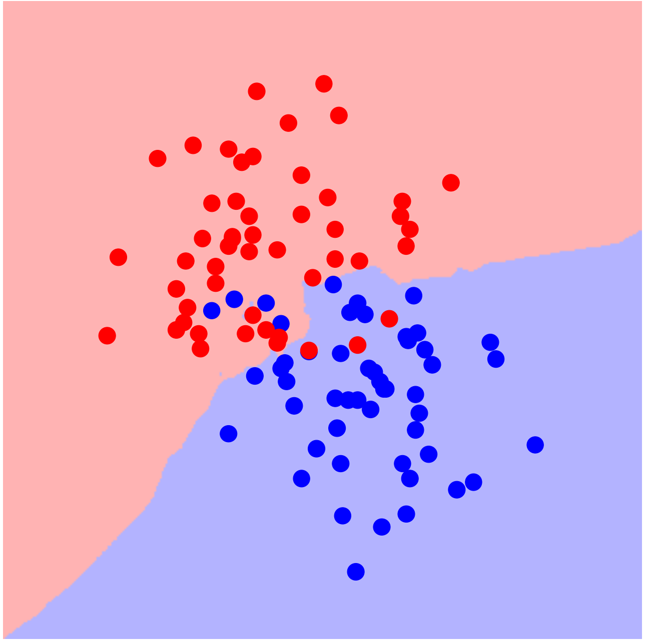
Increasing k "simplifies" decision boundary

- Majority voting means less emphasis on individual points

- K = 1



K = 7



Using these Classifier Scores

The classifier score (ideally) correlates with the prediction accuracy

How would you use these scores?

- Identify examples where your classifier is unconfident and bring in human involvement (Active learning)
- Identify cases where the classifier is confident and add them to the training set (Self-training – we'll talk about this in a future lecture)
- If you have multiple classifiers, you can use their scores to vote! (Ensembles)
  - E.g. if you have two classifiers and they disagree but one is confident and the other is unconfident
  - More on this today!

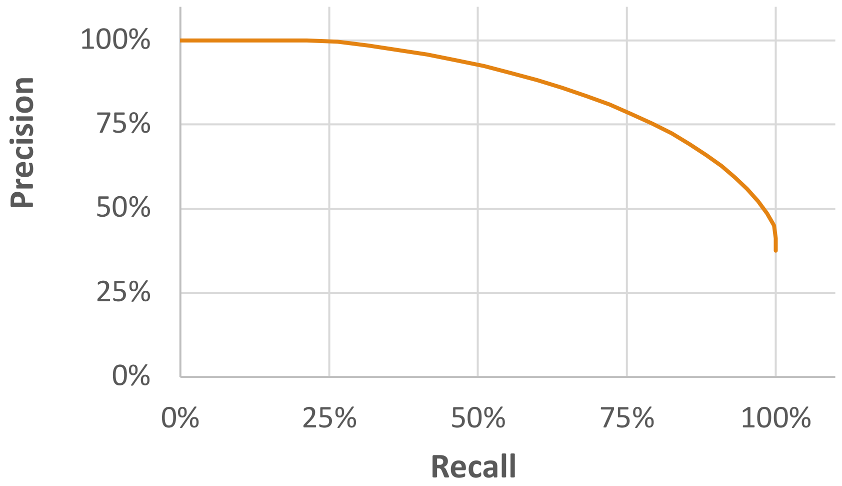
PR Curve: Relevance Classification

$$\text{Precision} = \frac{\text{relevant results returned}}{\text{results returned}} \qquad \text{Recall} = \frac{\text{relevant results returned}}{\text{relevant results}}$$

Can trade-off precision vs. recall by setting **confidence threshold**

Measure the curve on annotated dev data (or test data)

Choose a threshold where user is comfortable



Output of Classifiers

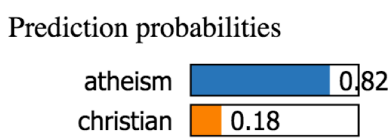
BEYOND THE CLASS LABELS: FEATURE WEIGHTS

-WHY DID THE CLASSIFIER MAKE THIS DECISION?

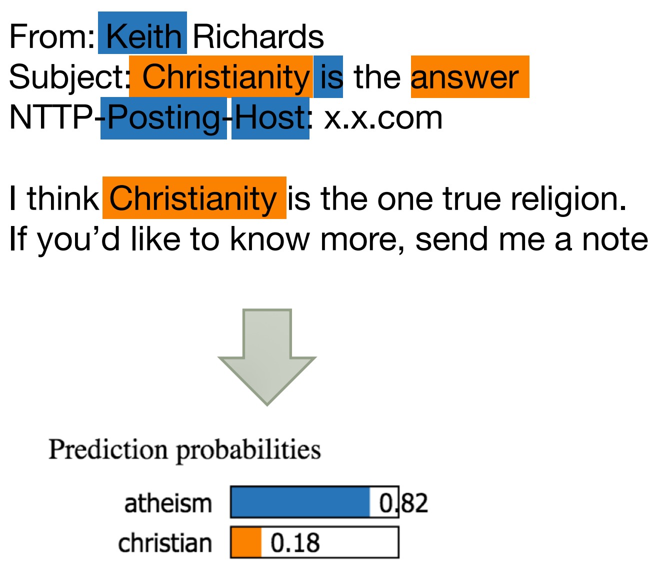
Why did the classifier make this classification?

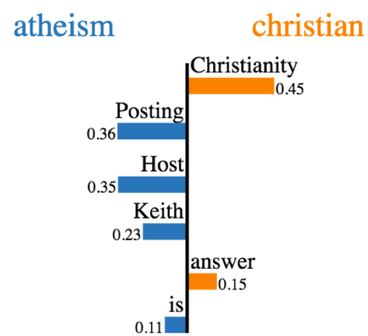
From: Keith Richards
Subject: Christianity is the answer
NTTP-Posting-Host: x.x.com

I think Christianity is the one true religion.
If you'd like to know more, send me a note



Can look at weights for features



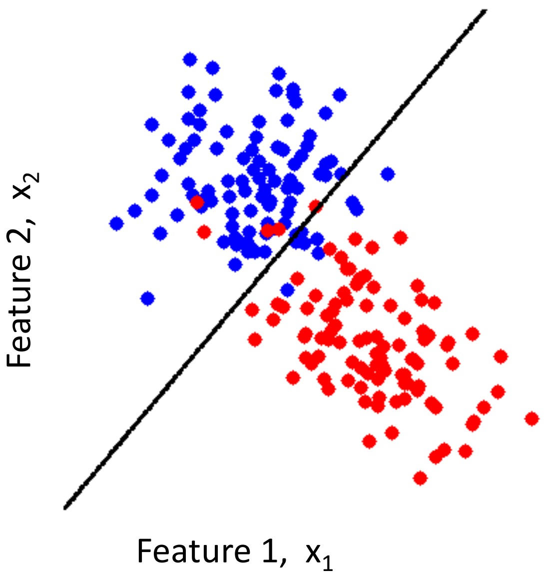


Logistic Regression

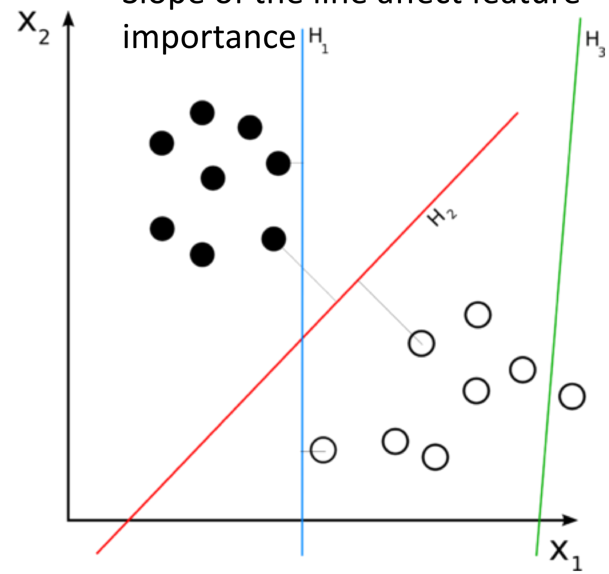
HOW IT WORKS AND ITS FEATURE WEIGHTS

Linear Classification, Binary

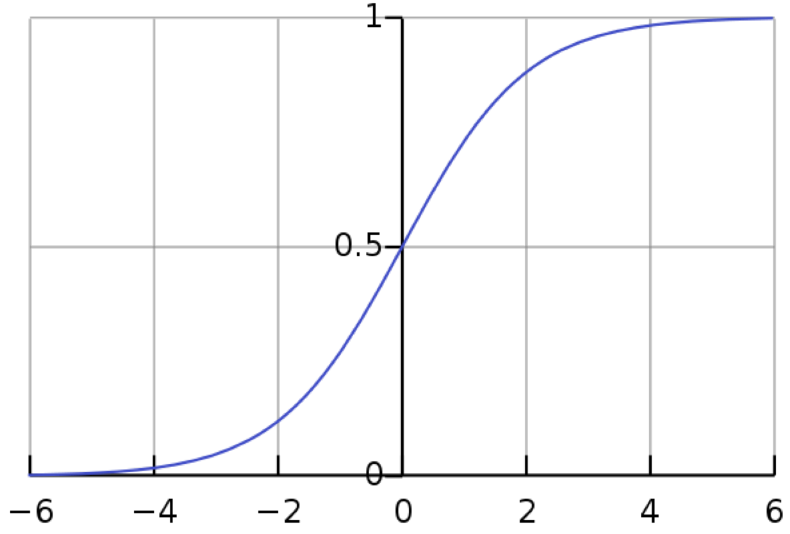
Decision boundary: $ax_1 + bx_2 = c$



Multiple decision boundaries!
H1 and H2 are equally good at minimizing training error.
Slope of the line affect feature importance



Logistic Regression



$$y = \frac{1}{1 + e^{-x}}$$

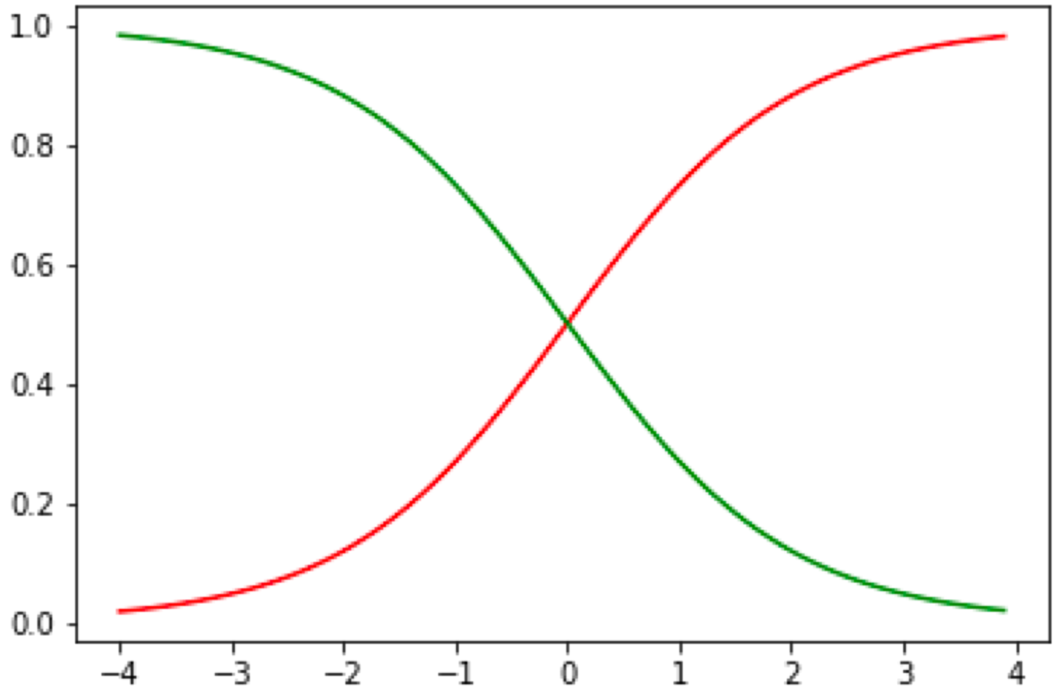
Function always gives a score between 0 and 1, making it a good function to use for a classifier

Can be extended to multiple features:

$$y = \frac{1}{1 + e^{a + bx_0 + cx_1 + dx_2 + \cdots}}$$

What do these parameters mean?

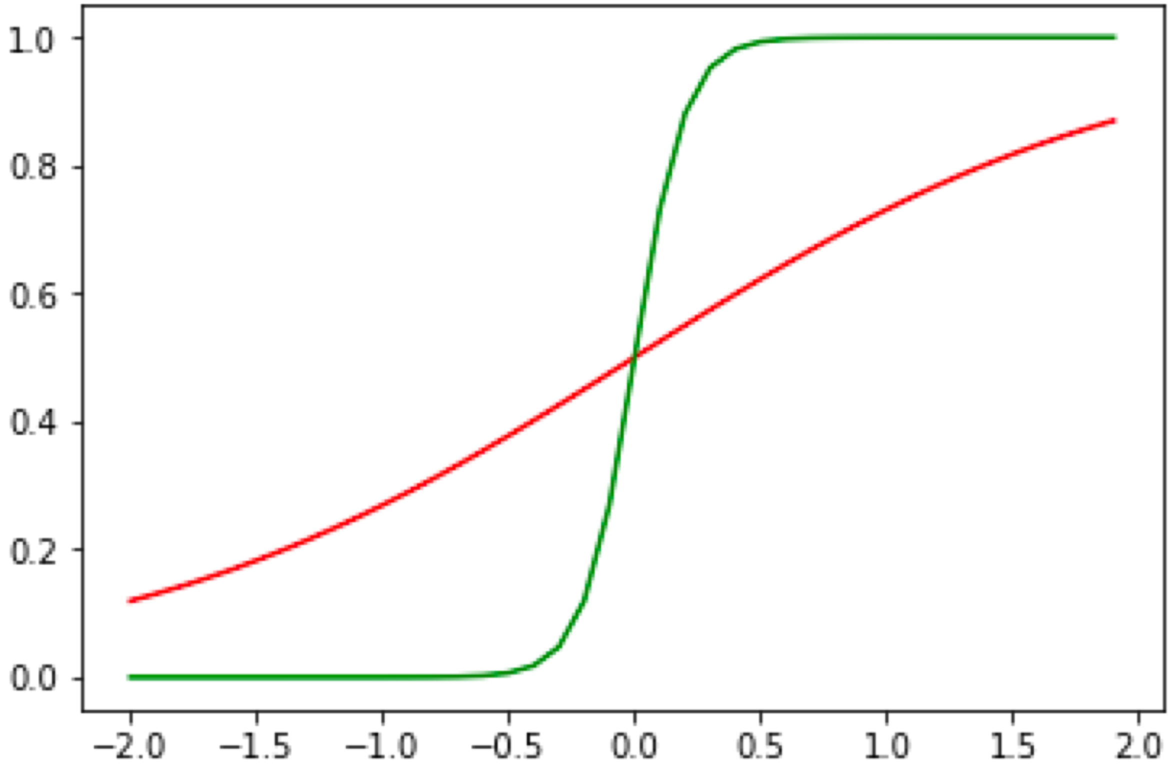
Logistic Regression



$$y = \frac{1}{1 + e^{-x}}$$

$$y = \frac{1}{1 + e^{x}}$$

Logistic Regression



$$y = \frac{1}{1 + e^{-10*x}}$$

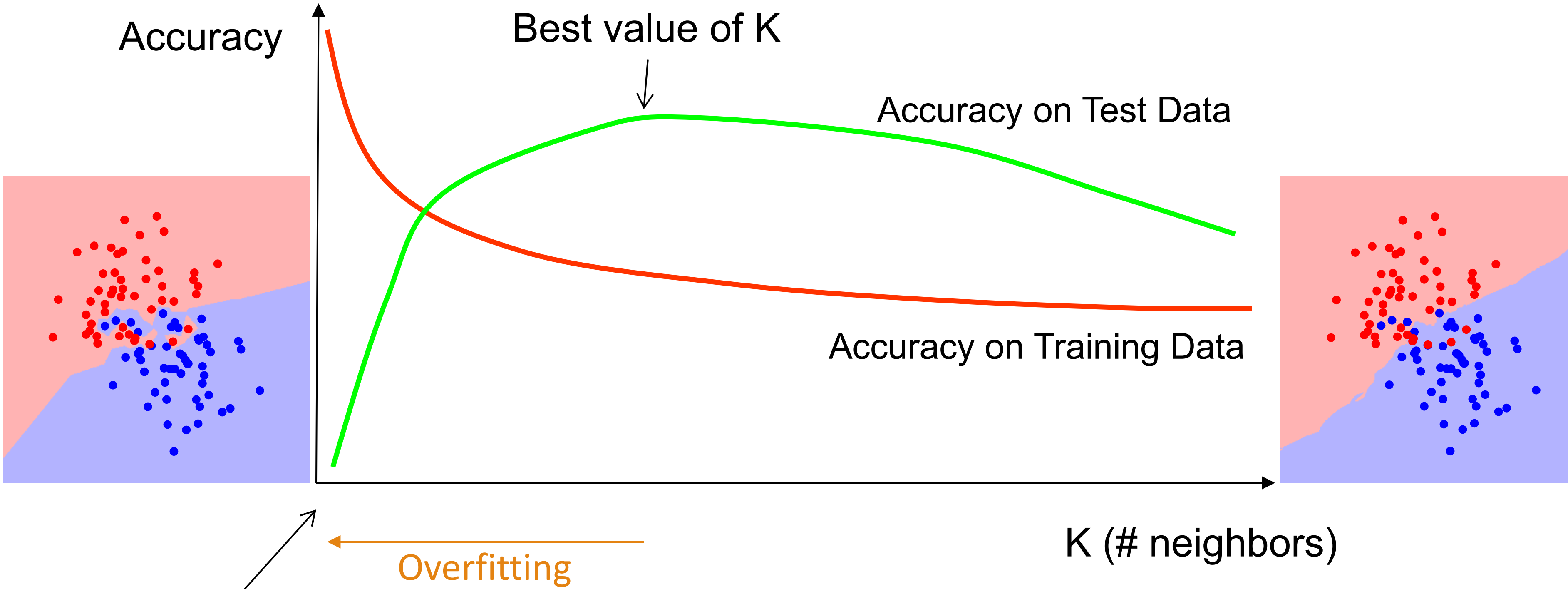
$$y = \frac{1}{1 + e^{-x}}$$

What about overfitting?

In K-Nearest Neighbors, we could modify *k* to affect overfitting

In Logistic Regression, we can modify the regularization parameter

Overfitting in K-Nearest Neighbors



K=1? Perfect accuracy!
Training data has been memorized...

BANA 290: ML FOR TEXT (SPRING 2018) 19

Overfitting in Logistic Regression

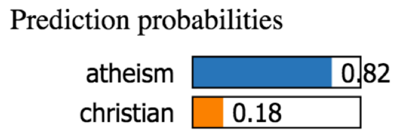
If you have lots of features and not much training examples, very likely to overfit

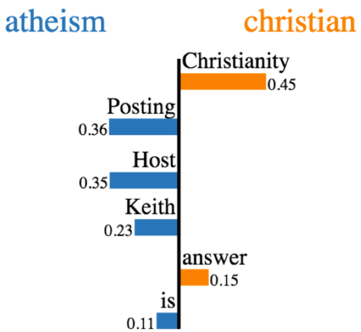
Why?

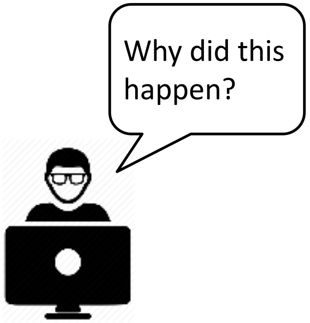
Fitting to noise in training data

From: Keith Richards
Subject: Christianity is the answer
NTTP-Posting-Host: x.x.com

I think Christianity is the one true religion.
If you'd like to know more, send me a note







Overfitting in Logistic Regression

If you have lots of features and not much training examples, very likely to overfit

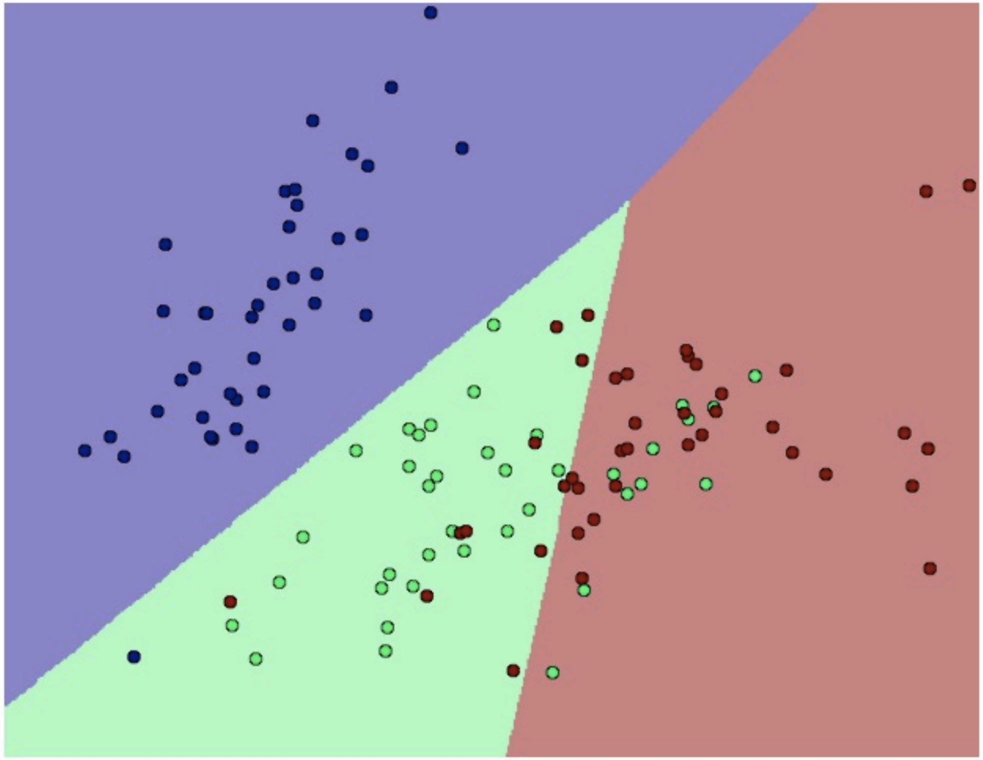
- Very likely that some subset of these 10K features are very important, while the rest is noise
- Regularization is to try to find the features that are noisy and bring their weights closer to 0

You have played with this parameter in previous in-class activities and will play with it for your homework assignment!

Code with ScikitLearn:

```
model = LogisticRegression(C=1)
model.fit(train_vecs, train.labels)
train_preds = model.predict(train_vecs)
```

Multi-class Linear Models



Instead of 1 weight per feature, now have *c* weights per feature!

Each feature as a weight for each class

y=comp.sys.mac.hardware top features

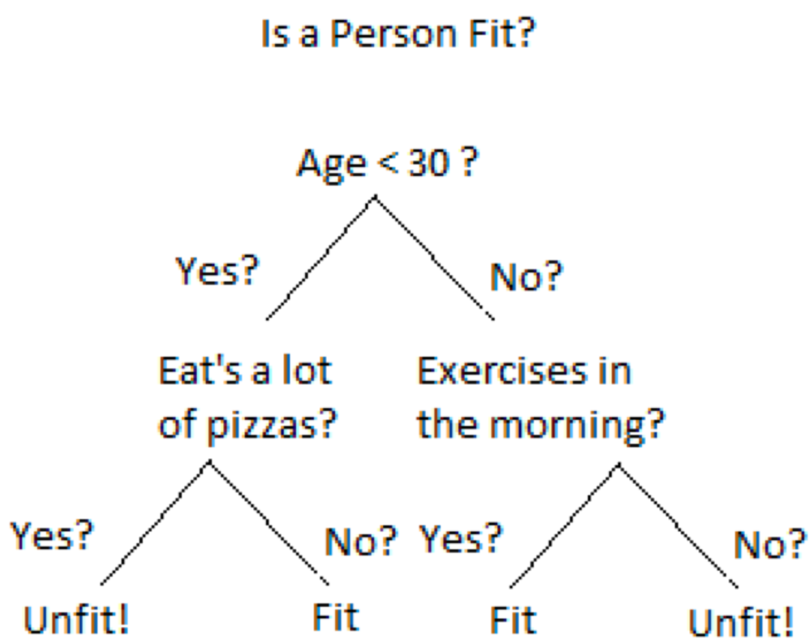
Weight?	Feature
+1.980	mac
+1.556	apple
+1.059	centris
+1.013	quadra
+0.883	se
… 6972 more positive …	
… 4950 more negative …	
-0.868	windows
-0.922	dos
-0.980	486
-1.142	controller
-1.355	pc

y=comp.graphics top features		**y=comp.os.ms-windows.misc** top features		**y=comp.sys.ibm.pc.hardware** top features		**y=comp.sys.mac.hardware** top features	
Weight?	Feature	Weight?	Feature	Weight?	Feature	Weight?	Feature
+1.643	graphics	+2.281	windows	+1.352	controller	+2.087	mac
+1.468	image	+1.375	cica	+1.348	ide	+1.704	apple
+1.262	3d	+1.188	win3	+1.198	os	+1.386	centris
+1.151	viewer	+1.087	change	+1.105	gateway	+1.188	se
+1.147	images	+1.067	mfc	+1.051	motherboard	+1.152	quadra
+1.094	68070	+1.030	win	+1.044	486	+1.057	powerbook
+1.062	algorithm	+0.959	nt	*… 8612 more positive …*		*… 7528 more positive …*	
… 10297 more positive …		*… 32492 more positive …*		*… 39039 more negative …*		*… 40123 more negative …*	
… 37354 more negative …		*… 15159 more negative …*		-1.136	windows	-1.022	dos
-1.032	monitor	-1.006	<BIAS>	-1.341	apple	-1.059	controller
-1.157	<BIAS>	-1.092	mac	-1.633	mac	-1.212	pc
-1.329	cica	-1.460	graphics	-1.645	<BIAS>	-1.730	windows

Non-linear Classifiers

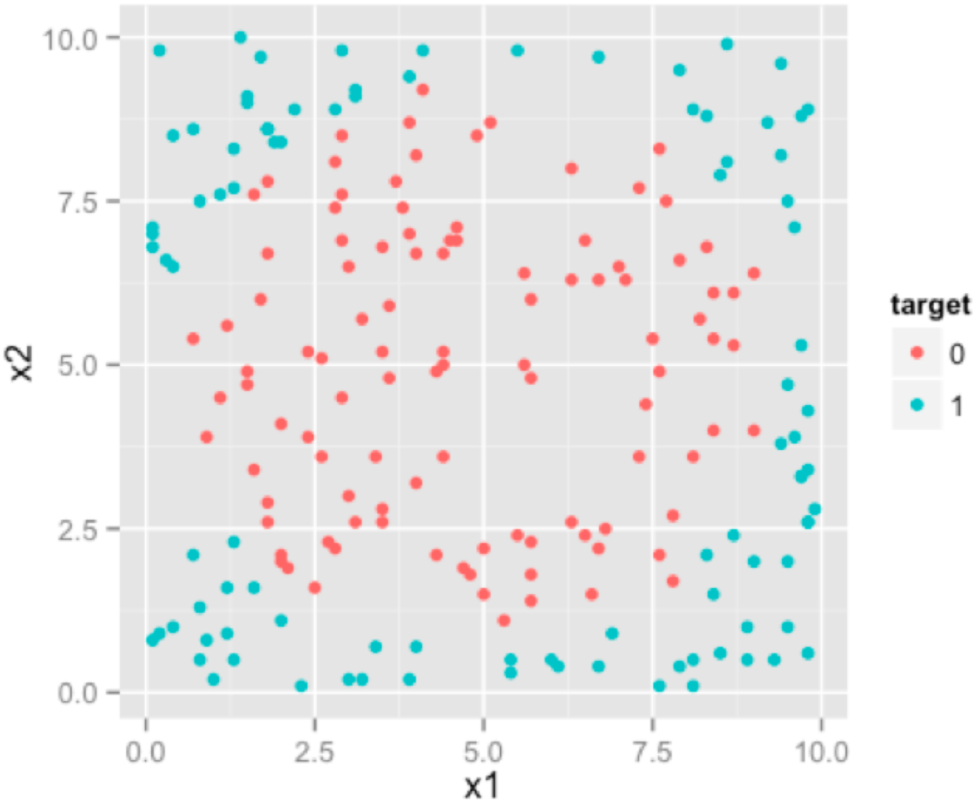
DECISION TREES

Decision Tree



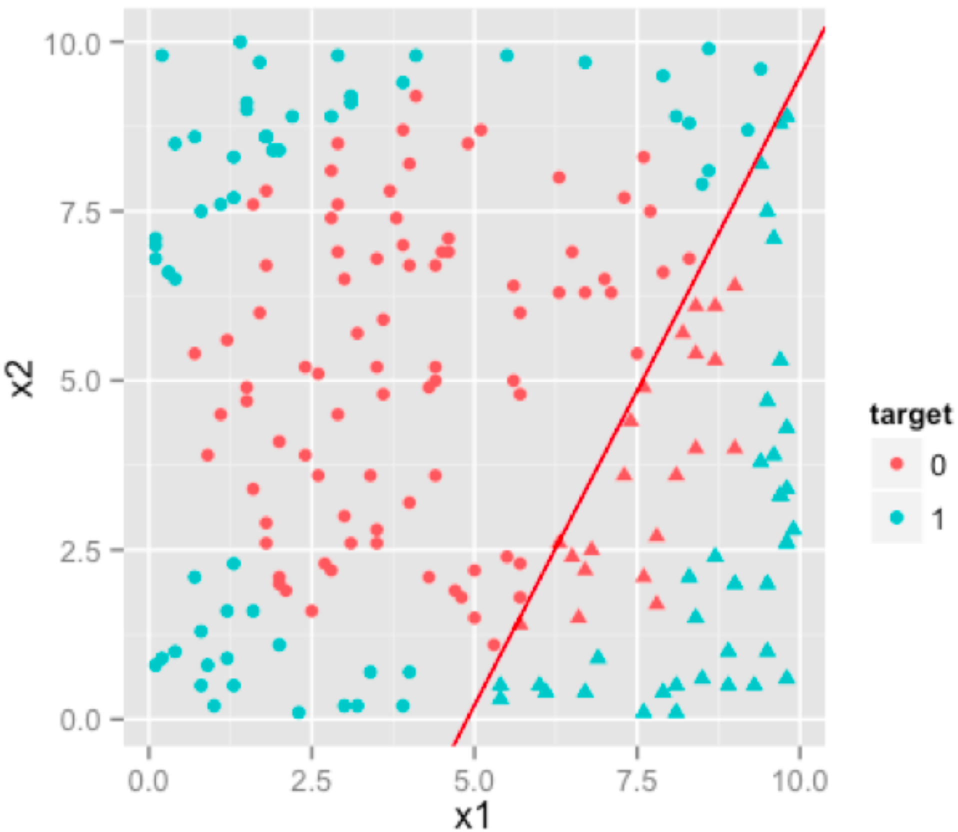
* Image comes from https://www.xoriant.com/blog/product-engineering/decision-trees-machine-learning-algorithm.html

Decision Tree vs. Linear Classifiers



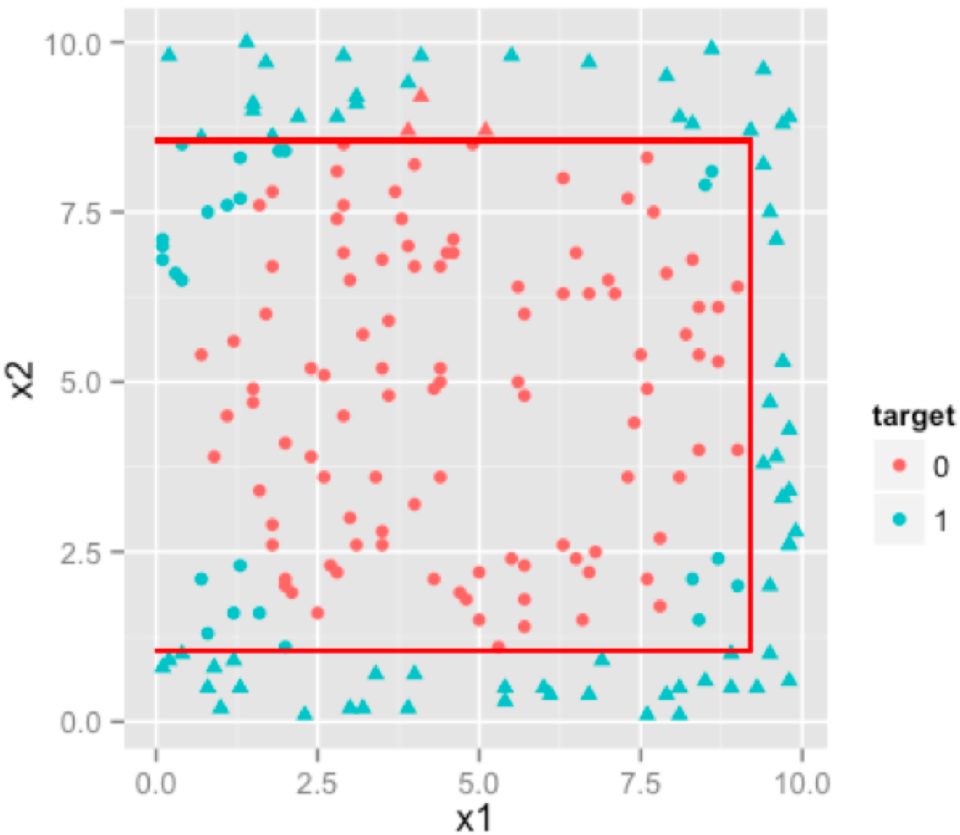
* Image comes from https://www.edvancer.in/logistic-regression-vs-decision-trees-vs-svm-part1/

Decision Tree vs. Linear Classifiers



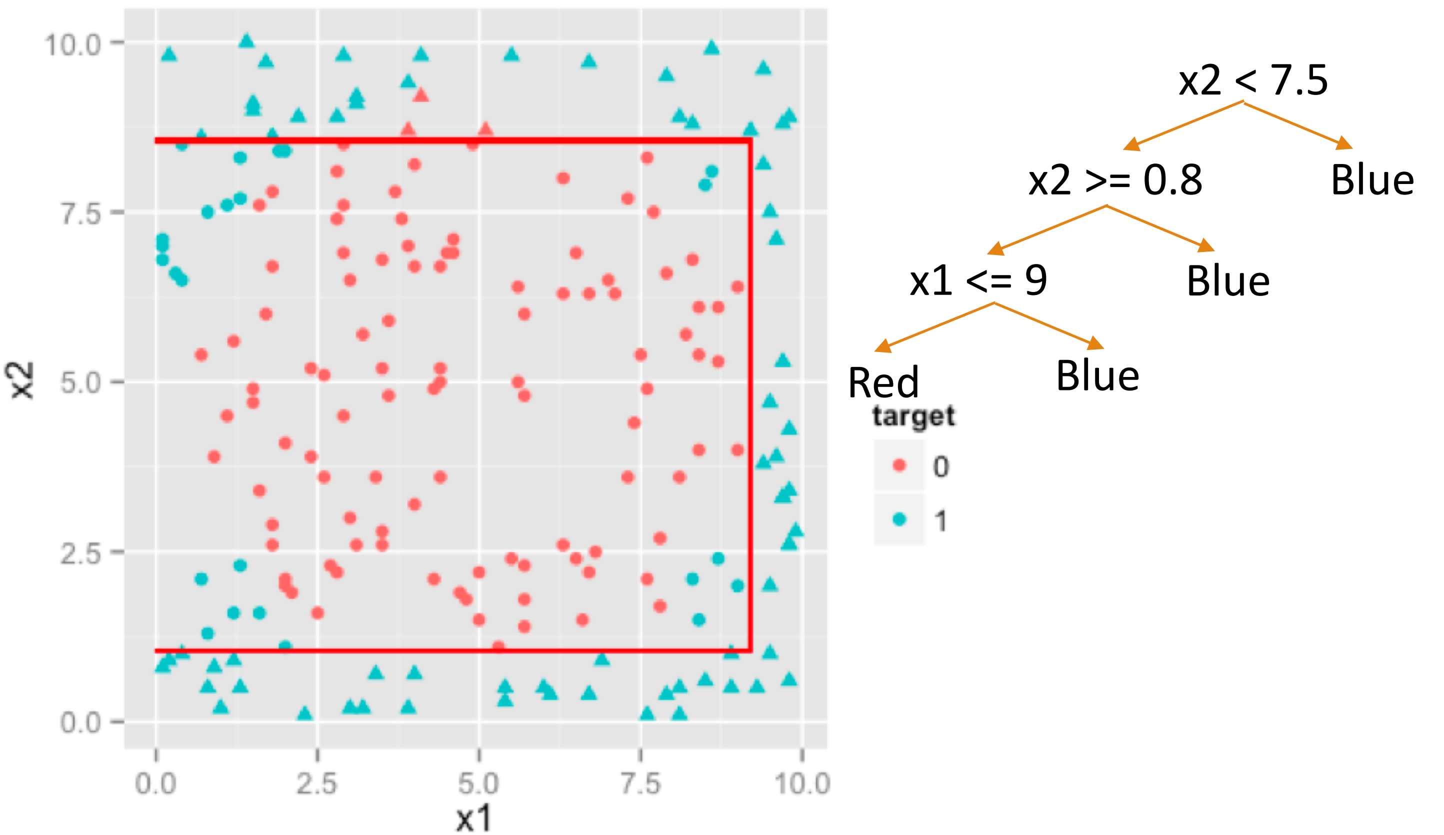
* Image comes from https://www.edvancer.in/logistic-regression-vs-decision-trees-vs-svm-part1/

Decision Tree vs. Linear Classifiers



* Image comes from https://www.edvancer.in/logistic-regression-vs-decision-trees-vs-svm-part1/

Decision Tree vs. Linear Classifiers



* Image comes from https://www.edvancer.in/logistic-regression-vs-decision-trees-vs-svm-part1/

Code for Decision Trees

Code with ScikitLearn:

```
model = DecisionTreeClassifier (min_samples_split=b)
model.fit(train_vecs, train.labels)
train_preds = model.predict(train_vecs)
```

In-Class Activity 1

PLAYING WITH LOGISTIC REGRESSION AND DECISION TREES

What about overfitting?

If the decision tree is too deep, more likely to overfit. Moreover, if the decision tree is too deep, then if you modify your training data even slightly, the decision tree changes drastically!

However if the decision tree is too shallow, becomes too simple of a model to be predictive

Like regularization, this is a parameter to tune!

Code with ScikitLearn:

```
model = DecisionTreeClassifier(max_depth=a, min_samples_split =b)
model.fit(train_vecs, train.labels)
train_preds = model.predict(train_vecs)
```

Ensemble Methods

COMBINING MULTIPLE CLASSIFIERS

Ensemble methods

Why learn one classifier when you can learn many?

Ensemble: combine many classifiers

Why do you think this would be useful?

Various options for getting help:

Why combine classifiers

- Each model has its own assumptions/pros/cons
- Combining them is like asking a panel of 5 doctors on which procedure to take instead of relying on one doctor (who has his/her own bias/experience/education)

Netflix Prize

Winning team combined hundreds of models together in an ensemble model!

Simple Majority Voting

Some python code to do this:

```
clf1 = DecisionTreeClassifier(....)

clf2 = KNeighborsClassifier(...)

clf3 = LogisticRegression(...)

eclf = VotingClassifier(estimators=[('dt', clf1 ), ('knn', clf2), ('lr', clf3)])
```

What are the pros and cons of this approach?

Remember classifiers give confidence scores?

If each classifier gives confidence scores, then weight each classifier's prediction with its confidence score!

Important assumption: each model would give different confidence scores for the same input data (at least for some portion of the input data)

Soft Voting

Some python code to do this:

```
clf1 = DecisionTreeClassifier(....)

clf2 = KNeighborsClassifier(...)

clf3 = LogisticRegression(...)

eclf = VotingClassifier(estimators=[('dt', clf1), ('knn', clf2), (‘lr', clf3)],
voting = ‘soft')
```

What are the pros and cons of this approach?

Soft Voting with weights

Some python code to do this:

```
clf1 = DecisionTreeClassifier(....)

clf2 = KNeighborsClassifier(...)

clf3 = LogisticRegression(...)

eclf = VotingClassifier(estimators=[('dt', clf1), ('knn', clf2), ('lr', clf3)],
voting = 'soft', weights=[2,1,3])
```

Add machine learning to machine learning?

Some python code to do this:

```
clf1 = DecisionTreeClassifier(....)

clf2 = KNeighborsClassifier(...)

clf3 = LogisticRegression(...)

eclf = VotingClassifier(estimators=[('dt', clf1), ('knn', clf2), ('lr', clf3)], voting = 'soft', weights=[2,1,3])
```

Could make these weights a hyperparameter that you learn or could feed the probabilities of each classifier into another classifier! (Stacked classifier)

Bagging

Bootstrap aggregation

- Learn many classifiers, each with only part of the data
- Combine through model averaging

Very helpful for decision trees which can overfit and be instable!

Because each classifier is training on different segments of data, they will model the data differently

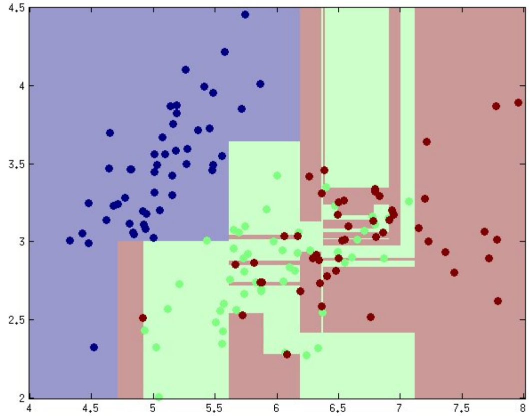
Bagging

Bootstrap
- Create a random subset of data by sampling
- Draw m’ of the m samples, with replacement
  - Some data left out; some data repeated several times

Bagging
- Repeat K times
  - Create a training set of m’ examples from the initial training set of m
  - Train a classifier on the random training set
- To test, run each trained classifier
  - Each classifier votes on the output, take majority

Some complexity control: harder for each to memorize data
  - Doesn’t work for linear models (average of linear functions is linear function…)
  - Works really well for decision trees
  - Don’t have to worry about those parameters for overfitting. Why?

Bagged decision trees



Full data set

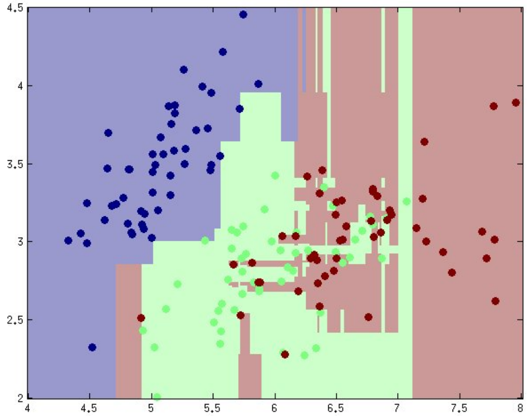
Average over collection

- Classification: majority vote

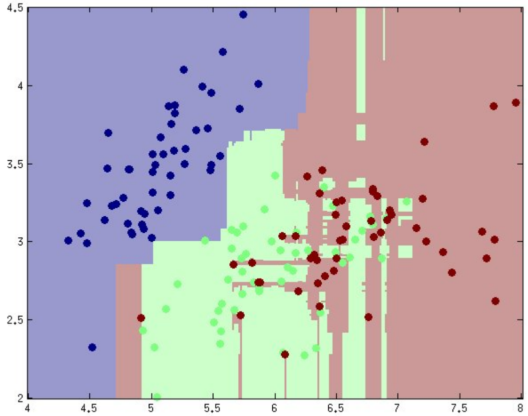
Reduces memorization effect

- Not every predictor sees each data point
- Lowers effective "complexity" of the overall average
- Usually, better generalization performance
- Intuition: reduces variance while keeping bias low

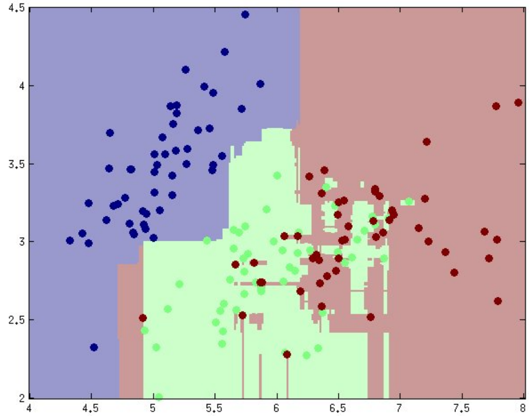
Avg of 5 trees



Avg of 25 trees



Avg of 100 trees



Random forests

Bagging applied to decision trees

Problem

- With lots of data, we can learn the same classifier -> Averaging doesn't help!

Introduce extra variation in learner

- At each step of training, only allow a subset of features
- Enforces diversity ("best" feature not available)
- Keeps bias low (every feature available eventually)
- Average over these learners (majority vote)

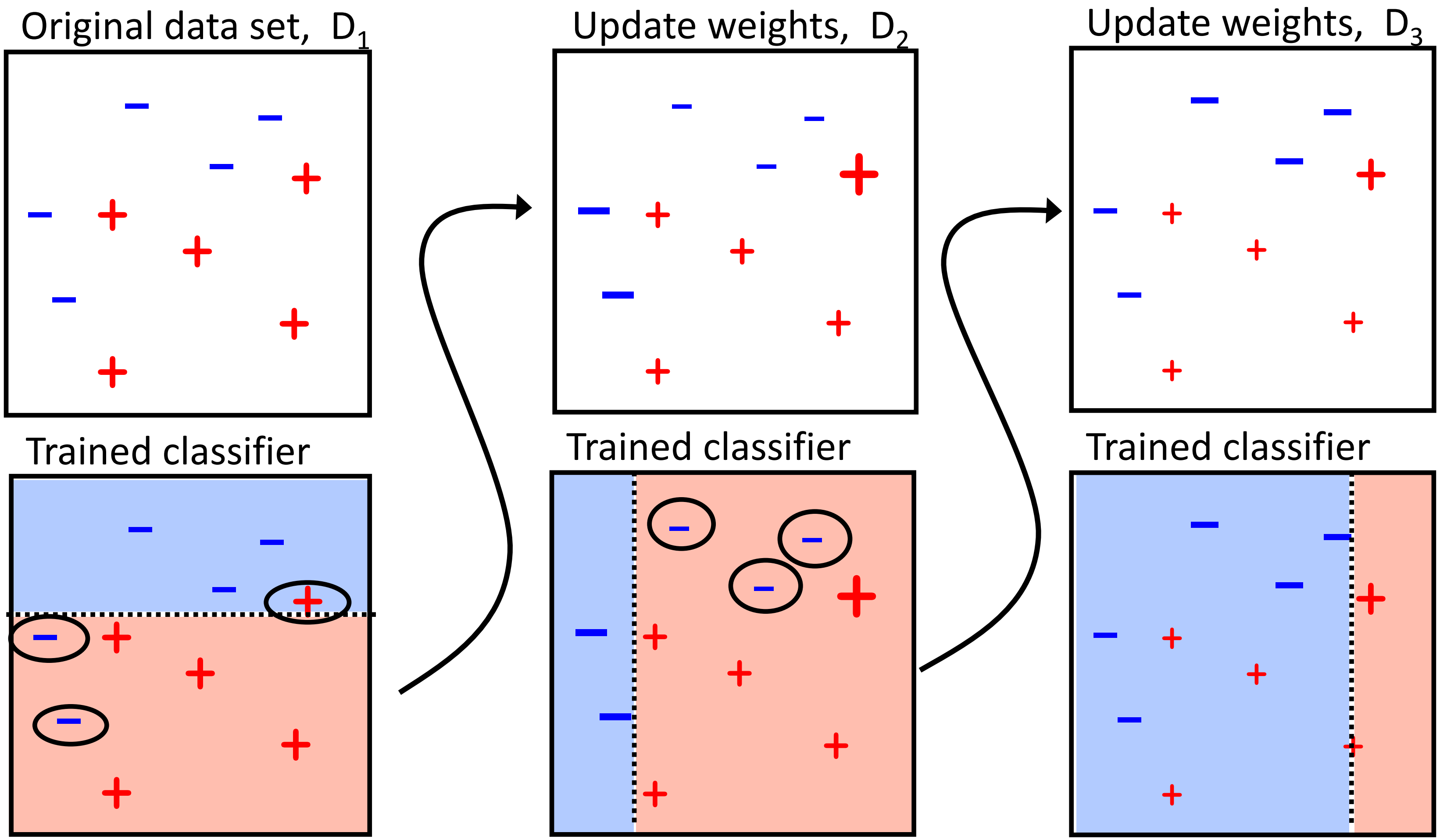
Code with ScikitLearn:

```
model = RandomForestClassifier(n_estimators=a)
model.fit(train_vecs, train.labels)
train_preds = model.predict(train_vecs)
```

Boosting

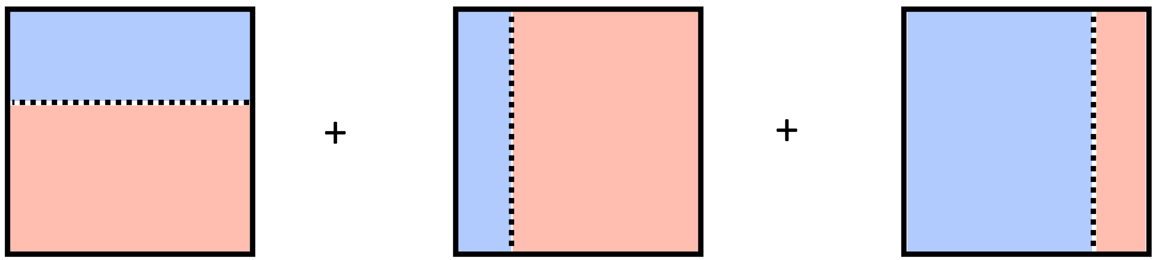
- Focus new learners on examples that others get wrong
- Train learners sequentially, rather than in parallel
- Errors of early predictions indicate the "hard" examples
- Focus later predictions on getting these examples right
- Combine the whole set in the end
- Convert many "weak" learners into a complex predictor

Boosting example

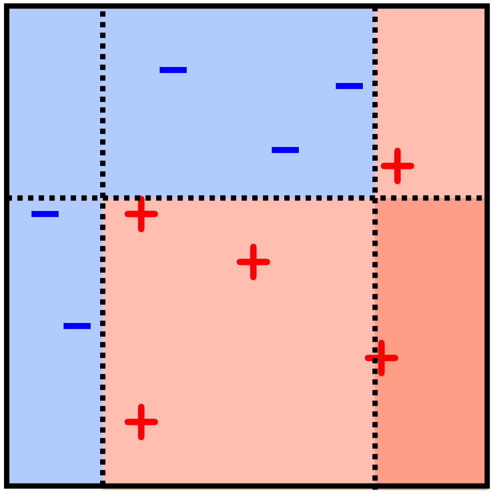


Boosting example

Weight each classifier and combine them:



Combined classifier



Code for Boosted Trees

Code with ScikitLearn:

```
model = GradientBoostingClassifier (n_estimators=a, min_samples_split=b)
model.fit(train_vecs, train.labels)
train_preds = model.predict(train_vecs)
```

When to use bagging vs. boosting?

If a classifier has 95% training error and 70% test, what would you use?

Bagging makes complex classifiers simple

Boosting makes simple classifiers complex

You have a very large training set and it takes a long time for a single model to train. What would you use?

Bagging is done in parallel so if you have access to multiple machines/CPUs, training can be done quickly.

Cons to ensemble methods?

In-Class Activity 2

PLAYING WITH ENSEMBLE METHODS