ENSEMBLE LEARNING

David Kauchak
CS451 – Fall 2013

Admin

Assignment 8
- 8.2 graded

Hadoop/MapReduce: was it worthwhile?

Final project

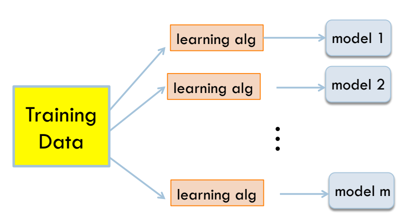
Ensemble learning

Basic idea: if one classifier works well, why not use multiple classifiers!

Ensemble learning

Basic idea: if one classifier works well, why not use multiple classifiers!

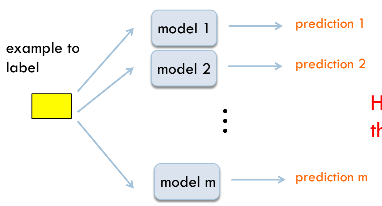
Training



Ensemble learning

Basic idea: if one classifier works well, why not use multiple classifiers!

Testing



How do we decide on the final prediction?

Ensemble learning

Basic idea: if one classifier works well, why not use multiple classifiers!

Testing

prediction 1

prediction 2

⋮

prediction m

- take majority vote
- if they output probabilities, take a weighted vote

How does having multiple classifiers help us?

Benefits of ensemble learning

Assume each classifier makes a mistake with some probability (e.g. 0.4, that is a 40% error rate)

Assuming the decisions made between classifiers are independent, what will be the probability that we make a mistake (i.e. error rate) with three classifiers for a binary classification problem?

Benefits of ensemble learning

Assume each classifier makes a mistake with some probability (e.g. 0.4, that is a 40% error rate)

model 1	model 2	model 3	prob
C	C	C	.6*.6*.6=0.216
C	C	I	.6*.6*.4=0.144
C	I	C	.6*.4*.6=0.144
C	I	I	.6*.4*.4=0.096
I	C	C	.4*.6*.6=0.144
I	C	I	.4*.6*.4=0.096
I	I	C	.4*.4*.6=0.096
I	I	I	.4*.4*.4=0.064

Benefits of ensemble learning

Assume each classifier makes a mistake with some probability (e.g. 0.4, that is a 40% error rate)

model 1	model 2	model 3	prob
C	C	C	.6*.6*.6=0.216
C	C	I	.6*.6*.4=0.144
C	I	C	.6*.4*.6=0.144
C	I	I	.6*.4*.4=0.096
I	C	C	.4*.6*.6=0.144
I	C	I	.4*.6*.4=0.096
I	I	C	.4*.4*.6=0.096
I	I	I	.4*.4*.4=0.064

0.096+
0.096+
0.096+
0.064 =
35% error!

Benefits of ensemble learning

3 classifiers in general, for r = probability of mistake for individual classifier:

$$p(error) = 3r^2(1-r) + r^3$$

binomial distribution

r	p(error)
0.4	0.35
0.3	0.22
0.2	0.10
0.1	0.028
0.05	0.0073

Benefits of ensemble learning

5 classifiers in general, for r = probability of mistake for individual classifier:

$$p(error) = 10r^3(1-r)^2 + 5r^4(1-r) + r^5$$

r	p(error) 3 classifiers	p(error) 5 classifiers
0.4	0.35	0.32
0.3	0.22	0.16
0.2	0.10	0.06
0.1	0.028	0.0086
0.05	0.0073	0.0012

Benefits of ensemble learning

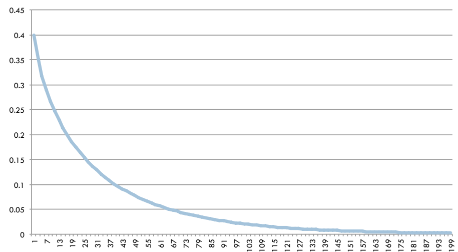
m classifiers in general, for r = probability of mistake for individual classifier:

$$p(error) = \sum_{i=(m+1)/2}^{m} \binom{m}{i} r^i (1-r)^{m-i}$$

(cumulative probability distribution for the binomial distribution)

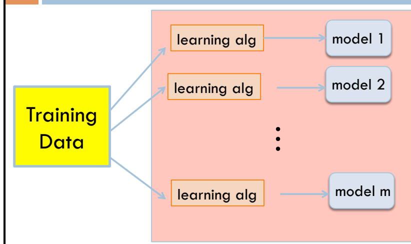
Given enough classifiers…

$$p(error) = \sum_{i=(m+1)/2}^{m} \binom{m}{i} r^i (1-r)^{m-i}$$



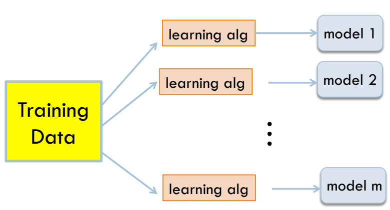
r = 0.4

Obtaining independent classifiers



Where to we get *m* independent classifiers?

Idea 1: different learning methods



decision tree

k-nn

perceptron

naïve bayes

gradient descent variant 1

gradient descent variant 2

⋮

Pros/cons?

Idea 1: different learning methods

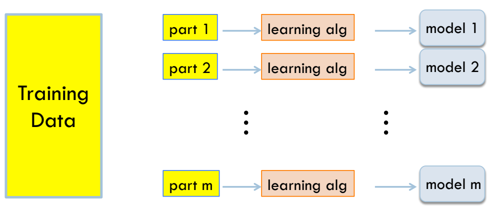
Pros:

- Lots of existing classifiers already
- Can work well for some problems

Cons/concerns:

- Often, classifiers are not independent, that is, **they make the same mistakes!**
  - e.g. many of these classifiers are linear models
  - voting won't help us if they're making the same mistakes

Idea 2: split up training data



Use the same learning algorithm, but train on different parts of the training data

Idea 2: split up training data

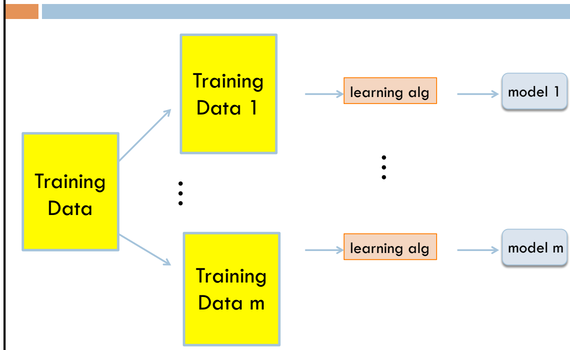
Pros:

- Learning from different data, so can't overfit to same examples
- Easy to implement
- fast

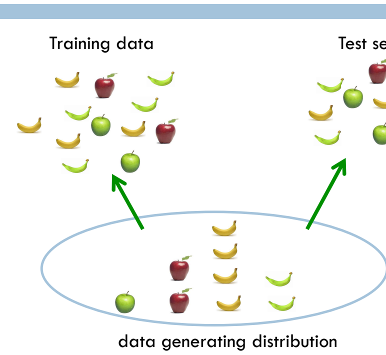
Cons/concerns:

- Each classifier is only training on a small amount of data
- Not clear why this would do any better than training on full data and using good regularization

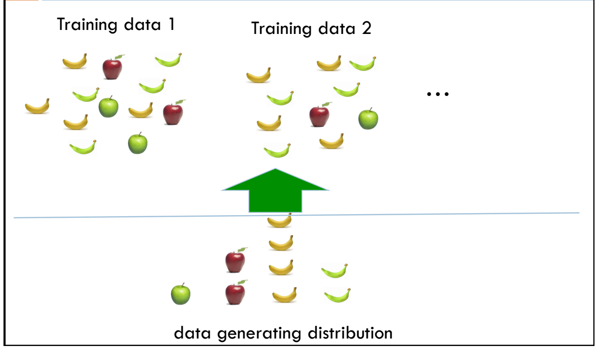
Idea 3: bagging



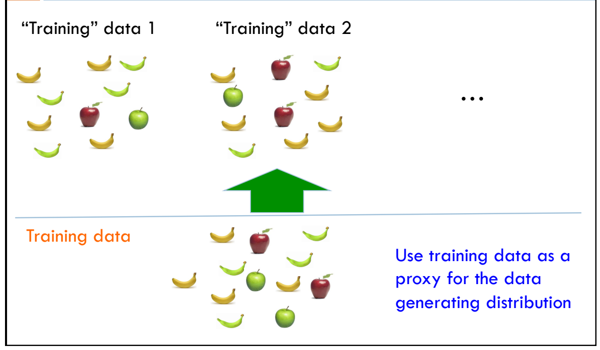
data generating distribution



Ideal situation

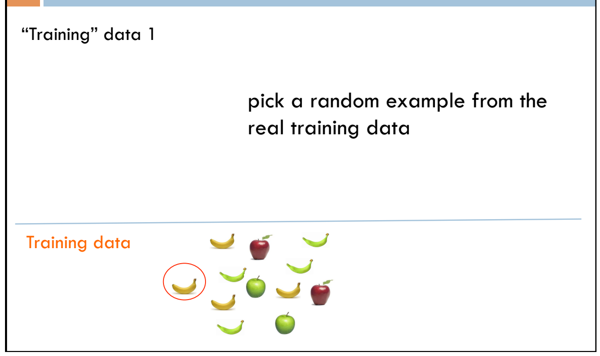


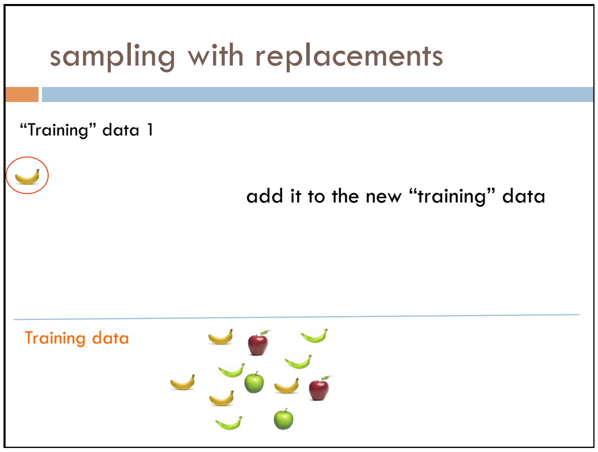
bagging

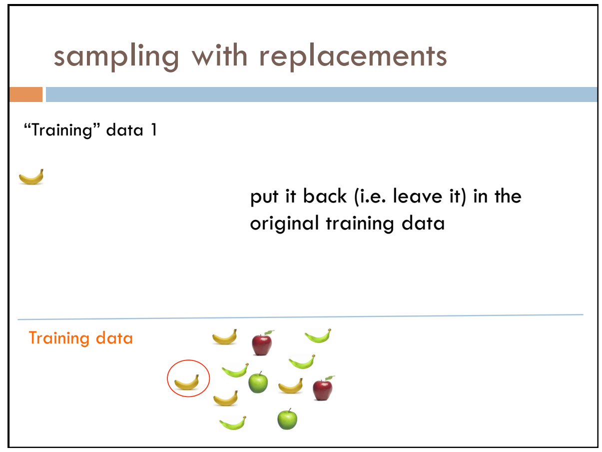


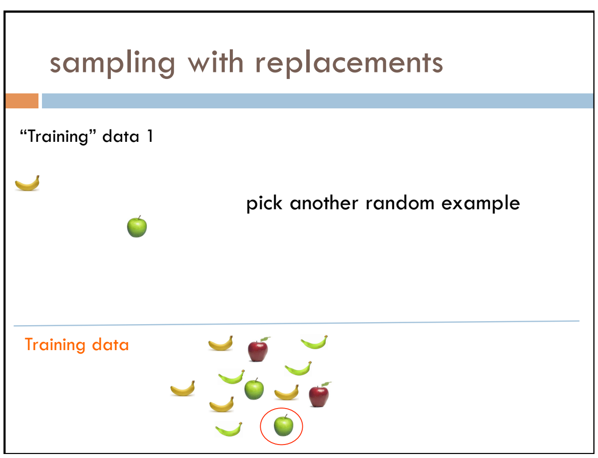
sampling with replacements

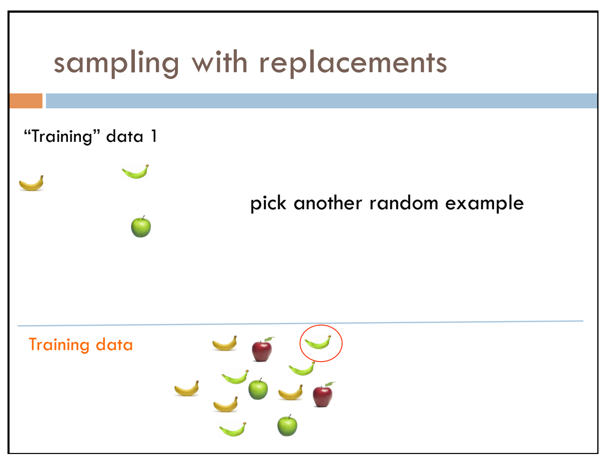
sampling with replacements




sampling with replacements
"Training" data 1
add it to the new "training" data
Training data

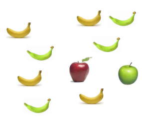

sampling with replacements
"Training" data 1
put it back (i.e. leave it) in the original training data
Training data


sampling with replacements
"Training" data 1
pick another random example
Training data


sampling with replacements
"Training" data 1
pick another random example
Training data

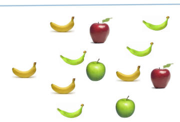
sampling with replacements

"Training" data 1



keep going until you've created a new "training" data set

Training data



bagging

create m "new" training data sets by sampling with replacement from the original training data set (called *m* "bootstrap" samples)

train a classifier on each of these data sets

to classify, take the majority vote from the *m* classifiers

bagging concerns

Training Data

Training Data 1

⋮

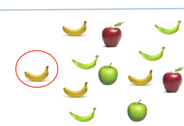
Training Data m

Won't these all be basically the same?

bagging concerns

For a data set of size n, what is the probability that a given example will **NOT** be select in a "new" training set sampled from the original?

Training data

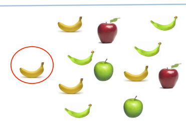


bagging concerns

What is the probability it isn't chosen the first time?

$$1 - 1/n$$

Training data



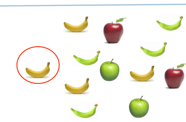
bagging concerns

What is the probability it isn't chosen the ***any*** of the n times?

$$(1 - 1/n)^n$$

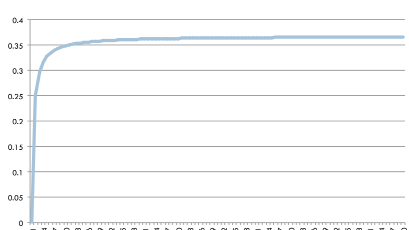
Each draw is independent and has the same probability

Training data



probability of overlap

$$(1 - 1/n)^n$$



Converges very quickly to 1/e ≈ 63%

bagging overlap

Training Data

Training Data 1

⋮

Training Data m

Won't these all be basically the same?

On average, a randomly sampled data set will only contain 63% of the examples in the original

When does bagging work

Let's say 10% of our examples are noisy (i.e. don't provide good information)

For each of the "new" data set, what proportion of noisy examples will they have?
- They'll still have ~10% of the examples as noisy
- However, these examples will only represent about a third of the original noisy examples

For some classifiers that have trouble with noisy classifiers, this can help

When does bagging work

Bagging tends to reduce the *variance* of the classifier

By voting, the classifiers are more robust to noisy examples

Bagging is most useful for classifiers that are:
- Unstable: small changes in the training set produce very different models
- Prone to overfitting

Often has similar effect to regularization