Ensemble Methods

Charles Sutton
Data Mining and Exploration
Spring 2012

Bias and Variance

Consider a regression problem

$$Y = f(X) + \epsilon \qquad \epsilon \sim N(0, \sigma^2)$$

With an estimate regression function $\hat{f}$, e.g.,

$$\hat{f}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$$

Suppose we care about the error at a particular $\mathbf{x}_0$

$$L(y, \hat{f}(\mathbf{x}_0)) = (y - \hat{f}(\mathbf{x}_0))^2$$

Let's think about the expected error:

$$E(L(y, \hat{f}(\mathbf{x}_0))) = \int_{-\infty}^{\infty} p(y|x_0)(y - \hat{f}(\mathbf{x}_0))^2 dy$$

Important: both y and $\hat{f}$ are random!

Bias and Variance

Consider a regression problem

$$Y = f(X) + \epsilon \qquad \epsilon \sim N(0, \sigma^2)$$

Let's think about the expected error:

$$E(L(y, \hat{f}(\mathbf{x}_0))) = E(y - \hat{f}(\mathbf{x}_0))^2$$

...after some algebra...

$$= \sigma^2 + \mathrm{Bias}^2(\hat{f}(\mathbf{x}_0)) + V\hat{f}(\mathbf{x}_0)$$

where

$$\mathrm{Bias}(\hat{f}(\mathbf{x}_0)) = E(f(\mathbf{x}_0) - \hat{f}(\mathbf{x}_0))$$

expectation taken over both y and $\hat{f}$

Bias and Variance

Stable classification methods:
- Lower variance
- Higher bias

Flexible methods
- Higher variance
- Lower bias

Like to minimize both, but often must trade off.

Data drawn from $p(\mathbf{x}, y)$

Bias and Variance

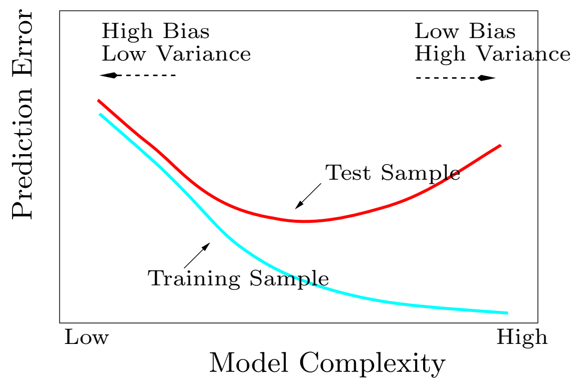


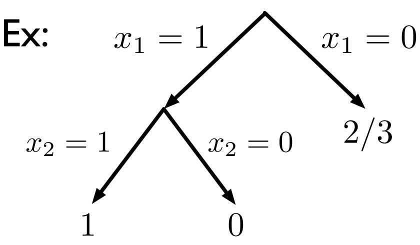
Figure from [Hastie, Tibshirani, and Friedman, 2009]

Notation

Feature vector: $x \in \mathbb{R}^n$

Classifier output: $h(x) \in \{-1, 1\}$

Ex:



$h([0, 1, 1]) = 1$

What is an ensemble?

- A group of classifiers that vote (perhaps weighted) on the answer

$$h(x) = \sum_{i=1}^{M} \beta_m h_m(x)$$

individual classifiers

weights

Why an ensemble?

- Smooth the variance of unstable classifiers
- Combine classifiers with different biases
- Different classifiers can "specialise" in different parts of the input space

Bagging

- We said decision trees are unstable
- Let's generate a bunch of data sets, and average the results!

$$h(x) = \sum_{i=1}^{M} \beta_m h_m(x)$$

each data set

probability from decision tree

The bootstrap

- But where do we get all of those data sets?
- Crazy idea: Let's get them from the training data. *Resample with replacement*

Example data

IDX	OUTLOOK	TEMP	HUMIDITY	WIND	PLAY
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

Resampled

IDX	OUTLOOK	TEMP	HUMIDITY	WIND	PLAY
1	sunny	hot	high	weak	no
1	sunny	hot	high	weak	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
4	rain	mild	high	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
7	overcast	cool	normal	strong	yes
7	overcast	cool	normal	strong	yes
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
11	sunny	mild	normal	strong	yes

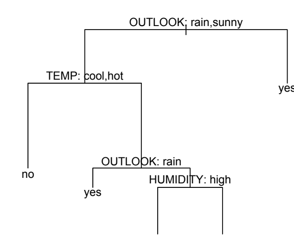
Example data

IDX	OUTLOOK	TEMP	HUMIDITY	WIND	PLAY
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

Resampled

IDX	OUTLOOK	TEMP	HUMIDITY	WIND	PLAY
1	sunny	hot	high	weak	no
1	sunny	hot	high	weak	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
4	rain	mild	high	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
7	overcast	cool	normal	strong	yes
7	overcast	cool	normal	strong	yes
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
11	sunny	mild	normal	strong	yes

Run decision tree learning



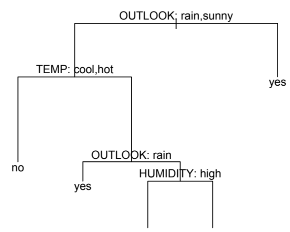
Example data

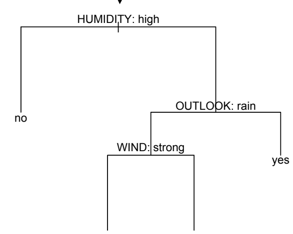
IDX	OUTLOOK	TEMP	HUMIDITY	WIND	PLAY
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

Resampled

IDX	OUTLOOK	TEMP	HUMIDITY	WIND	PLAY
1	sunny	hot	high	weak	no
1	sunny	hot	high	weak	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
4	rain	mild	high	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
7	overcast	cool	normal	strong	yes
7	overcast	cool	normal	strong	yes
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
11	sunny	mild	normal	strong	yes

Run decision tree learning

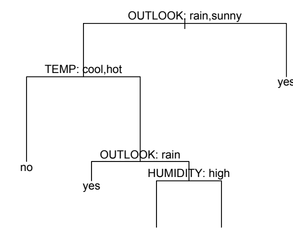


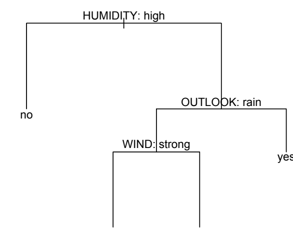


Example data

IDX	OUTLOOK	TEMP	HUMIDITY	WIND	PLAY
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

Resampled





[one tree for each resampled data set]
....

Back to bagging

INPUT: D denotes training data, of size N
for j from $1 \dots M$ **do**
  Sample data D_j of size N from D with replacement
  Train classifier h_j on D_j
end for
Return a new classifier h that classifies new examples x as $h(x) = \sum_{j=1}^{M} h_j(x)$

How bagging can help

Table 2. Misclassification Rates (%)

Data Set	$\bar{e}_S$	$\bar{e}_B$	**Decrease**
waveform	29.1	19.3	34%
heart	4.9	2.8	43%
breast cancer	5.9	3.7	37%
ionosphere	11.2	7.9	29%
diabetes	25.3	23.9	6%
glass	30.4	23.6	22%
soybean	8.6	6.8	21%

S: decision tree, B: bagging

[Breiman, 1996]

Example: Glass data set

Standard data set from UCI ML repository

7 classes, such as:
- building windows
- vehicle windows
- headlamps

10 features such as
- % Na by weight
- % Al by weight
- refractive index

RI	Na	Mg	Al		CLASS
1.51793	12.79	3.5	1.12	...	building (float)
1.51643	12.16	3.52	1.35	...	vehicle (float)
1.51793	13.21	3.48	1.41	...	building (float)
...					

When to bag

- Bagging decision trees usually helps
  - (but random forests, boosting better)
- Classifier needs to be unstable
- Bagging 1-nearest neighbour not so helpful

Boosting

- Idea was to transform a "weak learner" into a strong one
- The only requirement for a weak learner is that its accuracy is slightly above 50% (in two-class case)
- Examples of weak learners:
  - decision "stumps", naive Bayes

Ideas behind boosting

- Boosting is a general term for methods that try to "amplify" a weak learner into a better one.
- Rather than picking different training sets, reweight the training set
- Pick the weights based on which examples were misclassified previously

Weighted Examples

Up until now, our data sets have been

$$\mathcal{D} = \{\mathbf{x}_i, y_i \,|\, i \in [1, N]\}$$

Now we need to handle weighted data sets

$$\mathcal{D} = \{D(i), \mathbf{x}_i, y_i \,|\, i \in [1, N]\}$$

where $D(i)$ is a distribution over instances

$$D(i) \geq 0 \qquad \sum_{i=1}^{N} D(i) = 1$$

Most classifiers can handle this, no problem.
(how would decision trees?)

AdaBoost.M1

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train base learner using distribution D_t.
- Get base classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final classifier:

$$H(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Updating weights

- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

$$\mathrm{err} = \sum_{i=1}^{N} D_t(i) I(y_i \neq h_t(x_i))$$

Use the following "magic" choice

$$\alpha_t = \frac{1}{2} \log \frac{1 - \mathrm{err}}{\mathrm{err}}$$

Ex: Boosted decision stumps

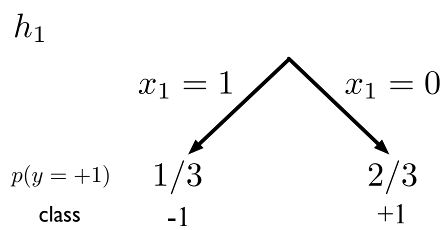
y	x_1	x_2	x_3
1	1	1	0
-1	1	0	1
-1	0	0	1
1	0	0	1
1	0	1	1
-1	1	0	0

$$D_1(i) = 1/6 \text{ for } i \in \{1, 2, \ldots 6\}$$

Let's do the first iteration of boosting on this data set

Ex: Boosted decision stumps

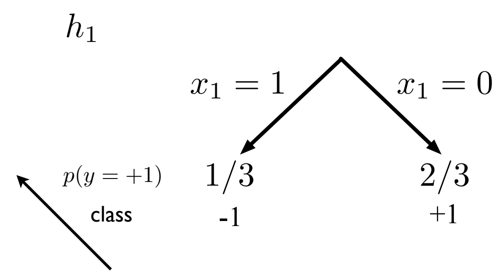
y	x_1	x_2	x_3
1	1	1	0
-1	1	0	1
-1	0	0	1
1	0	0	1
1	0	1	1
-1	1	0	0



$$D_1(i) = 1/6 \text{ for } i \in \{1, 2, \ldots 6\}$$

Ex: Boosted decision stumps

y	x_1	x_2	x_3
1	1	1	0
-1	1	0	1
-1	0	0	1
1	0	0	1
1	0	1	1
-1	1	0	0



$$D_1(i) = 1/6 \text{ for } i \in \{1, 2, \ldots 6\}$$

$$\text{err} = 0.3333$$

$$\alpha_t = \frac{1}{2} \log \frac{1 - \text{err}}{\text{err}} = \frac{1}{2} \log 2 = 0.3465$$

$$\exp\{\alpha_t\} = 1.414 \qquad \exp\{-\alpha_t\} = 0.707$$

Ex: Boosted decision stumps

y	x_1	x_2	x_3	$\exp\{-\alpha_1 y_i h_1(x_i)\}$	$D_1(i)$
1	1	1	0	1.414	0.25
-1	1	0	1	0.707	0.125
-1	0	0	1	1.414	0.25
1	0	0	1	0.707	0.125
1	0	1	1	0.707	0.125
-1	1	0	0	0.707	0.125

SUM=5.65

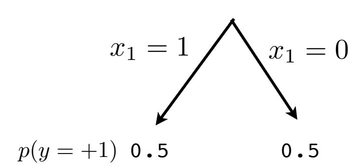
$$\exp\{\alpha_t\} = 1.414 \qquad \exp\{-\alpha_t\} = 0.707$$

Ex: Iteration 2

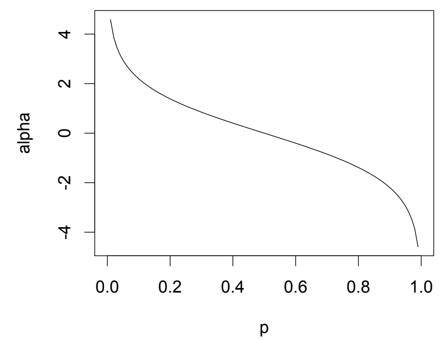
y	x_1	x_2	x_3	$D_1(i)$
1	1	1	0	0.25
-1	1	0	1	0.125
-1	0	0	1	0.25
1	0	0	1	0.125
1	0	1	1	0.125
-1	1	0	0	0.125

Now induce another decision stump, with examples weighted by D1

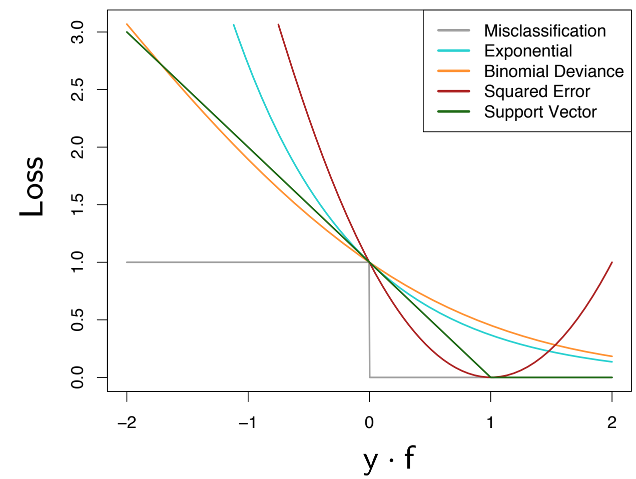
e.g., if split on x1



$$\alpha_t = \log \frac{1 - \text{err}}{\text{err}}$$



Loss Functions



Loss functions

Can view AdaBoost as approximately minimizing prediction error

i.e., We want to learn an ensemble

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

that minimizes training error

$$\text{Err} = \frac{1}{N} \sum_{i=1}^{N} I(y_i \neq H(x_i))$$

with respect to $\{\alpha_t\}, \{h_t\}$

Difficult to optimize directly (e.g., why not gradient descent?) so we approximate...

Loss functions

AdaBoost trick: Minimize upper bound on error

$$\frac{1}{N} \sum_{i=1}^{N} I(y_i \neq H(x_i)) \leq \frac{1}{N} \sum_{i=1}^{N} \exp\{-y_i H(x_i)\}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \exp\left\{-y_i \sum_t \alpha_t h_t(x_i)\right\} \quad = \prod_t Z_t$$

$$:= L(\alpha_1, \alpha_2, \ldots, \alpha_t, h_1, h_2, \ldots h_t)$$

(!!!)

We still can't minimise this exactly, so be greedy. Alternately minimise with respect to alpha and h_t.

AdaBoost.M1

Minimizes L wrt h_t

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train base learner using distribution D_t.
- Get base classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$$\alpha_t = \frac{1}{2} \log \frac{1 - \text{err}}{\text{err}}$$

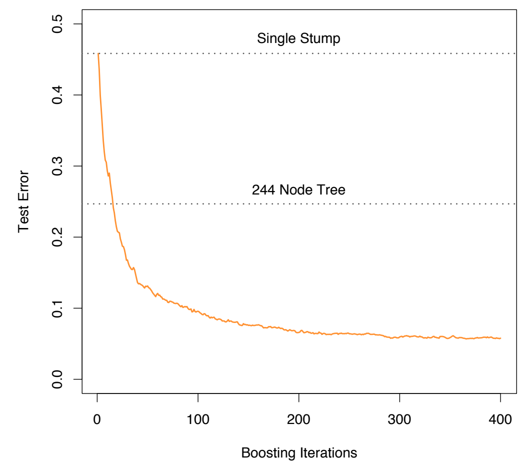
Minimizes L wrt α_t

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

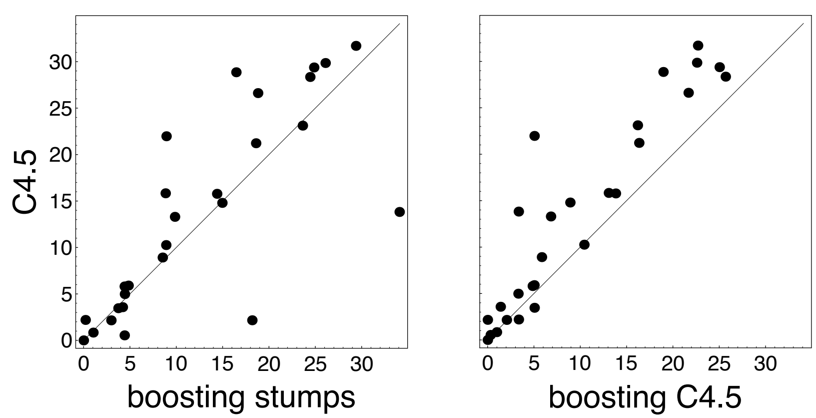
Output the final classifier:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Boosting can help



Boosting can help



On glass data

Standard DT	61%
Bagged DT	68%
AdaBoost	70%

Differences between bagging and boosting

- Bagging for unstable (i.e., high variance) classifiers
- Boosting often useful for biased classifiers as well
- Both improve performance
- Both need to choose base classifier
- Boosting typically performs better
- Both lose interpretability

Heterogeneous ensembles

- Data mining competitions usually won by ensemble methods (e.g., Netflix)
- Often the classifiers are completely heterogeneous

Examinable reading (on Web site):

Rob Shapire, The Boosting Approach to Machine Learning (Sections 4-8 not examinable)

Leo Breiman, Bagging predictors, Machine Learning, 1996