Machine Learning

A. G. Schwing & M. Telgarsky

February 15, 2018

Slides heavily using material from Daniel Hsu (Columbia)!
Thanks!

L10: Ensemble Methods.

Slides heavily using material from Daniel Hsu (Columbia)!
Thanks!

Lecture outline.

1. Brief review.
2. Combining classifiers with majority vote.
3. Interlude: decision trees.
4. Practical majority vote: bagging and random forests.
5. Non-independent errors with majority vote: boosting.

Reading.

- K. Murphy; Machine Learning: A Probabilistic Perspective; Chapter 16.

Brief review.

Classifiers we've seen so far.

- Nearest neighbor.
  (Lecture 1.)
- Linear predictors: least squares, logistic regression, SVM.
  (Lectures 2, 3, 6.)
- Kernel (nonlinear) SVM.
  (Lecture 7.)
- Neural networks.
  (Lectures 8, 9.)

Brief review.

Classifiers we've seen so far.

- Nearest neighbor.
  (Lecture 1.)
- Linear predictors: least squares, logistic regression, SVM.
  (Lectures 2, 3, 6.)
- Kernel (nonlinear) SVM.
  (Lecture 7.)
- Neural networks.
  (Lectures 8, 9.)

Suppose we train one of each.
Do we choose best and throw rest away?
Can we somehow combine?

Combining classifiers with majority vote.

Why ensembles?

Standard machine learning practice:
We have some data, we try 10 different predictors.
(3-nn, least squares, logistic regression, SVM, some deep nets, . . .)

Rather than taking the *best*, can we combine them and do better?

Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

Combining classifiers.

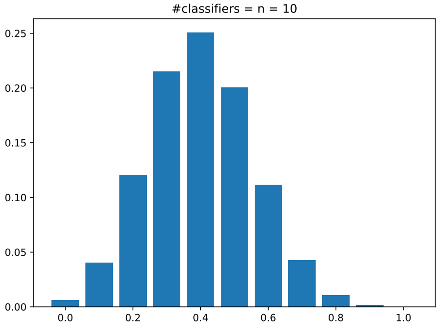
Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

We can model the distribution of errors with $\text{Binom}(n, 0.4)$.

Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

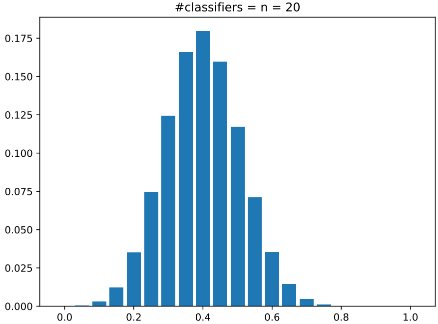
We can model the distribution of errors with $\text{Binom}(n, 0.4)$.



Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

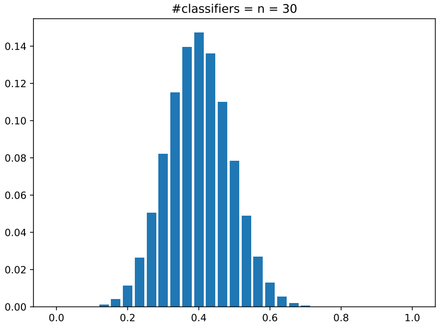
We can model the distribution of errors with $\text{Binom}(n, 0.4)$.



Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

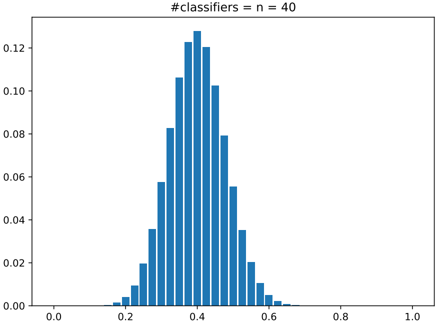
We can model the distribution of errors with $\text{Binom}(n, 0.4)$.



Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

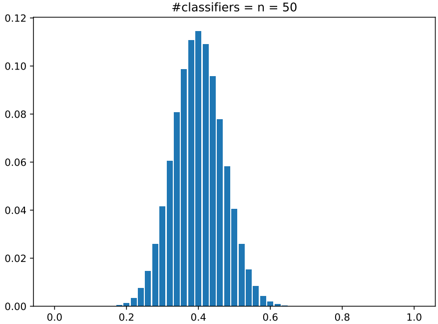
We can model the distribution of errors with $\mathrm{Binom}(n, 0.4)$.



Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

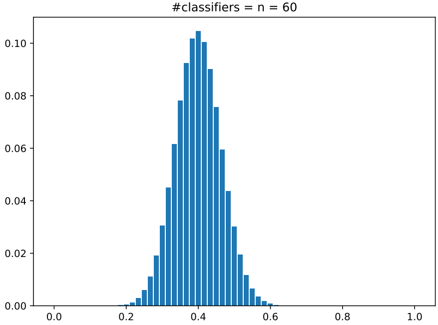
We can model the distribution of errors with $\text{Binom}(n, 0.4)$.



Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

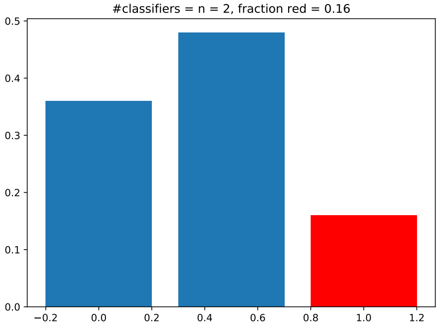
We can model the distribution of errors with $\text{Binom}(n, 0.4)$.



Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

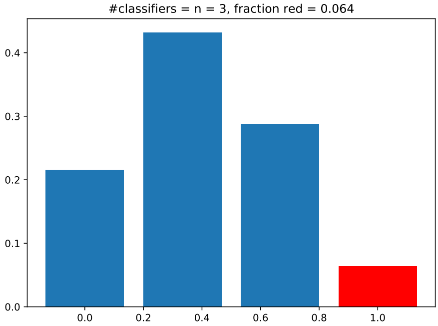
We can model the distribution of errors with $\text{Binom}(n, 0.4)$.
Red: *all* wrong.



Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

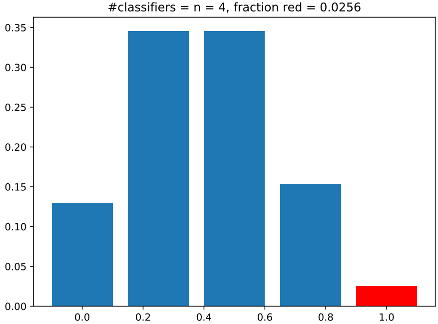
We can model the distribution of errors with $\text{Binom}(n, 0.4)$.
Red: *all* wrong.



Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

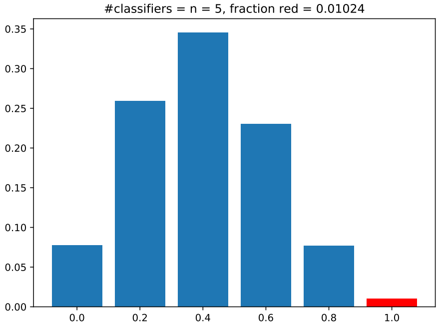
We can model the distribution of errors with $\text{Binom}(n, 0.4)$.
Red: *all* wrong.



Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

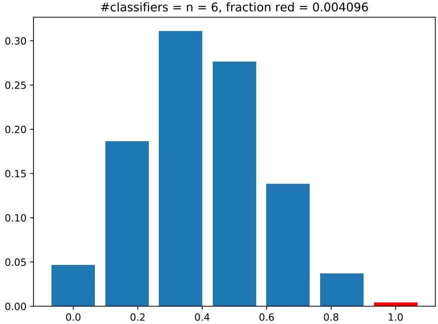
We can model the distribution of errors with $\text{Binom}(n, 0.4)$.
Red: *all* wrong.



Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

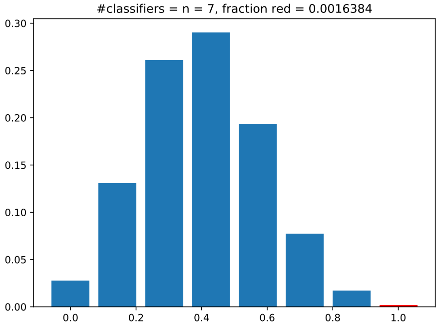
We can model the distribution of errors with $\text{Binom}(n, 0.4)$.
Red: *all* wrong.



Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

We can model the distribution of errors with $\text{Binom}(n, 0.4)$.
Red: *all* wrong.

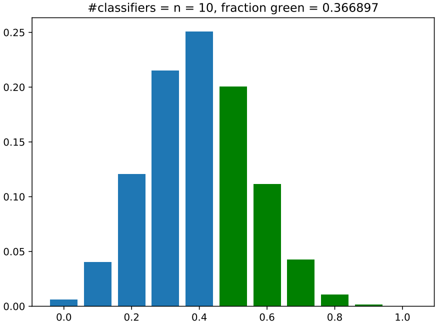


Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

We can model the distribution of errors with $\text{Binom}(n, 0.4)$.

Green: *at least half* wrong.

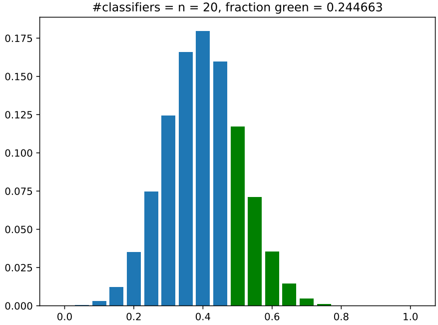


Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

We can model the distribution of errors with Binom$(n, 0.4)$.

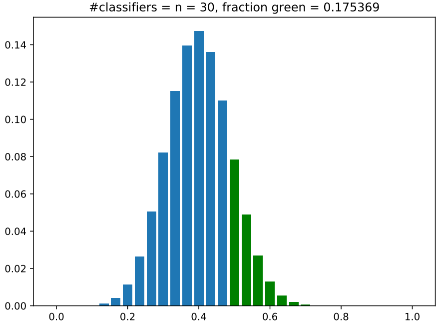
Green: *at least half* wrong.



Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ $(\mathbb{E}(X_i) = 0.4)$.

We can model the distribution of errors with $\mathrm{Binom}(n, 0.4)$.
Green: *at least half* wrong.

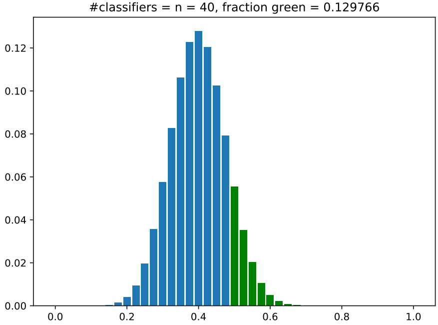


Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

We can model the distribution of errors with $\text{Binom}(n, 0.4)$.

Green: *at least half* wrong.

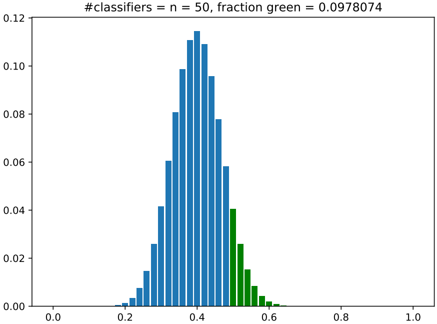


Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

We can model the distribution of errors with $\text{Binom}(n, 0.4)$.

Green: *at least half* wrong.

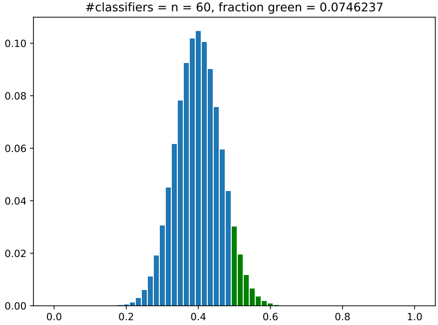


Combining classifiers.

Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^n$ ($\mathbb{E}(X_i) = 0.4$).

We can model the distribution of errors with $\text{Binom}(n, 0.4)$.

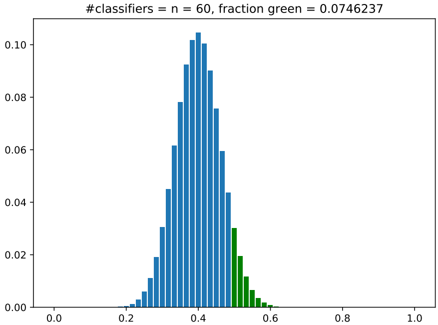
Green: *at least half* wrong.



Combining classifiers.

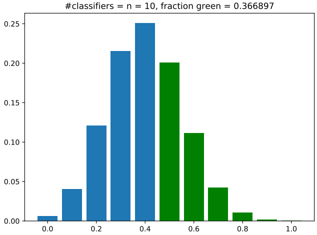
Suppose we have n classifiers.
Suppose each is wrong *independently* with probability 0.4.
Model error of classifiers as random variables $(X_i)_{i=1}^{n}$ $(\mathbb{E}(X_i) = 0.4)$.

We can model the distribution of errors with $\text{Binom}(n, 0.4)$.
Green: *at least half* wrong.



Green region is error of majority vote! $0.075 \ll 0.4$!!!

Majority vote.



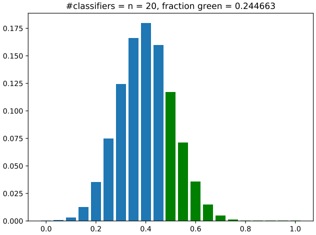
Green region is error of majority vote!
Suppose $y_i \in \{-1, +1\}$.

$$\mathrm{MAJ}(y_1, \ldots, y_n) := \begin{cases} +1 & \text{when } \sum_i y_i \geq 0, \\ -1 & \text{when } \sum_i y_i < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability p):

$$\Pr[\mathrm{Binom}(n, p) \geq n/2] = \sum_{i=n/2}^{n} \binom{n}{i} p^i (1-p)^{n-i} \leq \exp\left(-n(1/2 - p)^2\right).$$

Majority vote.



Green region is error of majority vote!

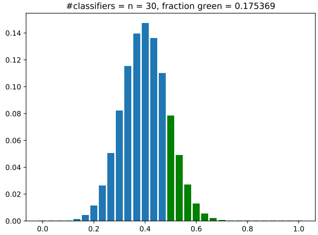
Suppose $y_i \in \{-1, +1\}$.

$$\mathrm{MAJ}(y_1, \ldots, y_n) := \begin{cases} +1 & \text{when } \sum_i y_i \geq 0, \\ -1 & \text{when } \sum_i y_i < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability p):

$$\Pr[\mathrm{Binom}(n,p) \geq n/2] = \sum_{i=n/2}^{n} \binom{n}{i} p^i (1-p)^{n-i} \leq \exp\left(-n(1/2-p)^2\right).$$

Majority vote.



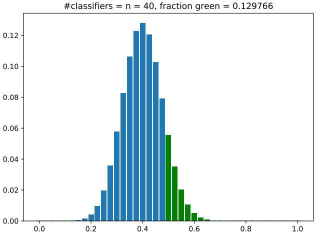
Green region is error of majority vote!
Suppose $y_i \in \{-1, +1\}$.

$$\mathrm{MAJ}(y_1, \ldots, y_n) := \begin{cases} +1 & \text{when } \sum_i y_i \geq 0, \\ -1 & \text{when } \sum_i y_i < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability p):

$$\Pr[\mathrm{Binom}(n, p) \geq n/2] = \sum_{i=n/2}^{n} \binom{n}{i} p^i (1-p)^{n-i} \leq \exp\left(-n(1/2 - p)^2\right).$$

Majority vote.



Green region is error of majority vote!

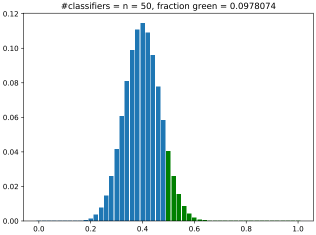
Suppose $y_i \in \{-1, +1\}$.

$$\mathrm{MAJ}(y_1, \ldots, y_n) := \begin{cases} +1 & \text{when } \sum_i y_i \geq 0, \\ -1 & \text{when } \sum_i y_i < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability p):

$$\Pr[\mathrm{Binom}(n, p) \geq n/2] = \sum_{i=n/2}^{n} \binom{n}{i} p^i (1-p)^{n-i} \leq \exp\left(-n(1/2 - p)^2\right).$$

Majority vote.



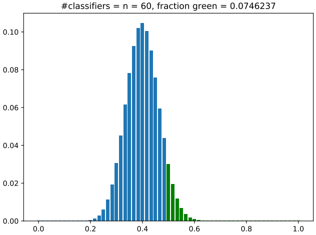
Green region is error of majority vote!
Suppose $y_i \in \{-1, +1\}$.

$$\mathrm{MAJ}(y_1, \ldots, y_n) := \begin{cases} +1 & \text{when } \sum_i y_i \geq 0, \\ -1 & \text{when } \sum_i y_i < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability p):

$$\Pr[\mathrm{Binom}(n, p) \geq n/2] = \sum_{i=n/2}^{n} \binom{n}{i} p^i (1-p)^{n-i} \leq \exp\left(-n(1/2 - p)^2\right).$$

Majority vote.



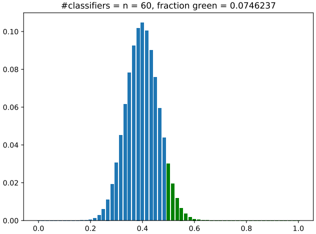
Green region is error of majority vote!
Suppose $y_i \in \{-1, +1\}$.

$$\mathrm{MAJ}(y_1, \ldots, y_n) := \begin{cases} +1 & \text{when } \sum_i y_i \geq 0, \\ -1 & \text{when } \sum_i y_i < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability p):

$$\Pr[\mathrm{Binom}(n, p) \geq n/2] = \sum_{i=n/2}^{n} \binom{n}{i} p^i (1-p)^{n-i} \leq \exp\left(-n(1/2 - p)^2\right).$$

Bottom line.



Green region is error of majority vote!

Error of majority vote classifier goes down **exponentially** in n.

Let's use it in practice!

- Version 1: trying for independent errors.
- Version 2: allowing non-independent errors with adaptive classifiers.

Interlude: decision trees.

Interlude: decision trees.

(Why now? Work well with ensemble methods. . .)

Decision trees.

A **decision tree** is a function $f\colon \mathcal{X} \to \mathcal{Y}$, represented by a binary tree in which:

- Each **tree node** is associated with a splitting rule $g\colon \mathcal{X} \to \{0, 1\}$.
- Each **leaf node** is associated with a label $y \in \mathcal{Y}$.

Decision trees.

A **decision tree** is a function $f \colon \mathcal{X} \to \mathcal{Y}$, represented by a binary tree in which:

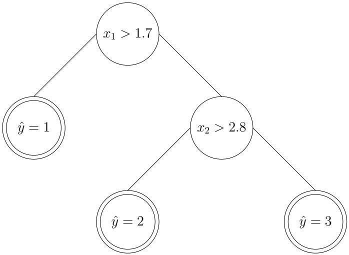
- Each **tree node** is associated with a splitting rule $g \colon \mathcal{X} \to \{0, 1\}$.
- Each **leaf node** is associated with a label $y \in \mathcal{Y}$.

When $\mathcal{X} = \mathbb{R}^d$, typically only consider splitting rules of the form

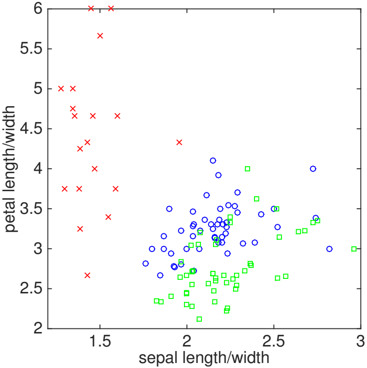
$$g(\boldsymbol{x}) \;=\; \mathbb{1}[x_i > t]$$

for some $i \in [d]$ and $t \in \mathbb{R}$.
Called *axis-aligned* or *coordinate* splits.

(Notation: $[d] := \{1, 2, \ldots, d\}$)



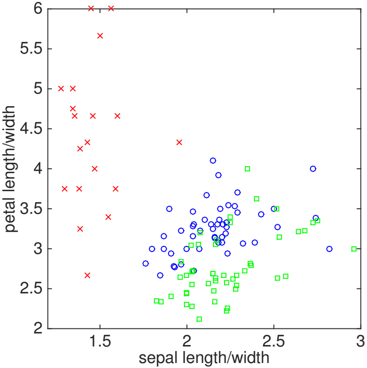
Decision tree example.



Classifying irises by sepal and petal measurements

- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
- x_1 = ratio of sepal length to width
- x_2 = ratio of petal length to width

Decision tree example.

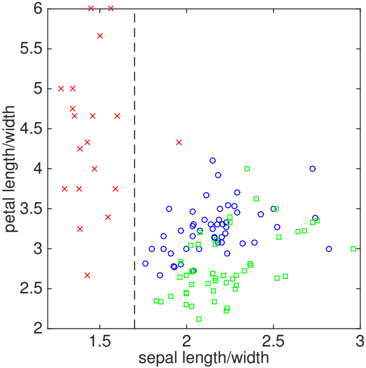


Classifying irises by sepal and petal measurements

- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
- x_1 = ratio of sepal length to width
- x_2 = ratio of petal length to width

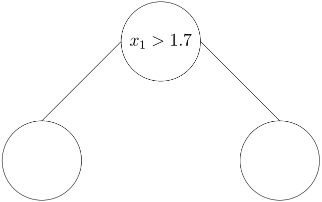
$$\hat{y} = 2$$

Decision tree example.

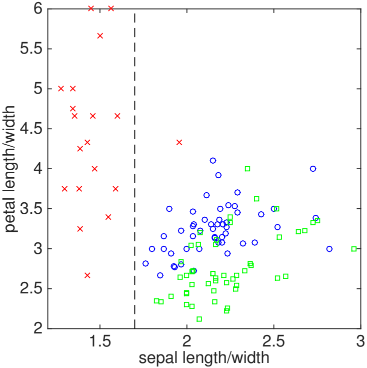


Classifying irises by sepal and petal measurements

- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
- x_1 = ratio of sepal length to width
- x_2 = ratio of petal length to width

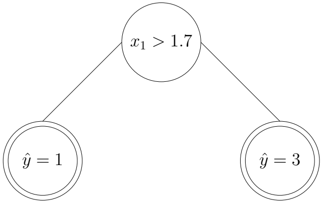


Decision tree example.

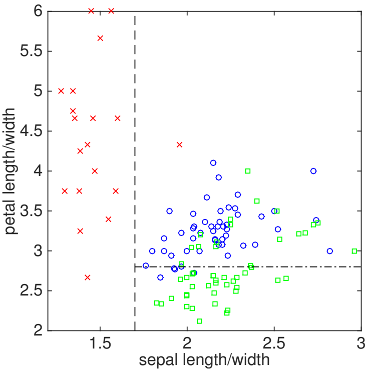


Classifying irises by sepal and petal measurements

- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
- x_1 = ratio of sepal length to width
- x_2 = ratio of petal length to width

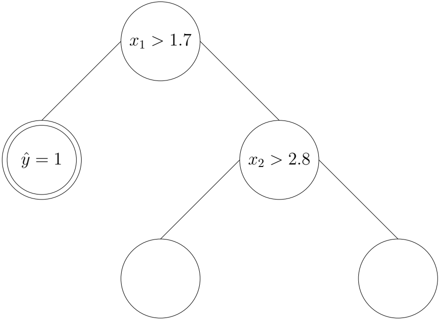


Decision tree example.

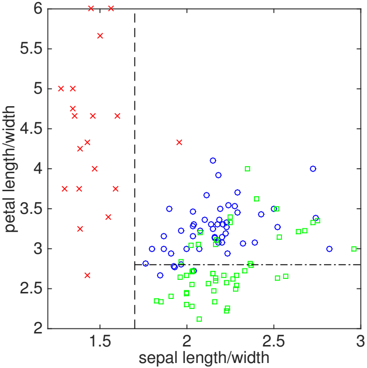


Classifying irises by sepal and petal measurements

- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
- x_1 = ratio of sepal length to width
- x_2 = ratio of petal length to width

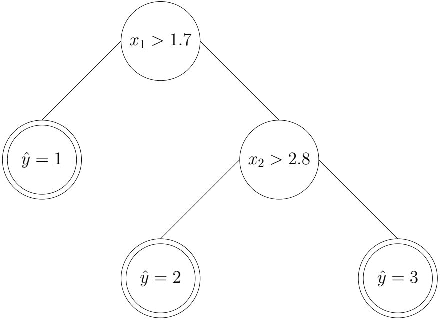


Decision tree example.

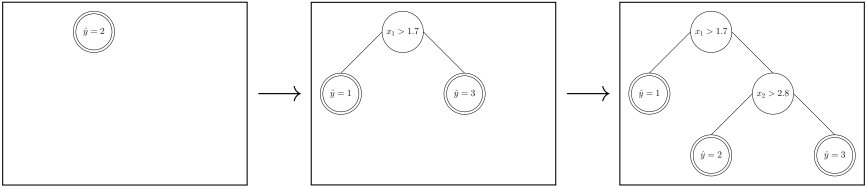


Classifying irises by sepal and petal measurements

- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
- x_1 = ratio of sepal length to width
- x_2 = ratio of petal length to width



Basic decision tree learning algorithm.



Basic "top-down" greedy algorithm.

- Initially, tree is a single leaf node containing all (training) data.
- Loop:
  - Pick the leaf ℓ and rule h that maximally reduces uncertainty.
  - Split data in ℓ using h, and grow tree accordingly.

…until some stopping criterion is satisfied.

[Leaves are labeled with **plurality label** of data reaching them.]

Stopping criterion.

Many alternatives; two common choices are:

Stopping criterion.

Many alternatives; two common choices are:

1. Stop when the **tree reaches a pre-specified size**.

Stopping criterion.

Many alternatives; two common choices are:

1. Stop when the **tree reaches a pre-specified size**.

   Involves additional "tuning parameters" (similar to k in k-NN).

Stopping criterion.

Many alternatives; two common choices are:

1. Stop when the **tree reaches a pre-specified size**.

   Involves additional “tuning parameters” (similar to k in k-NN).

2. Stop when **every leaf is pure.** (More common.)

Stopping criterion.

Many alternatives; two common choices are:

1. Stop when the **tree reaches a pre-specified size**.

   Involves additional "tuning parameters" (similar to k in k-NN).

2. Stop when **every leaf is pure**. (More common.)

   Serious danger of **overfitting** spurious structure due to sampling.

Stopping criterion.

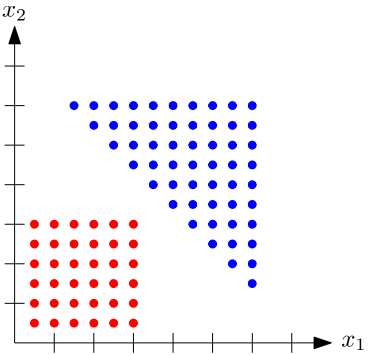
Many alternatives; two common choices are:

1. Stop when the **tree reaches a pre-specified size.**

   Involves additional "tuning parameters" (similar to k in k-NN).

2. Stop when **every leaf is pure.** (More common.)

   Serious danger of **overfitting** spurious structure due to sampling.



Stopping criterion.

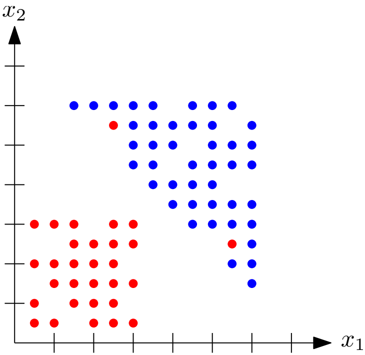
Many alternatives; two common choices are:

1. Stop when the **tree reaches a pre-specified size.**

   Involves additional "tuning parameters" (similar to k in k-NN).

2. Stop when **every leaf is pure.** (More common.)

   Serious danger of **overfitting** spurious structure due to sampling.



Stopping criterion.

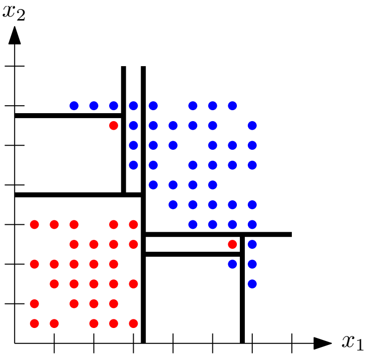
Many alternatives; two common choices are:

1. Stop when the **tree reaches a pre-specified size.**

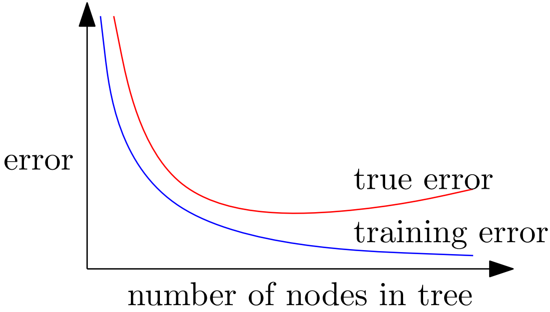
   Involves additional "tuning parameters" (similar to k in k-NN).

2. Stop when **every leaf is pure.** (More common.)

   Serious danger of **overfitting** spurious structure due to sampling.



Overfitting.



- Training error **goes to zero** as number of tree nodes increases.
- True error decreases initially, but eventually **increases (overfitting)**.

Example: Spam filtering.

Data.

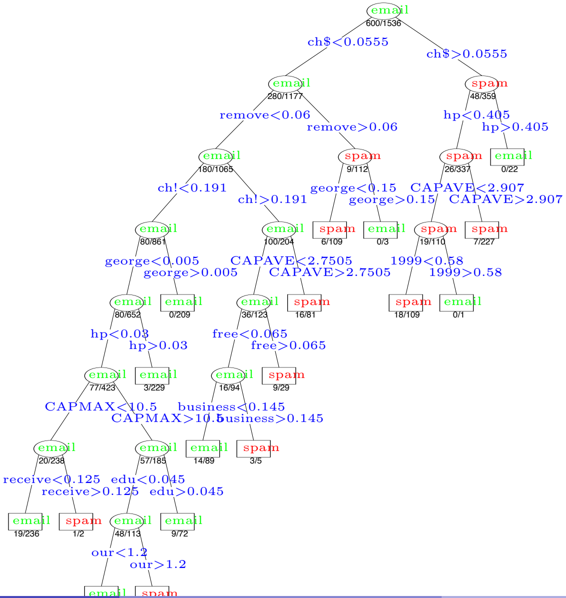
- 4601 e-mail messages, 39.4% are spam.
- $\mathcal{Y} = \{\text{spam}, \text{not spam}\}$
- E-mails represented by 57 features:
  - 48: percentange of e-mail words that is specific word (e.g., “free”, “business”)
  - 6: percentage of e-mail characters that is specific character (e.g., “!”).
  - 3: other features (e.g., average length of ALL-CAPS words).

Results. Using variant of greedy algorithm to grow tree; prune tree using validation set.

Chosen tree has just **17 leaves.** Test error is 9.3%.

	$\hat{y} = \text{not spam}$	$\hat{y} = \text{spam}$
$y = \text{not spam}$	57.3%	4.0%
$y = \text{spam}$	5.3%	33.4%

Example: Spam filtering.



Final remarks.

- Decision trees are very flexible classifiers (like NN).
  - Certain greedy strategies for training decision trees are **consistent**.
  - But also **very prone to overfitting** in most basic form.
  - (NP-hard to find smallest decision tree consistent with data.)

Practical majority vote: bagging and random forests.

Combining decision trees.

Let's majority vote a few decision trees.

Problem: Decision tree method we suggested is deterministic.

Combining decision trees.

Let's majority vote a few decision trees.
Problem: Decision tree method we suggested is deterministic.

(Why decision trees? . . . This is research from the 90s. . .)

Bagging

Bagging = **Bootstrap aggregating** (Leo Breiman, 1994).

Input: training data $\{(x_i, y_i)\}_{i=1}^n$ from $\mathcal{X} \times \{-1, +1\}$.

For $t = 1, 2, \ldots, T$:

1. Randomly pick n examples *with replacement* from training data
   $\longrightarrow \quad \{(x_i^{(t)}, y_i^{(t)})\}_{i=1}^n$ (a *bootstrap* sample).
2. Run learning algorithm on $\{(x_i^{(t)}, y_i^{(t)})\}_{i=1}^n$
   $\longrightarrow \quad$ classifier f_t.

Return a majority vote classifier over $f_1, f_2, \ldots, f_T$.

Aside: sampling with replacement.

Question: if n individuals are picked from a population of size n *u.a.r. with replacement*, what is the probability that a given individual is *not* picked?

Aside: sampling with replacement.

Question: if n individuals are picked from a population of size n *u.a.r. with replacement*, what is the probability that a given individual is *not* picked?

Answer:

$$\left(1 - \frac{1}{n}\right)^n$$

Aside: sampling with replacement.

Question: if n individuals are picked from a population of size n *u.a.r. with replacement*, what is the probability that a given individual is *not* picked?

Answer:

$$\left(1 - \frac{1}{n}\right)^n$$

For large n:

$$\lim_{n \to \infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e} \approx 0.3679\,.$$

Aside: sampling with replacement.

Question: if n individuals are picked from a population of size n *u.a.r. with replacement*, what is the probability that a given individual is *not* picked?

Answer:

$$\left(1 - \frac{1}{n}\right)^n$$

For large n:

$$\lim_{n\to\infty}\left(1 - \frac{1}{n}\right)^n = \frac{1}{e} \approx 0.3679\,.$$

Implications for bagging:

Aside: sampling with replacement.

Question: if n individuals are picked from a population of size n *u.a.r. with replacement*, what is the probability that a given individual is *not* picked?

Answer:

$$\left(1 - \frac{1}{n}\right)^n$$

For large n:

$$\lim_{n\to\infty}\left(1 - \frac{1}{n}\right)^n = \frac{1}{e} \approx 0.3679\,.$$

Implications for bagging:

- Each bootstrap sample contains about 63% of the data set.

Aside: sampling with replacement.

Question: if n individuals are picked from a population of size n *u.a.r. with replacement*, what is the probability that a given individual is *not* picked?

Answer:

$$\left(1 - \frac{1}{n}\right)^n$$

For large n:

$$\lim_{n \to \infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e} \approx 0.3679 .$$

Implications for bagging:

- Each bootstrap sample contains about 63% of the data set.
- Remaining 37% can be used to estimate error rate of classifier trained on the bootstrap sample.

Random Forests.

Random Forests (Leo Breiman, 2001).

Input: training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ from $\mathbb{R}^d \times \{-1, +1\}$.

For $t = 1, 2, \ldots, T$:

1. Randomly pick n examples *with replacement* from training data
$\longrightarrow \quad \{(\boldsymbol{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^n$ (a *bootstrap* sample).

2. Run variant of decision tree learning algorithm on $\{(\boldsymbol{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^n$, where each split is chosen by only considering a random subset of $\sqrt{d}$ features (rather than all d features)
$\longrightarrow \quad$ decision tree classifier f_t.

Return a majority vote classifier over $f_1, f_2, \ldots, f_T$.

Non-independent errors with majority vote: boosting.

Non-indepedent errors.

So far, we combined classifiers with **independent errors.**
(This never happens / is expensive.)

How can we handle dependent errors?

Non-indepedent errors.

So far, we combined classifiers with **independent errors**.
(This never happens / is expensive.)

How can we handle dependent errors?

We'll use an assumption on how we get classifiers.

Reminder:
old setting is we have n classifiers handed to us
and then majority vote over them.

Non-indepedent errors.

So far, we combined classifiers with **independent errors.**
(This never happens / is expensive.)

How can we handle dependent errors?

We'll use an assumption on how we get classifiers.

- We can adaptively choose classifiers.

Reminder:
old setting is we have n classifiers handed to us
and then majority vote over them.

Non-indepedent errors.

So far, we combined classifiers with **independent errors.**
(This never happens / is expensive.)

How can we handle dependent errors?

We'll use an assumption on how we get classifiers.

- We can adaptively choose classifiers.
- We can **reweight the dataset.**

Reminder:
old setting is we have n classifiers handed to us
and then majority vote over them.

Boosting.

We will call this a **weak learning oracle**
with **weak learning rate** $\gamma > 0$.

- We have a black box ("**weak learning oracle (WLO)**") which we feed *reweighted* data set and it gives us back a classifier with error $\leq 1/2 - \gamma$.

Algorithm scheme.

1. Start with uniform distribution over dataset.
2. Ask *weak learning oracle* for a new classisfier.
3. Reweight dataset:
   examples where current ensemble is bad
   will have more weight.
4. Go back to # 2.

AdaBoost (Adaptive Boosting).

input Training data $\{(x_i, y_i)\}_{i=1}^n$ from $\mathcal{X} \times \{-1, +1\}$.

1: **initialize** $D_1(i) := 1/n$ for each $i = 1, 2, \ldots, n$ (a probability distribution).

2: **for** $t = 1, 2, \ldots, T$ **do**

3: Give D_t-weighted examples to WLO; get back $f_t \colon \mathcal{X} \to \{-1, +1\}$.

4: Update weights:

$$\begin{aligned} z_t &:= \sum_{i=1}^{n} D_t(i) \cdot y_i f_t(x_i) \in [-1, +1] \\ \alpha_t &:= \frac{1}{2} \ln \frac{1 + z_t}{1 - z_t} \in \mathbb{R} \quad \text{(weight of } f_t\text{)} \\ D_{t+1}(i) &:= D_t(i) \exp\big(-\alpha_t \cdot y_i f_t(x_i)\big) / Z_t \quad \text{for each } i = 1, 2, \ldots, n, \end{aligned}$$

where $Z_t > 0$ is normalizer that makes D_{t+1} a probability distribution.

5: **end for**

6: **return** Final classifier $\hat{f}(x) := \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t \cdot f_t(x)\right)$.

(Let $\operatorname{sign}(z) := 1$ if $z > 0$ and $\operatorname{sign}(z) := -1$ if $z \leq 0$.)

Interpretation.

Interpreting z_t. Suppose $(X, Y) \sim D_t$. If

$$P(f(X) = Y) \; = \; \frac{1}{2} + \gamma_t \, ,$$

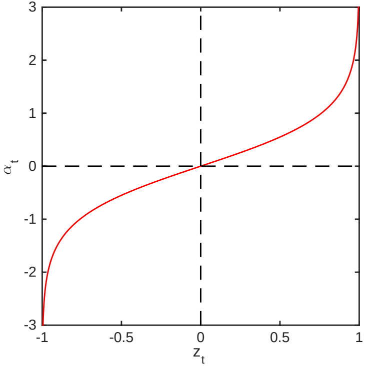
then

$$z_t \; = \; \sum_{i=1}^{n} D_t(i) \cdot y_i f(x_i) \; = \; 2\gamma_t \; \in \; [-1, +1] \, .$$

- $z_t = 0 \iff$ random guessing w.r.t. D_t.
- $z_t > 0 \iff$ better than random guessing w.r.t. D_t.
- $z_t < 0 \iff$ better off using the opposite of f's predictions.

Interpretation.

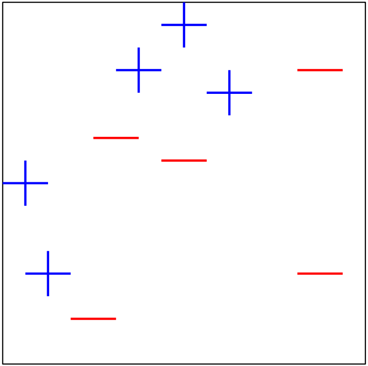
Classifier weights $\alpha_t = \frac{1}{2} \ln \frac{1+z_t}{1-z_t}$



Example weights $D_{t+1}(i)$

$$D_{t+1}(i) \propto D_t(i) \cdot \exp(-\alpha_t \cdot y_i f_t(x_i)) \,.$$

Example: AdaBoost with decision stumps.



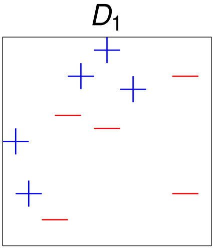
Weak learning algorithm : ERM with $\mathcal{F} =$ "decision stumps" on $\mathbb{R}^2$

(i.e., axis-aligned threshold functions $\boldsymbol{x} \mapsto \text{sign}(\boldsymbol{x}_i - t)$).

Straightforward to handle importance weights in ERM.

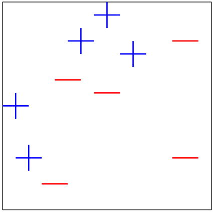
(Example from Figures 1.1 and 1.2 of Schapire & Freund text.)

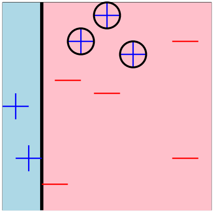
Example: execution of AdaBoost.



Example: execution of AdaBoost.

D_1

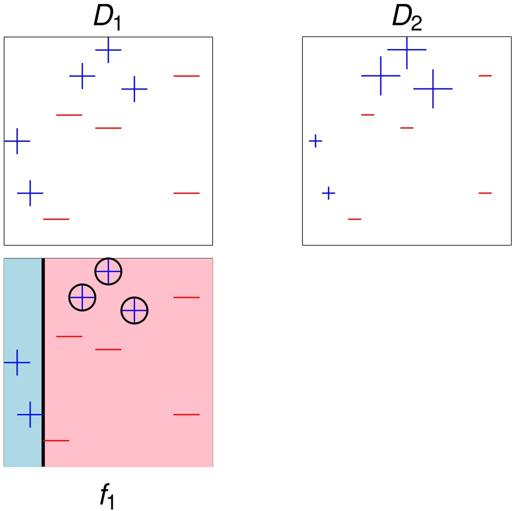




f_1

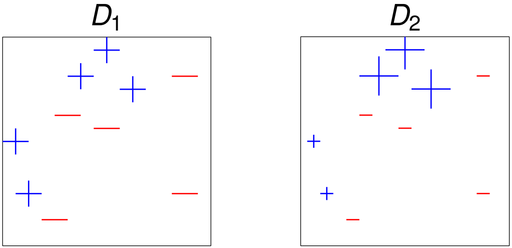
$z_1 = 0.40$, $\alpha_1 = 0.42$

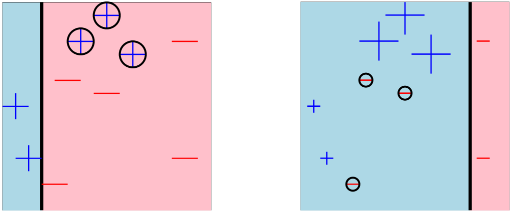
Example: execution of AdaBoost.



$z_1 = 0.40$, $\alpha_1 = 0.42$

Example: execution of AdaBoost.

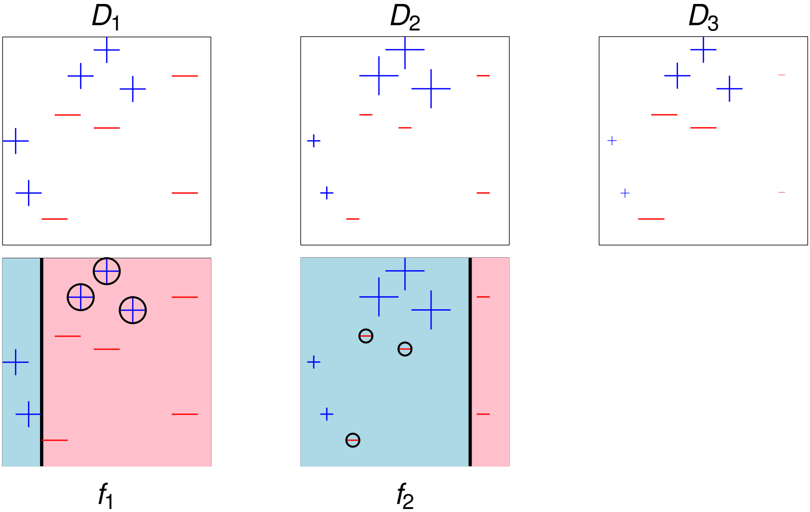




f_1 f_2

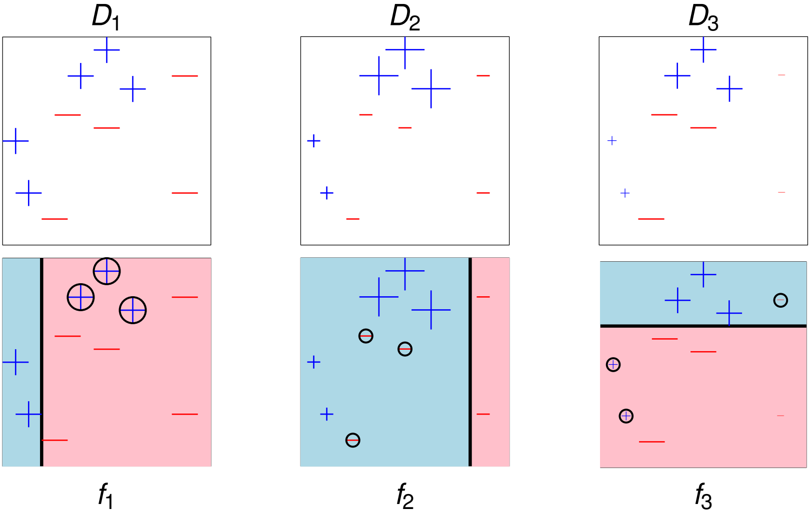
$z_1 = 0.40$, $\alpha_1 = 0.42$ $z_2 = 0.58$, $\alpha_2 = 0.65$

Example: execution of AdaBoost.



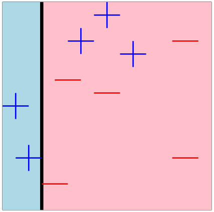
$z_1 = 0.40$, $\alpha_1 = 0.42$ $z_2 = 0.58$, $\alpha_2 = 0.65$

Example: execution of AdaBoost.



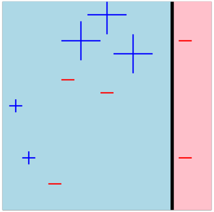
$z_1 = 0.40,\ \alpha_1 = 0.42$ $\quad z_2 = 0.58,\ \alpha_2 = 0.65$ $\quad z_3 = 0.72,\ \alpha_3 = 0.92$

Example: final classifier from AdaBoost.



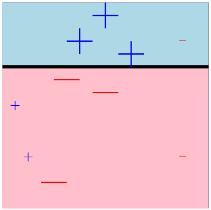
f_1

$z_1 = 0.40$, $\alpha_1 = 0.42$



f_2

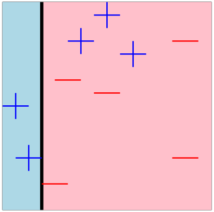
$z_2 = 0.58$, $\alpha_2 = 0.65$



f_3

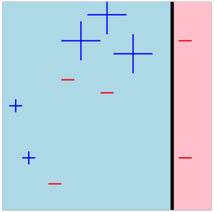
$z_3 = 0.72$, $\alpha_3 = 0.92$

Example: final classifier from AdaBoost.



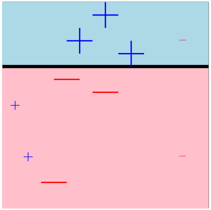
f_1

$z_1 = 0.40, \alpha_1 = 0.42$



f_2

$z_2 = 0.58, \alpha_2 = 0.65$



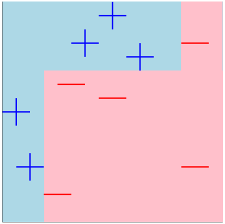
f_3

$z_3 = 0.72, \alpha_3 = 0.92$

Final classifier

$\hat{f}(x) = \text{sign}(0.42 f_1(x) + 0.65 f_2(x) + 0.92 f_3(x))$

(Zero training error rate!)



Training error rate of final classifier.

Recall $\gamma_t := P(f_t(X) = Y) - 1/2 = z_t/2$ when $(X, Y) \sim D_t$.

Training error rate of final classifier from AdaBoost:

$$(\hat{f}, \{(x_i, y_i)\}_{i=1}^n) \;\leq\; \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right).$$

Training error rate of final classifier.

Recall $\gamma_t := P(f_t(X) = Y) - 1/2 = z_t/2$ when $(X, Y) \sim D_t$.

Training error rate of final classifier from AdaBoost:

$$(\hat{f}, \{(x_i, y_i)\}_{i=1}^{n}) \;\leq\; \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right).$$

If average $\bar{\gamma}^2 := \frac{1}{T}\sum_{t=1}^{T}\gamma_t^2 > 0$, then training error rate is $\leq \exp\left(-2\bar{\gamma}^2 T\right)$.

Training error rate of final classifier.

Recall $\gamma_t := P(f_t(X) = Y) - 1/2 = z_t/2$ when $(X, Y) \sim D_t$.

Training error rate of final classifier from AdaBoost:

$$(\hat{f}, \{(x_i, y_i)\}_{i=1}^{n}) \;\le\; \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right).$$

If average $\bar{\gamma}^2 := \frac{1}{T}\sum_{t=1}^{T}\gamma_t^2 > 0$, then training error rate is $\le \exp\left(-2\bar{\gamma}^2 T\right)$.

"AdaBoost" = "<u>Ada</u>ptive <u>Boost</u>ing"

Some γ_t could be small, even negative—only care about overall average $\bar{\gamma}^2$.

Training error rate of final classifier.

Recall $\gamma_t := P(f_t(X) = Y) - 1/2 = z_t/2$ when $(X, Y) \sim D_t$.

Training error rate of final classifier from AdaBoost:

$$(\hat{f}, \{(x_i, y_i)\}_{i=1}^{n}) \;\le\; \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right).$$

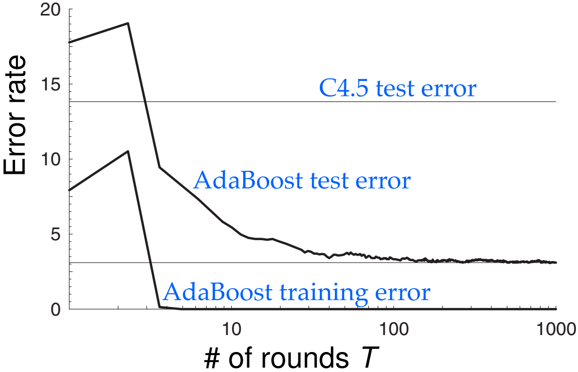
If average $\bar{\gamma}^2 := \frac{1}{T}\sum_{t=1}^{T}\gamma_t^2 > 0$, then training error rate is $\le \exp\left(-2\bar{\gamma}^2 T\right)$.

"AdaBoost" = "<u>Ada</u>ptive <u>Boost</u>ing"
Some γ_t could be small, even negative—only care about overall average $\bar{\gamma}^2$.

What about true error rate?

A typical run of boosting.

AdaBoost+C4.5 on “letters” dataset.



(# nodes across all decision trees in $\hat{f}$ is $>2 \times 10^6$)

Training error rate is zero after just five rounds,
but test error rate continues to decrease, even up to 1000 rounds!

(Figure 1.7 from Schapire & Freund text)

Boosting the margin.

Final classifier from AdaBoost:

$$\hat{f}(x) \;=\; \operatorname{sign} \underbrace{\left(\frac{\sum_{t=1}^{T} \alpha_t f_t(x)}{\sum_{t=1}^{T} |\alpha_t|} \right)}_{g(x) \in [-1,+1]} .$$

Call $y \cdot g(x) \in [-1, +1]$ the **margin** achieved on example (x, y).

Boosting the margin.

Final classifier from AdaBoost:

$$\hat{f}(x) \;=\; \operatorname{sign}\underbrace{\left(\frac{\sum_{t=1}^{T} \alpha_t f_t(x)}{\sum_{t=1}^{T} |\alpha_t|}\right)}_{g(x) \in [-1,+1]}.$$

Call $y \cdot g(x) \in [-1, +1]$ the **margin** achieved on example (x, y).

New theory [Schapire, Freund, Bartlett, and Lee, 1998]:

- **Larger margins** $\Rightarrow$ **better resistance to overfitting**, independent of T.
- AdaBoost tends to increase margins on training examples.

(Similar but not the same as SVM margins.)

Boosting the margin.

Final classifier from AdaBoost:

$$\hat{f}(x) \;=\; \operatorname{sign}\underbrace{\left(\frac{\sum_{t=1}^{T}\alpha_t f_t(x)}{\sum_{t=1}^{T}|\alpha_t|}\right)}_{g(x)\in[-1,+1]}.$$

Call $y \cdot g(x) \in [-1, +1]$ the **margin** achieved on example (x, y).

New theory [Schapire, Freund, Bartlett, and Lee, 1998]:

- **Larger margins** $\Rightarrow$ **better resistance to overfitting**, independent of T.
- AdaBoost tends to increase margins on training examples.

(Similar but not the same as SVM margins.)

On "letters" dataset:

	$T = 5$	$T = 100$	$T = 1000$
training error rate	0.0%	0.0%	0.0%
test error rate	8.4%	3.3%	3.1%
% margins $\leq$0.5	7.7%	0.0%	0.0%

Final remarks.

Miscellaneous remarks.

- Popular boosting + decision tree framework: `xgboost`.
- Dropout in neural nets sometimes explained as ensemble / averaging.
- Other forms of ensemble/aggregation used with neural nets as well.

Summary.

- **Majority vote:**
  an effective way to combine "uncorrelated" classifiers.
- **Decision trees:**
  a flexible space-partitioning classifier.
- **Bagging / Random forests:**
  ensemble methods in the general case and for decision trees.
- **Boosting:**
  ensemble method for non-independent errors.