Applied Machine Learning

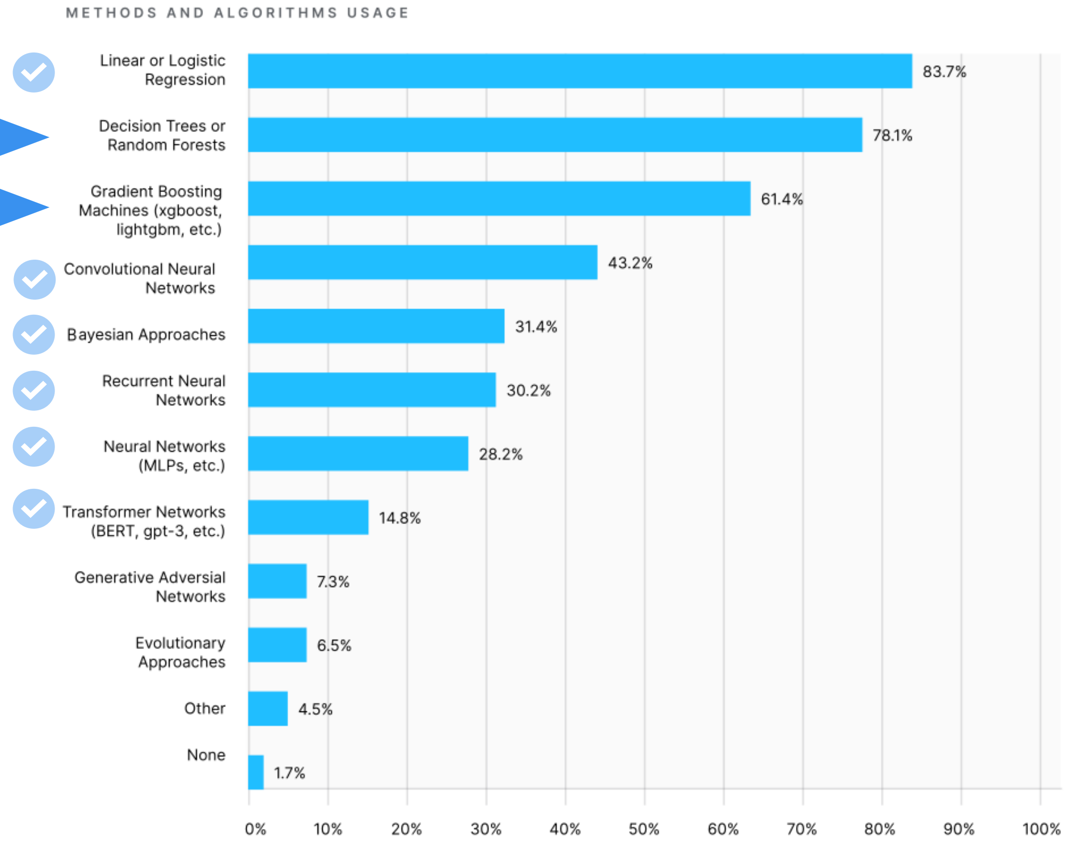
Bagging & Boosting

Reihaneh Rabbany

McGill
School of Computer Science

Motivation

Today's topic is highly practical



from 2020 Kaggle's survey on the state of Machine Learning and Data Science,
you can read the full version here

Learning objectives

bootstrap for uncertainty estimation

bagging (bootstrap aggregation) for variance reduction

- random forests

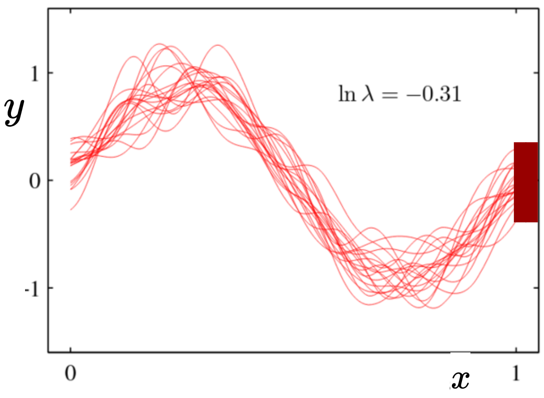
boosting

- AdaBoost
- gradient boosting
- relationship to L1 regularization

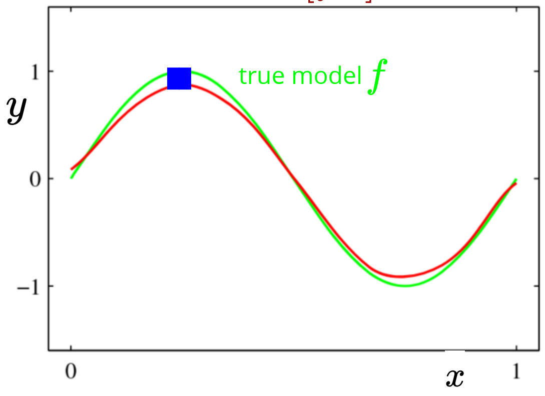
Reminder: bias vs. variance

variance is the average difference (in squared L2 norm) between these curves and their average

models for different datasets $\hat{f}_{\mathcal{D}}$



the average model $\mathbb{E}[\hat{f}_{\mathcal{D}}]$



bias is the difference (in L2 norm) between two curves

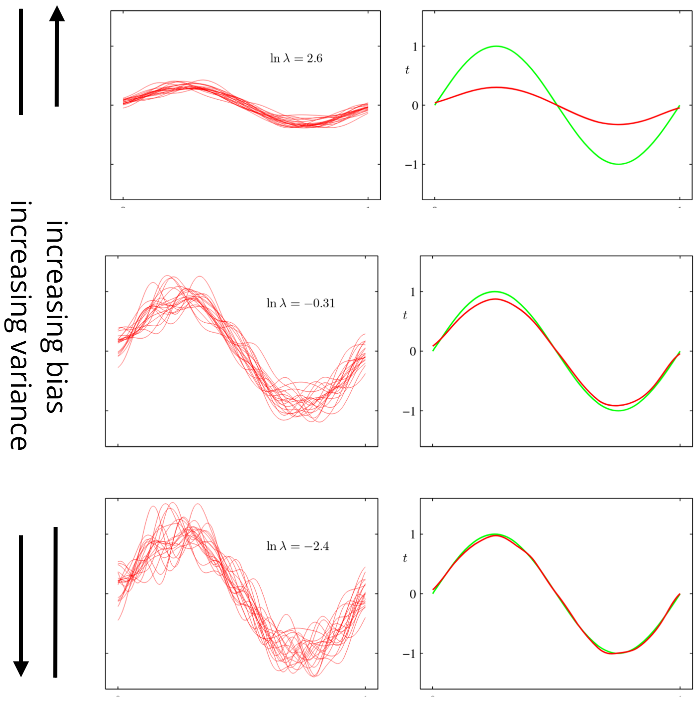
🟥 **variance:** $\mathbb{E}[(\hat{f}_{\mathcal{D}}(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2]$

how change of dataset affects the prediction

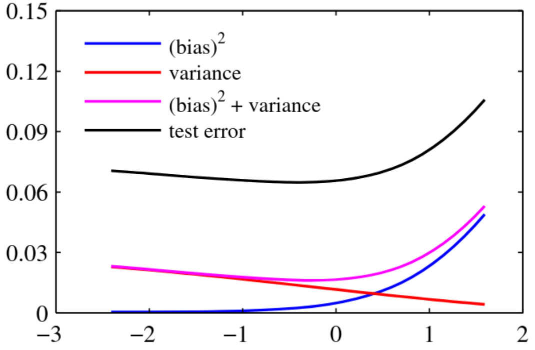
🟦 **bias:** $\mathbb{E}[(f(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2]$

how average over all datasets differs from the regression function

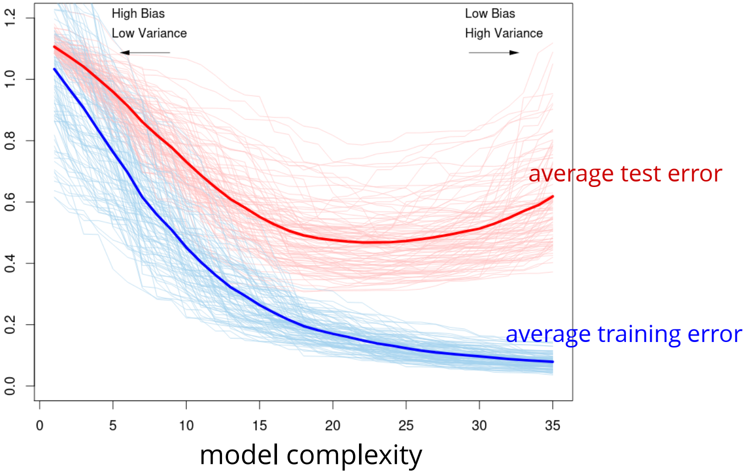
Reminder: bias vs. variance



the expected loss (test error) increases with both bias and variance



simple or weak models seem to do good on average and have lower variance



Reducing Bias & variance

we saw a trade-off between bias (simplicity) and variance (complexity)

reduce the variance of a model w/o increasing its bias?

Bagging

average multiple models trained on subsets of the data

reduce the bias of a model w/o increasing its variance?

Boosting

reduce the bias of (simple models) by adding them in steps

Bagging (Bootstrap aggregation)

given the dataset $\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$

subsample **with replacement** B datasets of size N

(non-parametric) **bootstrapping**

$$\mathcal{D}_b = \{(x^{(n,b)}, y^{(n,b)})\}_{n=1}^{N} \ , \ b = 1, \ldots, B$$

train a model $\hat{f}_b$ on each of these bootstrap datasets (called *bootstrap samples*)

aggregate the predictions of these models (Bootstrap aggregation)

$$\hat{f}(x) = \frac{1}{B} \sum_b \hat{f}_b(x)$$

bootstrapping can also use to produce a **measure of uncertainty** in predictions

Bootstrap for uncertainty estimation

a simple approach to estimate the uncertainty in prediction

non-parametric bootstrap

given the dataset $\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$

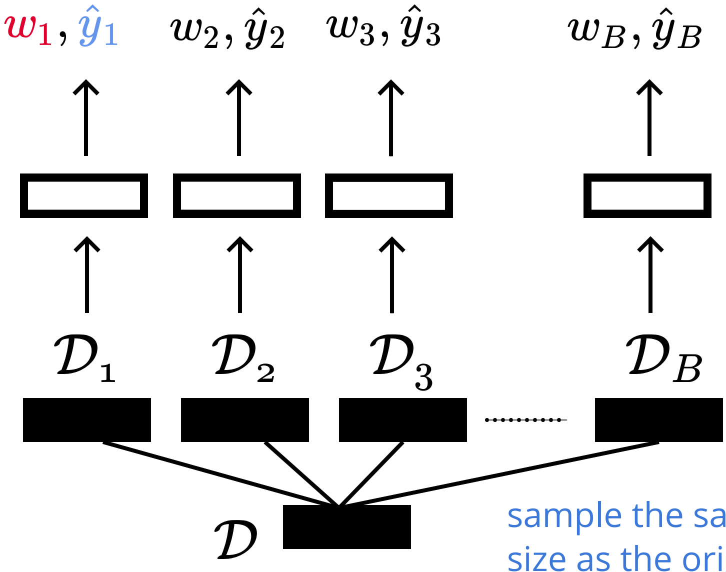
subsample **with replacement** B datasets of size N

$$\mathcal{D}_b = \{(x^{(n,b)}, y^{(n,b)})\}_{n=1}^{N}, b = 1, \ldots, B$$

train a model on each of these bootstrap datasets (called *bootstrap samples*)

measure of uncertainty from these models

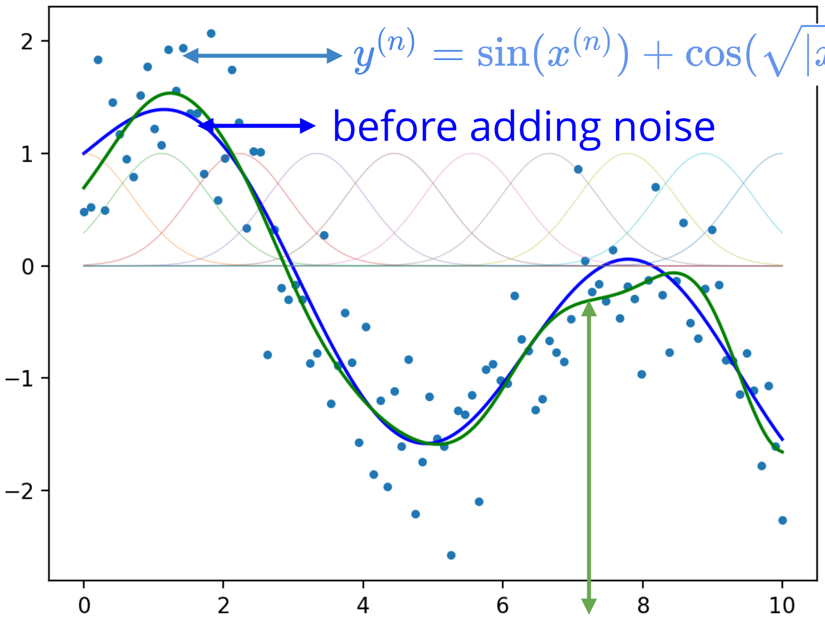
- for model parameters
- for predictions



sample the same size as the original training set

Bootstrap for uncertainty estimation

example **recall** our running example with nonlinear Gaussian bases (N=100 training data points)



$$y^{(n)} = \sin(x^{(n)}) + \cos(\sqrt{|x^{(n)}|}) + \underbrace{\epsilon}_{\text{noise}}$$

before adding noise

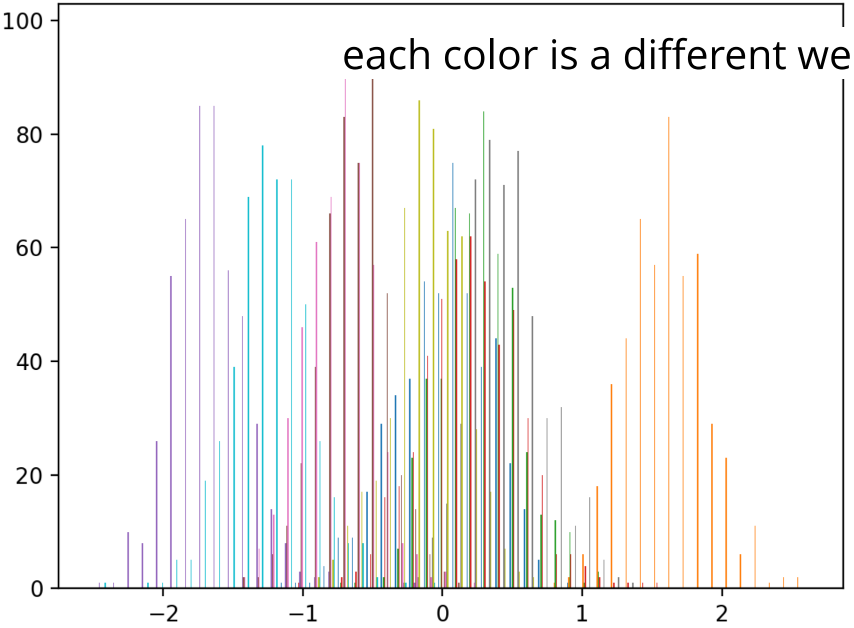
our fit to all datapoints using 10 Gaussian bases

Bootstrap for uncertainty estimation

example **recall** our running example with nonlinear Gaussian bases (N=100 training data points)

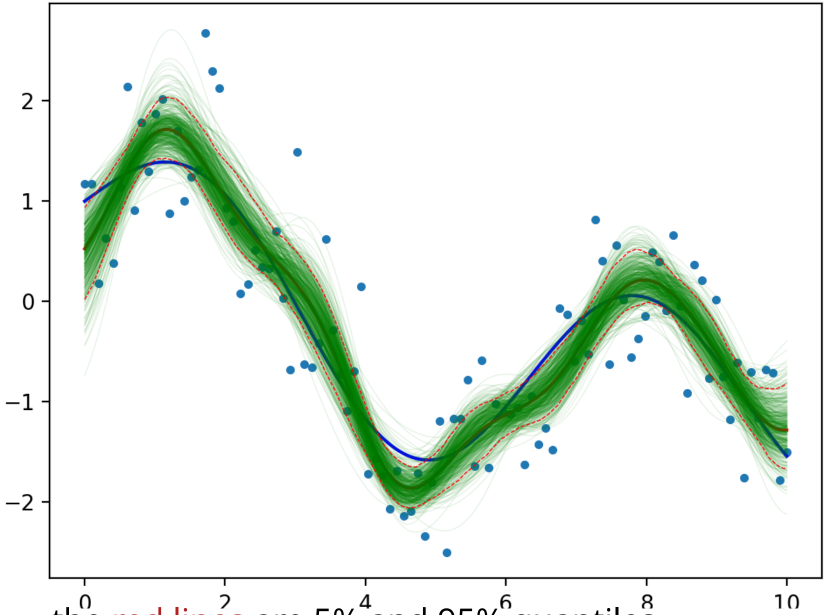
using B=500 bootstrap samples

gives a measure of uncertainty of the parameters



each color is a different weight w_d

also gives a measure of **uncertainty of the predictions**



the red lines are 5% and 95% quantiles
(for each point we can get these across bootstrap model predictions)

Bagging (Bootstrap aggregation)

why using average predictions reduces variance?

variance of the sum of random variables

$$\begin{aligned}\mathrm{Var}(z_1+z_2) &= \mathbb{E}[(z_1+z_2)^2] - \mathbb{E}[z_1+z_2]^2 \\ &= \mathbb{E}[z_1^2+z_2^2+2z_1z_2] - (\mathbb{E}[z_1]+\mathbb{E}[z_2])^2 \\ &= \mathbb{E}[z_1^2] + \mathbb{E}[z_2^2] + \mathbb{E}[2z_1z_2] - \mathbb{E}[z_1]^2 - \mathbb{E}[z_2]^2 - 2\mathbb{E}[z_1]\mathbb{E}[z_2] \\ &= \mathrm{Var}(z_1) + \mathrm{Var}(z_2) + 2\mathrm{Cov}(z_1,z_2)\end{aligned}$$

for uncorrelated variables this term is zero

variance of the **sum** of $z_1 \ldots z_B$ **uncorrelated** random variables

$$\mathrm{Var}(\sum_b z_b) = \sum_b \mathrm{Var}(z_b)$$

variance of the **average** of $z_1 \ldots z_B$ **uncorrelated** random variables, all with variance of σ^2

$$\mathrm{Var}(\frac{1}{B}\sum_b z_b) = \frac{1}{B^2}\mathrm{Var}(\sum_b z_b) = \frac{1}{B^2}B\sigma^2 = \frac{1}{B}\sigma^2$$

Bagging (Bootstrap aggregation)

averaging uncorrelated variables reduces the variance of our model by a factor of B (number of bootstraps)

so the Bagging reduces variance

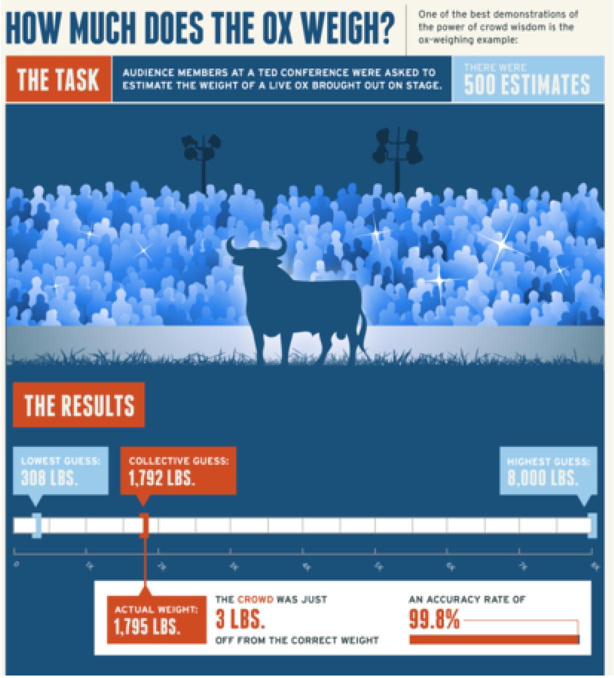
(in reality, predictions are not uncorrelated)

for regression

$$\hat{f}(x) = \frac{1}{B}\sum_b \hat{f}_b(x)$$

prediction using bootstrap sample b

Example


HOW MUCH DOES THE OX WEIGH?
THE TASK
THERE WERE 500 ESTIMATES
THE RESULTS
LOWEST GUESS: 308 LBS.
COLLECTIVE GUESS: 1,792 LBS.
HIGHEST GUESS: 8,000 LBS.
ACTUAL WEIGHT: 1,795 LBS.
THE CROWD WAS JUST 3 LBS. OFF FROM THE CORRECT WEIGHT
AN ACCURACY RATE OF 99.8%

One of the best demonstrations of the power of crowd wisdom is the ox-weighing example:

AUDIENCE MEMBERS AT A TED CONFERENCE WERE ASKED TO ESTIMATE THE WEIGHT OF A LIVE OX BROUGHT OUT ON STAGE.

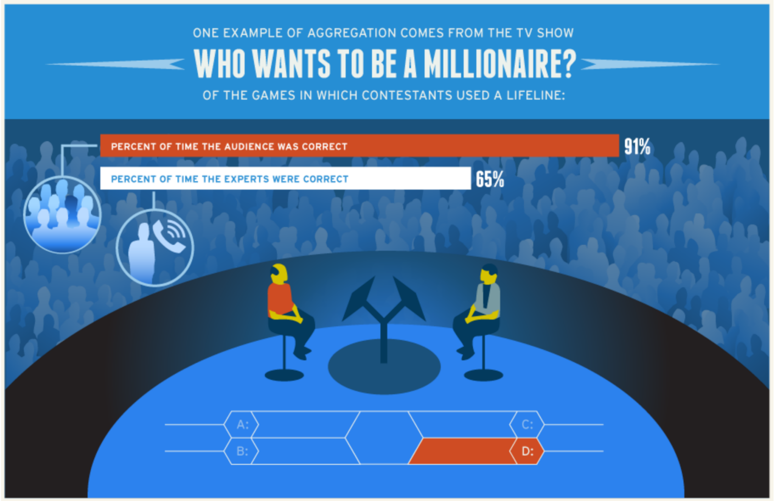
infographics from: domo.com

Bagging (Bootstrap aggregation)

for classification we cannot use mean of classifiers, use voting (i.e., mode)

$z_1, \ldots, z_B \in \{0,1\}$ are IID Bernoulli random variables with mean $\mu = .5 + \epsilon$ $(\epsilon > 0)$, then $P(\frac{1}{B}\sum_b z_b > .5) \to 1$ as B grows

i.e., even if individual predictions are very noisy, average prediction can be accurate



infographics from: domo.com

e.g. with 10K classifiers that are each only slightly better than chance (0.51percent accurate), we get an overall accuracy of 0.97

wisdom of crowds

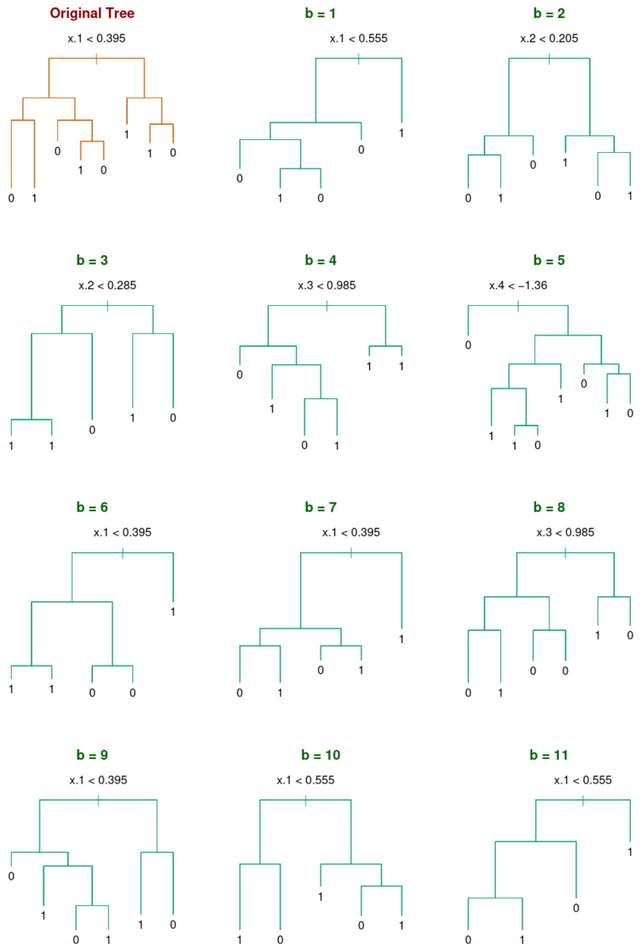
bagging produces a better classifier!

crowds are wise when

- individuals are better than random
- votes are uncorrelated

Bagging decision trees

example

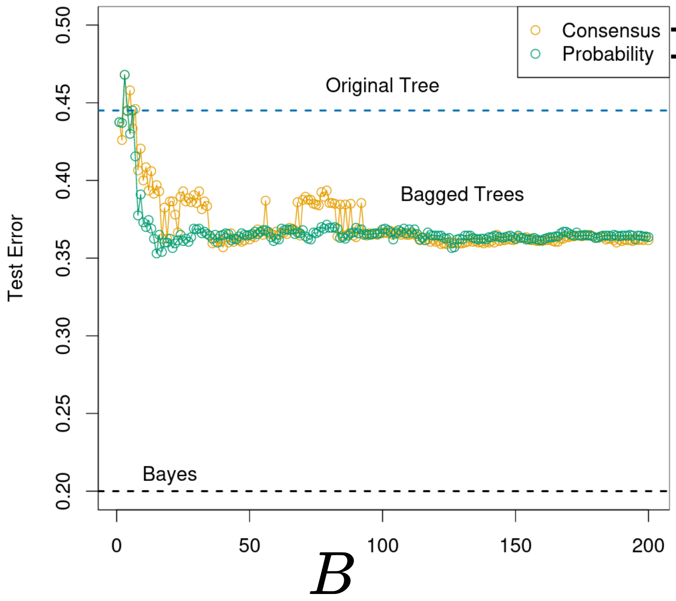


setup

- *synthetic dataset*
- *5 correlated features*
- *1st feature is a noisy predictor of the label*

Bootstrap samples create different decision trees (due to high variance of decision trees)

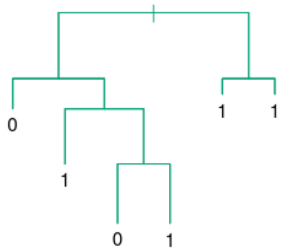
compared to decision trees, no longer **interpretable**!



voting for the most probably class

averaging probabilities

Random forests



further reduce the correlation between decision trees

feature sub-sampling

only a random subset of features are available for split at each step

further reduce the dependence between decision trees

magic number? $\sqrt{D}$

this is a hyper-parameter, can be optimized using CV

Out Of Bag (OOB) samples:

- the instances not included in a bootsrap dataset can be used for validation
- simultaneous validation of decision trees in a forest
- no need to set aside data for **cross validation**

Spam detection

example

Dataset

N=4601 emails

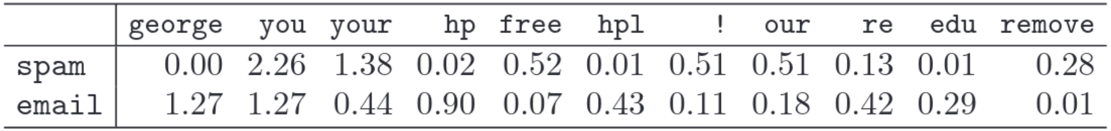
binary classification task: *spam - not spam*

D=57 features:

- **48** words: percentage of words in the email that match these words
  - *e.g.,* business,address,internet, free, George (customized per user)
- **6** characters: again percentage of characters that match these
  - *ch; , ch(,ch[,ch! ,ch$, ch#*
- average, max, sum of length of uninterrupted sequences of capital letters:
  - CAPAVE, CAPMAX, CAPTOT

an example of
feature engineering

average value of these features in the spam and non-spam emails

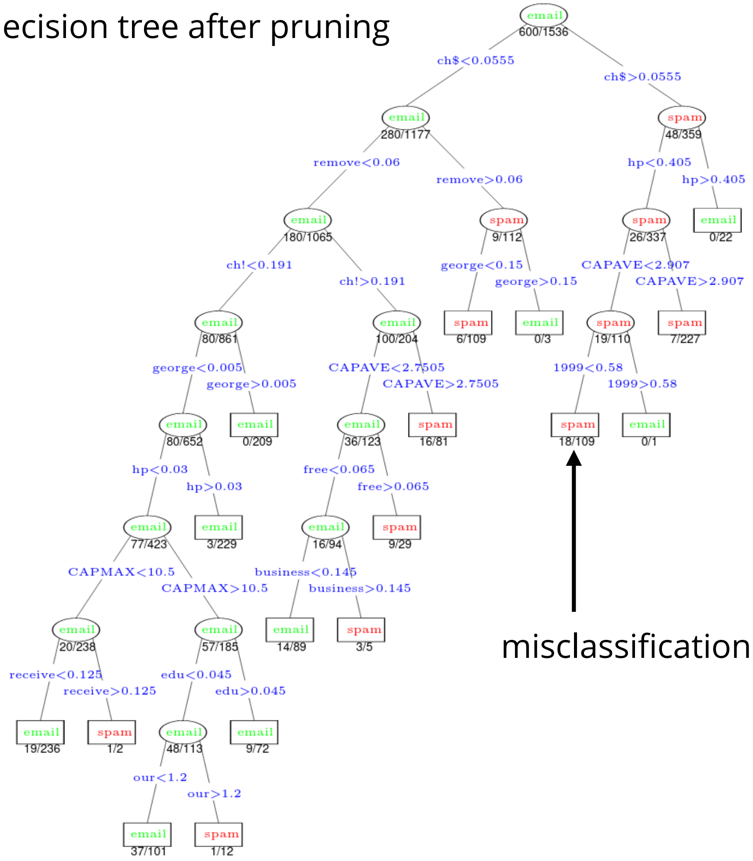


	george	you	your	hp	free	hpl	!	our	re	edu	remove
spam	0.00	2.26	1.38	0.02	0.52	0.01	0.51	0.51	0.13	0.01	0.28
email	1.27	1.27	0.44	0.90	0.07	0.43	0.11	0.18	0.42	0.29	0.01

Spam detection

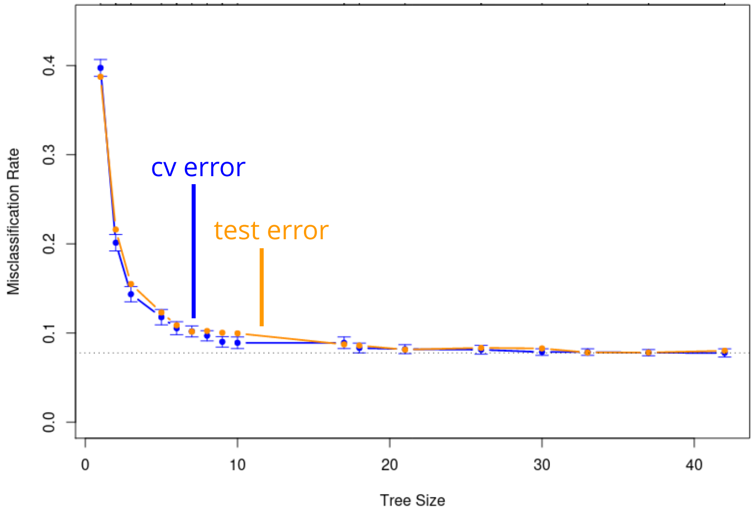
example

decision tree after pruning



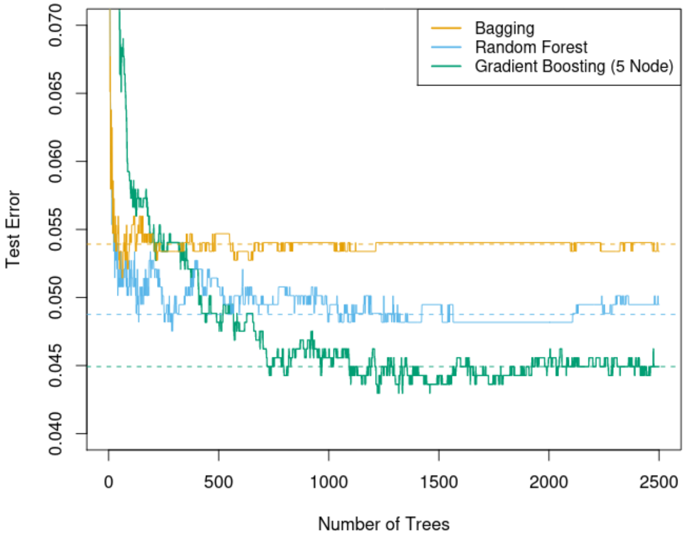
misclassification rate on test data

number of leaves (17) in optimal pruning decided based on cross-validation error

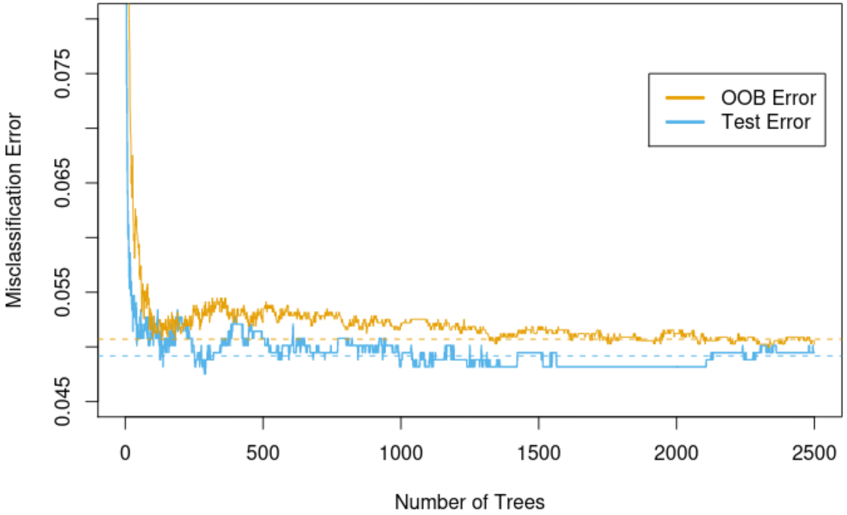


Spam detection example

Bagging and Random Forests do much better than a single decision tree!



Out Of Bag (OOB) error can be used for parameter tuning *(e.g., size of the forest)*



Summary so far…

Ensemble learning

- Bootstrap is a powerful technique to get uncertainty estimates
- Bootstrap aggregation (Bagging) can reduce the variance of unstable models
- Random forests:
  - Bagging + further de-corelation of features at each split
  - OOB validation instead of CV
  - no more interpretability of decision trees
  - perform well in practice
  - can fail if only few relevant features exist (due to feature-sampling)

train independently

Next:

Boosting

reduce the bias of a model w/o increasing its variance?

reduce the bias of (simple models) by adding them in steps

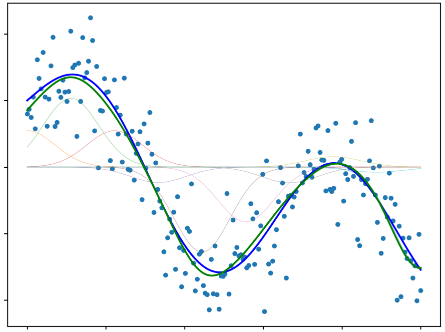
train sequentially

Adaptive bases

fixed set of bases in $f(x) = \sum_d w_d \phi_d(x)$

several methods can be classified as *learning these bases adaptively*

$$f(x) = \sum_d w_d \phi_d(x; v_d)$$



Gaussian bases example

- decision trees
- generalized additive models
- neural networks
- boosting

in boosting each basis is a classifier or regression function (**weak learner, or base learner**)
create a *strong learner* by sequentially combining *weak learners*

Forward stagewise additive modelling

model $f(x) = \sum_{t=1}^{T} w^{\{t\}} \phi(x; v^{\{t\}})$ a simple model, such as decision stump (decision tree with one node)

cost $J(\{w^{\{t\}}, v^{\{t\}}\}_t) = \sum_{n=1}^{N} L(y^{(n)}, f(x^{(n)}))$

e.g. L2 loss or hinge loss

optimizing this cost is difficult given the form of f

optimization idea add one weak-learner in each stage t, to reduce the error of previous stage

1. find the best weak learner

$$v^{\{t\}}, w^{\{t\}} = \arg\min_{v,w} \sum_{n=1}^{N} L(y^{(n)}, f^{\{t-1\}}(x^{(n)}) + w\phi(x^{(n)}; v))$$

2. add it to the current model

$$f^{\{t\}}(x) = f^{\{t-1\}}(x^{(n)}) + w^{\{t\}}\phi(x^{(n)}; v^{\{t\}})$$

optional

L_2 loss & linear modelling

model consider **weak learners** that are individual features $\phi^{\{t\}}(x) = w^{\{t\}} x_{d\{t\}}$

cost using L2 loss for regression $\frac{1}{2}(y - f(x))^2$

residual $r^{(n)}$

at stage t $\arg\min_{d,w_d} \frac{1}{2} \sum_{n=1}^{N} \left(y^{(n)} - (f^{\{t-1\}}(x^{(n)}) + w_d x_d^{(n)}) \right)^2$

optimization optimal weight for each d is $w_d = \frac{\sum_n x_d^{(n)} r_d^{(n)}}{\sum_n {x_d^{(n)}}^2}$

pick the feature that most significantly reduces the residual

the model at time-step t: $f^{\{t\}}(x) = \sum_t \alpha w_{d\{t\}}^{\{t\}} x_{d\{t\}}$

using a small α helps with test error

is this related to L1-regularized linear regression?

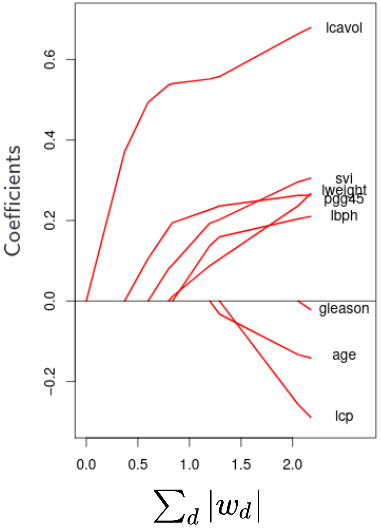
optional

L_2 loss & linear modelling

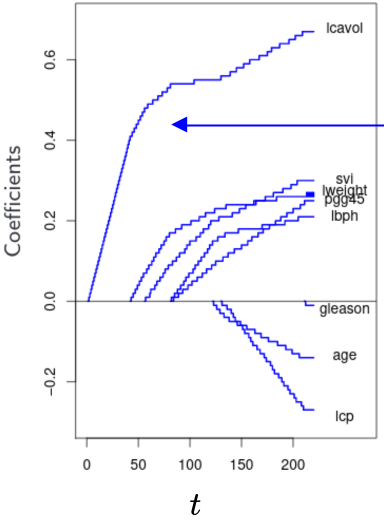
example

using small learning rate $\alpha = .01$ L2 Boosting has a similar regularization path to lasso

lasso



boosting



at each time-step only one feature $d^{\{t\}}$ is updated / added

$w_d^{\{t\}}$

we can view boosting as doing feature (base learner) selection in exponentially large spaces *(e.g., all trees of size K)*

the number of steps **t** plays a similar role to (the inverse of) regularization hyper-parameter

optional

Exponential loss & AdaBoost

loss functions for **binary classification** $y \in \{-1, +1\}$

predicted label is $\hat{y} = \text{sign}(f(x))$

misclassification loss $L(y, f(x)) = \mathbb{I}(yf(x) > 0)$

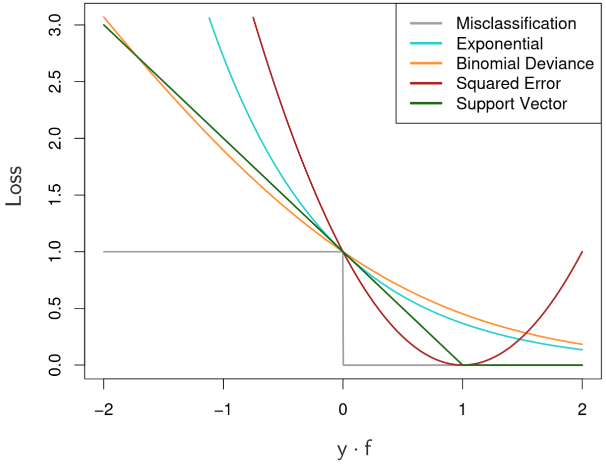
(0-1 loss)

log-loss $L(y, f(x)) = \log\left(1 + e^{-yf(x)}\right)$

(aka cross entropy loss or binomial deviance)

Hinge loss $L(y, f(x)) = \max(0, 1 - yf(x))$

support vector loss



yet another loss function is exponential loss $L(y, f(x)) = e^{-yf(x)}$

note that the loss grows faster than the other surrogate losses (more sensitive to outliers)

useful property when working with additive models:

$$L(y, f^{\{t-1\}}(x) + w^{\{t\}}\phi(x, v^{\{t\}})) = L(y, f^{\{t-1\}}(x)) \cdot L(y, w^{\{t\}}\phi(x, v^{\{t\}}))$$

treat this as a weight **q** for an instance

instances that are not properly classified before receive a higher weight

optional

Exponential loss & AdaBoost

cost using exponential loss

$$J(\{w^{\{t\}}, v^{\{t\}}\}_t) = \sum_{n=1}^{N} L(y^{(n)}, f^{\{t-1\}}(x^{(n)}) + w^{\{t\}}\phi(x^{(n)}, v^{\{t\}})) = \sum_n q^{(n)} L(y^{(n)}, w^{\{t\}}\phi(x^{(n)}, v^{\{t\}}))$$

loss for this instance at previous stage $L(y^{(n)}, f^{\{t-1\}}(x^{(n)}))$

discrete AdaBoost: assume this is a simple classifier, so its output is +/- 1

optimization objective is to find the weak learner minimizing the cost above

$$J(\{w^{\{t\}}, v^{\{t\}}\}_t) = \sum_n q^{(n)} e^{-y^{(n)} w^{\{t\}} \phi(x^{(n)}, v^{\{t\}})}$$

$$= e^{w^{\{t\}}} \sum_n q^{(n)} \mathbb{I}(y^{(n)} \neq \phi(x^{(n)}, v^{\{t\}})) + e^{-w^{\{t\}}} \sum_n q^{(n)} \mathbb{I}(y^{(n)} = \phi(x^{(n)}, v^{\{t\}}))$$

$$= e^{-w^{\{t\}}} \sum_n q^{(n)} + (e^{w^{\{t\}}} - e^{-w^{\{t\}}}) \sum_n q^{(n)} \mathbb{I}(y^{(n)} \neq \phi(x^{(n)}, v^{\{t\}}))$$

does not depend on the weak learner

assuming $w^{\{t\}} \geq 0$ the weak learner should minimize this cost
this is classification with weighted intances

optional

Exponential loss & AdaBoost

cost $$J(\{w^{\{t\}}, v^{\{t\}}\}_t) = \sum_n q^{(n)} L(y^{(n)}, w^{\{t\}} \phi(x^{(n)}, v^{\{t\}}))$$

$$= e^{-w^{\{t\}}} \sum_n q^{(n)} + (e^{w^{\{t\}}} - e^{-w^{\{t\}}}) \sum_n q^{(n)} \mathbb{I}(y^{(n)} \neq \phi(x^{(n)}, v^{\{t\}}))$$

does not depend on the weak learner

assuming $w^{\{t\}} \geq 0$ the weak learner should minimize this cost
this is classification with weighted instances
this gives $v^{\{t\}}$

still need to find the optimal $w^{\{t\}}$

setting $\frac{\partial J}{\partial w^{\{t\}}} = 0$ gives $w^{\{t\}} = \frac{1}{2} \log \frac{1-\ell^{\{t\}}}{\ell^{\{t\}}}$

weight-normalized misclassification error

$$\ell^{\{t\}} = \frac{\sum_n q^{(n)} \mathbb{I}(\phi(x^{(n)}; v^{\{t\}}) \neq y^{(n)})}{\sum_n q^{(n)}}$$

since weak learner is better than chance $\ell^{\{t\}} < .5$ and so $w^{\{t\}} \geq 0$

we can now update instance weights q for next iteration $q^{(n),\{t+1\}} = q^{(n),\{t\}} e^{-w^{\{t\}} y^{(n)} \phi(x^{(n)}; v^{\{t\}})}$

(multiply by the new loss)

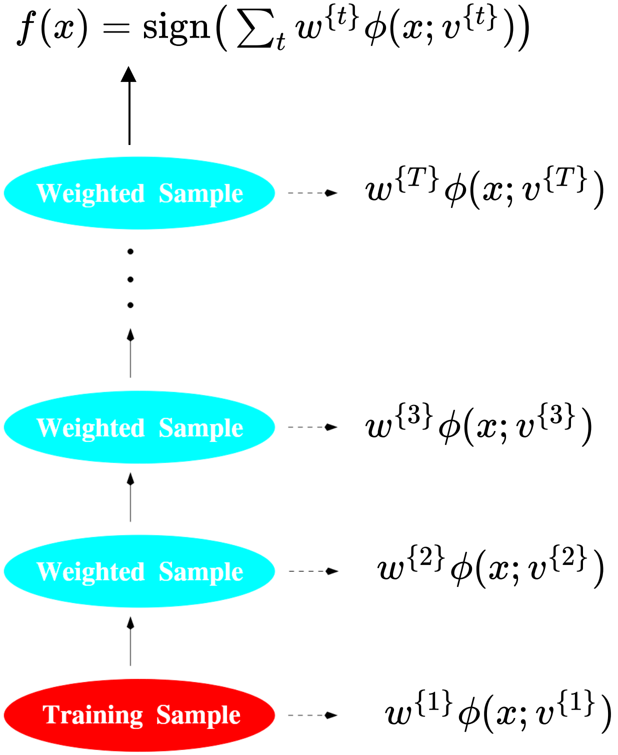
since w > 0, the weight q of misclassified points increase and the rest decrease

optional

Exponential loss & **AdaBoost**

overall algorithm for discrete AdaBoost

```
initialize $q^{(n)} := \frac{1}{N} \quad \forall n$
for t=1:T
    fit the simple classifier $\phi(x, v^{\{t\}})$ to the weighted dataset
    $\ell^{\{t\}} := \frac{\sum_n q^{(n)} \mathbb{I}(\phi(x^{(n)}; v^{\{t\}}) \neq y^{(n)})}{\sum_n q^{(n)}}$
    $w^{\{t\}} := \frac{1}{2} \log \frac{1 - \ell^{\{t\}}}{\ell^{\{t\}}}$
    $q^{(n)} := q^{(n)} e^{-w^{\{t\}} y^{(n)} \phi(x^{(n)}; v^{\{t\}})} \quad \forall n$
return $f(x) = sign\left( \sum_t w^{\{t\}} \phi(x; v^{\{t\}}) \right))$
```



$f(x) = \text{sign}\left(\sum_t w^{\{t\}} \phi(x; v^{\{t\}}) \right))$

Weighted Sample $\dashrightarrow$ $w^{\{T\}} \phi(x; v^{\{T\}})$

Weighted Sample $\dashrightarrow$ $w^{\{3\}} \phi(x; v^{\{3\}})$

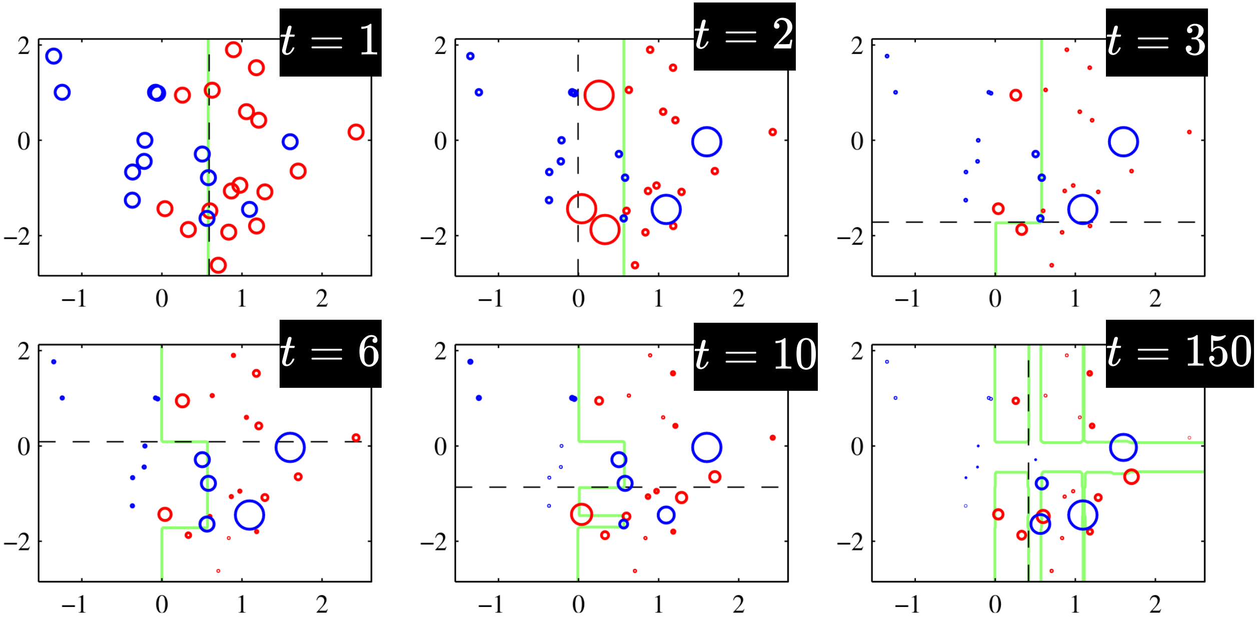
Weighted Sample $\dashrightarrow$ $w^{\{2\}} \phi(x; v^{\{2\}})$

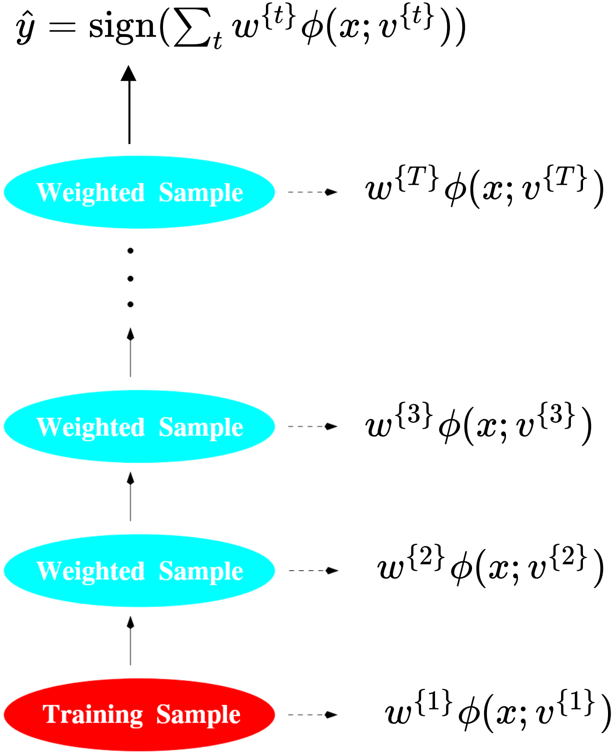
Training Sample $\dashrightarrow$ $w^{\{1\}} \phi(x; v^{\{1\}})$

AdaBoost example

each weak learner is a decision stump (dashed line)

green is the decision boundary of $f^{\{t\}}$



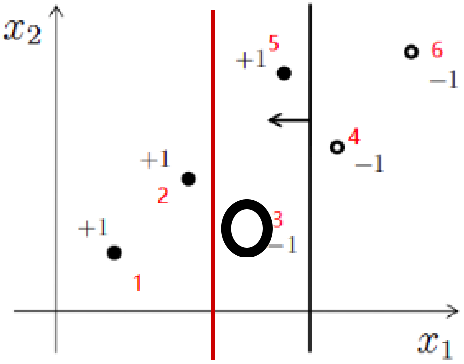


$\hat{y} = \text{sign}(\sum_t w^{\{t\}} \phi(x; v^{\{t\}}))$
Weighted Sample
$w^{\{T\}} \phi(x; v^{\{T\}})$
Weighted Sample
$w^{\{3\}} \phi(x; v^{\{3\}})$
Weighted Sample
$w^{\{2\}} \phi(x; v^{\{2\}})$
Training Sample
$w^{\{1\}} \phi(x; v^{\{1\}})$

Example optional

Discrete AdaBoost Algorithm

```
initialize $q^{(n)} := \frac{1}{N} \quad \forall n$
for t=1:T
    fit the simple classifier $\phi(x, v^{\{t\}})$ to the weighted dataset
    $\ell^{\{t\}} := \frac{\sum_n q^{(n)} \mathbb{I}(\phi(x^{(n)}; v^{\{t\}}) \neq y^{(n)})}{\sum_n q^{(n)}}$
    $w^{\{t\}} := \frac{1}{2} \log \frac{1-\ell^{\{t\}}}{\ell^{\{t\}}}$
    $q^{(n)} := q^{(n)} e^{-w^{\{t\}} y^{(n)} \phi(x^{(n)}; v^{\{t\}})} \quad \forall n$
return $f(x) = sign\left(\sum_t w^{\{t\}} \phi(x; v^{\{t\}})\right))$
```



$$q := [\tfrac{1}{6}, \tfrac{1}{6}, \tfrac{1}{6}, \tfrac{1}{6}, \tfrac{1}{6}, \tfrac{1}{6}]$$

$$\phi^{\{1\}} = [1, 1, 1, -1, 1, -1]$$

$$\ell = \frac{\frac{1}{6}}{\sum q} = \frac{1}{6}$$

$$w = \frac{1}{2} \log\left(\frac{1-\frac{1}{6}}{\frac{1}{6}}\right) = .5 \log(5) \approx 0.8$$

$$q := [\tfrac{1}{6\sqrt{5}}, \tfrac{1}{6\sqrt{5}}, \tfrac{\sqrt{5}}{6}, \tfrac{1}{6\sqrt{5}}, \tfrac{1}{6\sqrt{5}}, \tfrac{1}{6\sqrt{5}}]$$

$$\phi^{\{2\}} = [1, 1, -1, -1, -1, -1]$$

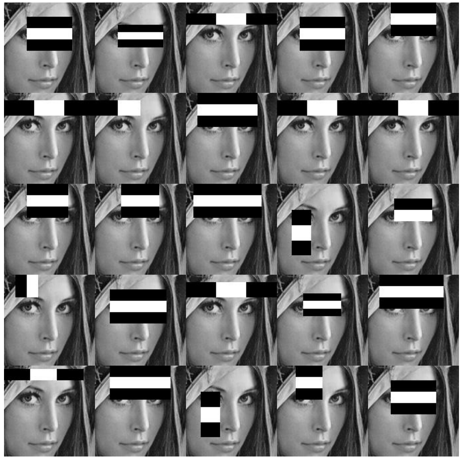
$$\ell = \frac{\frac{1}{6\sqrt{5}}}{\sum q} = \frac{1}{10}$$

$$w = \frac{1}{2} \log\left(\frac{1-\frac{1}{10}}{\frac{1}{10}}\right) = .5 \log(9) \approx 1.1$$

$$f = [sign(.8 + 1.1), sign(.8 + 1.1), sign(.8 - 1.1), sign(-.8 - 1.1), sign(.8 - 1.1), sign(-.8 - 1.1)]$$

optional

application: Viola-Jones face detection



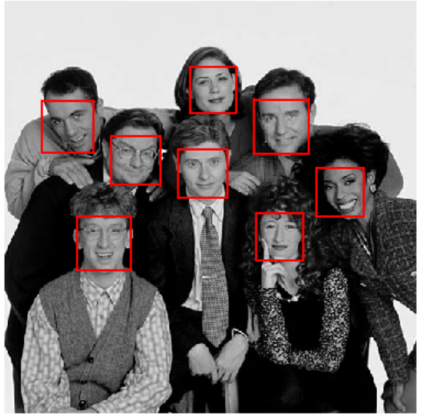
The first face detector

each feature is a weak learner, Haar features

- only compares the total intensity in rectangular pieces of the image, computationally efficient

fast enough for real-time (object) detection

AdaBoost picks one feature at a time (label: face/no-face)



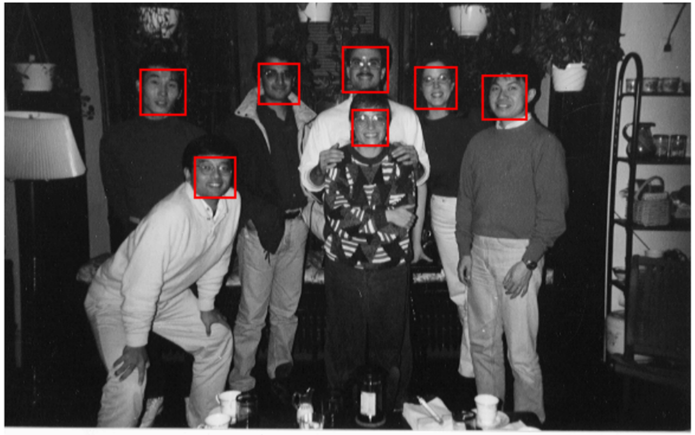


image credit: David Lowe

optional

Boosting

model $f(x) = \sum_{t=1}^{T} w^{\{t\}} \phi(x; v^{\{t\}})$ a simple model, such as decision stump (decision tree with one node)

cost $J(\{w^{\{t\}}, v^{\{t\}}\}_t) = \sum_{n=1}^{N} L(y^{(n)}, f(x^{(n)}))$

optimizing this cost is difficult given the form of f

1. find the best weak learner

$$v^{\{t\}}, w^{\{t\}} = \arg\min_{v,w} \sum_{n=1}^{N} L(y^{(n)}, f^{\{t-1\}}(x^{(n)}) + w\phi(x^{(n)}; v))$$

2. add it to the current model

$$f^{\{t\}}(x) = f^{\{t-1\}}(x^{(n)}) + w^{\{t\}}\phi(x^{(n)}; v^{\{t\}})$$

L2Boosting $L(y, f(x)) = \frac{1}{2}(y - f(x))^2$

AdaBoost $L(y, f(x)) = e^{-yf(x)}$

General algorithm for any loss ?

optional: extra

Gradient boosting

idea fit the weak learner to the gradient of the cost

let $\mathbf{f}^{\{t\}} = \left[f^{\{t\}}(x^{(1)}), \ldots, f^{\{t\}}(x^{(N)})\right]^\top$ and true labels $\mathbf{y} = \left[y^{(1)}, \ldots, y^{(N)}\right]^\top$

ignoring the structure of **f**

if we use gradient descent to minimize the loss $\hat{\mathbf{f}} = \arg\min_{\mathbf{f}} L(\mathbf{f}, \mathbf{y})$

$$\mathbf{f}^{\{t\}} = \mathbf{f}^{\{t-1\}} - w^{\{t\}} \mathbf{g}^{\{t\}}$$

$$w^{\{t\}} = \arg\min_w L(\mathbf{f}^{\{t-1\}} - w\mathbf{g}^{\{t\}}) \qquad \frac{\partial}{\partial \mathbf{f}} L(\mathbf{f}^{\{t-1\}}, \mathbf{y})$$

we can look for the optimal step size

optional

gradient vector

its role is similar to residual

write $\hat{\mathbf{f}}$ as a sum of steps $\hat{\mathbf{f}} = \mathbf{f}^{\{T\}} = \mathbf{f}^{\{0\}} - \sum_{t=1}^{T} w^{\{t\}} \mathbf{g}^{\{t\}}$

optional: extra

Gradient boosting

idea fit the weak learner to the gradient of the cost

let $\mathbf{f}^{\{t\}} = \left[f^{\{t\}}(x^{(1)}), \ldots, f^{\{t\}}(x^{(N)})\right]^\top$ and true labels $\mathbf{y} = \left[y^{(1)}, \ldots, y^{(N)}\right]^\top$

ignoring the structure of **f**

if we use gradient descent to minimize the loss $\hat{\mathbf{f}} = \arg\min_{\mathbf{f}} L(\mathbf{f}, \mathbf{y})$

write $\hat{\mathbf{f}}$ as a sum of steps $\hat{\mathbf{f}} = \mathbf{f}^{\{T\}} = \mathbf{f}^{\{0\}} - \sum_{t=1}^{T} w^{\{t\}} \mathbf{g}^{\{t\}}$

$$w^{\{t\}} = \arg\min_w L(\mathbf{f}^{\{t-1\}} - w\mathbf{g}^{\{t\}}) \quad \frac{\partial}{\partial \mathbf{f}} L(\mathbf{f}^{\{t-1\}}, \mathbf{y})$$

we can look for the optimal step size

gradient vector

its role is similar to residual

so far we treated **f** as a parameter vector of input size, **to generalize:**

fit the weak-learner to negative of the gradient $v^{\{t\}} = \arg\min_v \frac{1}{2}||\boldsymbol{\phi}_v - (-\mathbf{g})||_2^2$

we are fitting the gradient using L2 loss regardless of the original loss function

$$\boldsymbol{\phi}_v = \left[\phi(x^{(1)}; v), \ldots, \phi(x^{(N)}; v)\right]^\top$$

$$v^{\{t\}} = \arg\min_v \sum_n ((-g) - \phi(x^{(n)}, v))^2$$

optional: extra

Gradient boosting

```
initialize $f^{\{0\}}(x)$ using a base learner $\arg\min_v \sum_n L(y^{(n)}, \phi(x^{(n)}, v))$
for t=1:T     decide T using a validation set (early stopping)
    calculate the gradient $g^{(n),\{t\}} = \frac{\partial}{\partial f^{\{t-1\}}(x^{(n)})} L(f^{\{t-1\}}(x^{(n)}), y(x^{(n)}))$
    fit a weak learner to negative of gradient using $v^{\{t\}} = \arg\min_v \sum_n (g^{(n),\{t\}} + \phi(x^{(n)}, v))^2$
    find the optimal step size $w^{\{t\}} = \arg\min_w \sum_n L(f^{\{t-1\}}(x^{(n)}) - w g^{(n),\{t\}})$     optional, can use fixed rate as well
    Update the function $f^{\{t\}}(x) = f^{\{t-1\}}(x) + w^{\{t\}} \phi(x, v^{\{t\}})$
return $f^{\{T\}}(x)$
```

We can use different loss functions for example:

$$L(y, f(x)) = \frac{1}{2}(y - f(x))^2 \quad \Rightarrow \quad g = y - f(x)$$

L2Boosting

optional: extra

Gradient tree boosting

apply gradient boosting to CART (classification and regression trees)

```
initialize $\mathbf{f}^{\{0\}}$ to predict a constant
for t=1:T
    calculate the negative of the gradient $\mathbf{r} = -\frac{\partial}{\partial \mathbf{f}} L(\mathbf{f}^{\{t-1\}}, \mathbf{y})$
    fit a regression tree to $\underset{N \times D}{\mathbf{X}}, \underset{N}{\mathbf{r}}$ and produce regions $\mathbb{R}_1, \dots, \mathbb{R}_K$
    re-adjust predictions per region $w_k = \arg\min_w \sum_{x^{(n)} \in \mathbb{R}_k} L(y^{(n)}, f^{\{t-1\}}(x^{(n)}) + w)$
    update $f^{\{t\}}(x) = f^{\{t-1\}}(x) + \alpha \sum_{k=1}^{K} w_k \mathbb{I}(x \in \mathbb{R}_k)$
return $f^{\{T\}}(x)$
```

decide T using a validation set (early stopping)

shallow trees of K = 4-8 leaf usually work well as weak learners

refinement over the generic algorithm, re-adjust the weak learner

using a small learning rate here improves test error (**shrinkage**)

stochastic gradient boosting

- combines bootstrap and boosting
- use a subsample at each iteration above
- similar to stochastic gradient descent

a.k.a **MART**: multiple additive regression trees

XGBoost (extreme gradient boosting) is a widely used variation, which has some additional tricks

optional: extra

Gradient tree boosting example

recall the synthetic example:

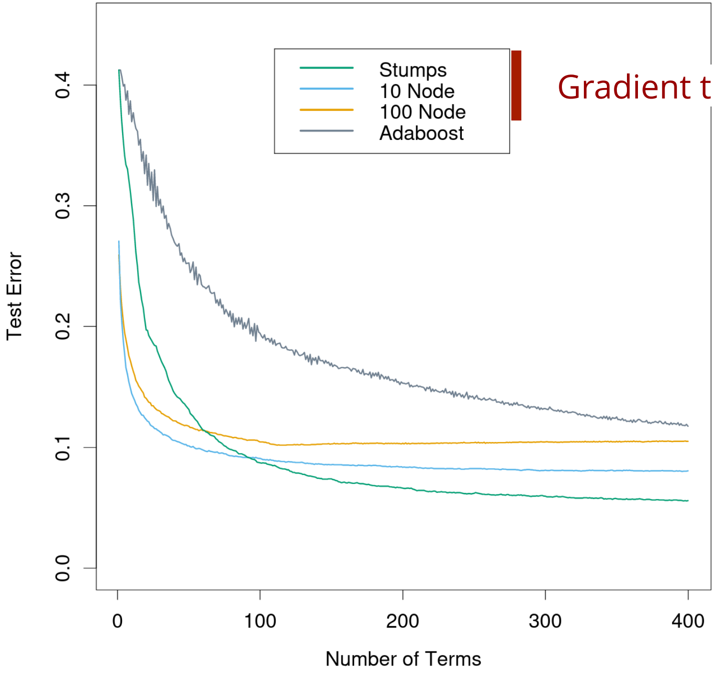
features $x_1^{(n)}, \ldots, x_{10}^{(n)}$ are samples from standard Gaussian

label $y^{(n)} = \mathbb{I}(\sum_d {x_d^{(n)}}^2 > 9.34)$

N=2000 training examples, (~1000+,~1000-)

Boosting with different sized trees

Entropy (a.k.a deviance) for the trees loss



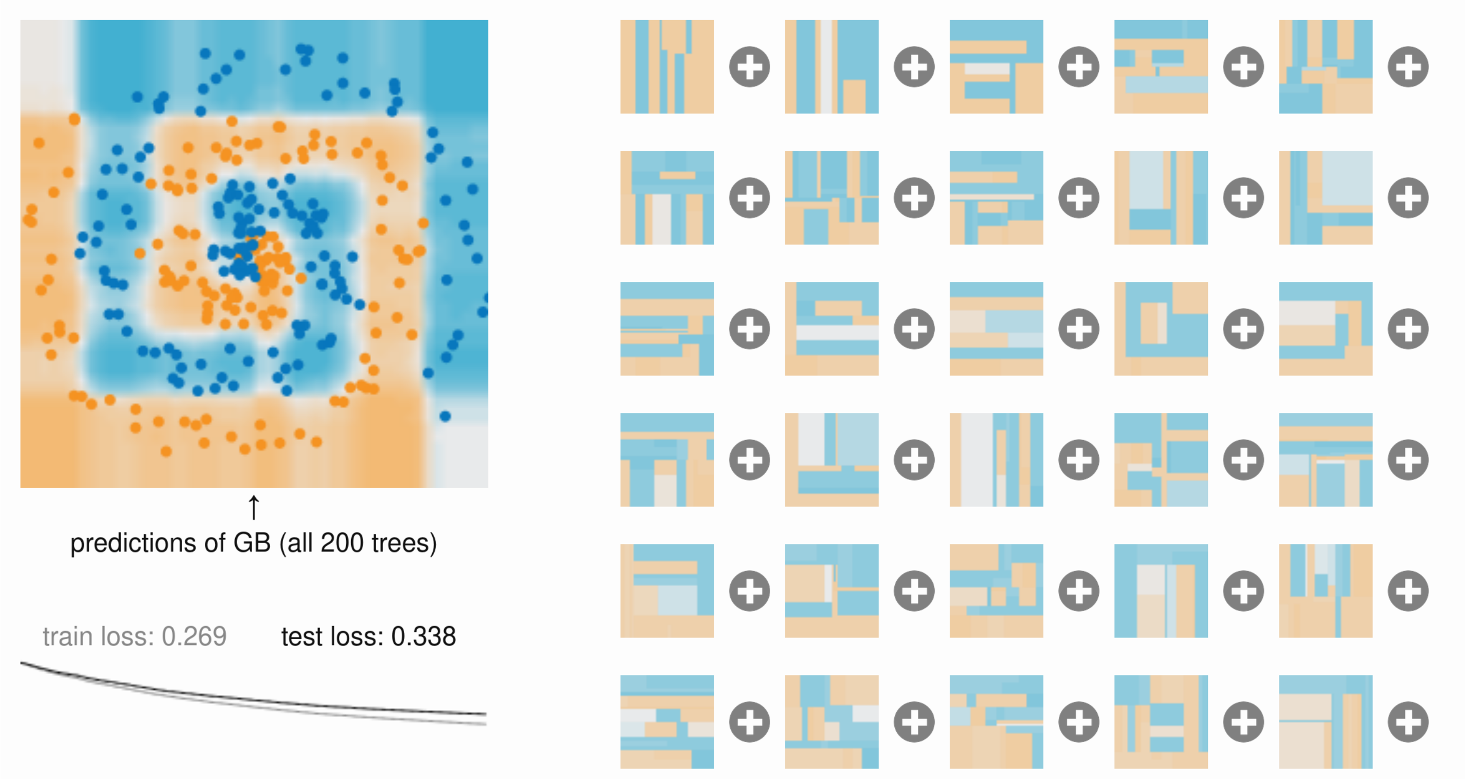
Gradient tree boosting (using log-loss) works better than Adaboost

since sum of features are used in prediction using stumps work best

Gradient tree boosting

example

optional: extra



see the interactive demo: https://arogozhnikov.github.io/2016/07/05/gradient_boosting_playground.html

Summary: Ensemble Methods

- bagging (reduce variance)
    - independent models
    - use their average prediction
    - OOB validation instead of CV
    - Random forests: produce models with minimal correlation
        - destroy interpretability of decision trees
        - perform well in practice
        - can fail if only a few relevant features exist (due to feature-sampling)
- boosting (reduces the bias of the weak learner)
    - models are added in steps
    - a single cost function is minimized
    - for exponential loss: interpret as re-weighting the instance (AdaBoost)
    - gradient boosting: fit the weak learner to the negative of the gradient
    - interpretation as L1 regularization for "weak learner"-selection
- random forests and (gradient) boosting generally perform very well