Caltech

Machine Learning & Data Mining

CS/CNS/EE 155

Lecture 10:

Boosting & Ensemble Selection

Announcements

- Homework 1 is Graded
  - Most people did very well (B+ or higher)
  - 55/60 – 60/60 ≈ A
  - 53/60 – 54/60 ≈ A-
  - 50/60 – 52/60 ≈ B+
  - 45/60 – 49/60 ≈ B
  - 41/60 – 44/60 ≈ B-
  - 37/60 – 40/60 ≈ C+
  - 31/60 – 36/60 ≈ C
  - ≤30/60 ≈ C-

Solutions will be Available On Moodle

Kaggle Mini-Project

- Small bug in data file
  - Was fixed this morning.
  - So if you downloaded already, download again.

- Finding Group Members
  - Offices hours today, mingle in person
  - Online signup sheet later

Today

- High Level Overview of Ensemble Methods

- Boosting
  - Ensemble Method for Reducing Bias

- Ensemble Selection

Recall: Test Error

- **"True" distribution:** P(x,y)
  - Unknown to us
- **Train:** $h_S(x) = y$
  - Using training data: $S = \left\{(x_i, y_i)\right\}_{i=1}^{N}$
  - Sampled from P(x,y)
- **Test Error:**

$$L_P(h_S) = E_{(x,y)\sim P(x,y)}\left[L(y, h_S(x))\right]$$

- **Overfitting:** Test Error >> Training Error

True Distribution P(x,y)

Person	Age	Male?	Height > 55"
James	11	1	1
Jessica	14	0	1
Alice	14	0	1
Amy	12	0	1
Bob	10	1	1
Xavier	9	1	0
Cathy	9	0	1
Carol	13	0	1
Eugene	13	1	0
Rafael	12	1	1
Dave	8	1	0
Peter	9	1	0
Henry	13	1	0
Erin	11	0	0
Rose	7	0	0
Iain	8	1	1
Paulo	12	1	0
Margaret	10	0	1
Frank	9	1	1
Jill	13	0	0
Leon	10	1	0
Sarah	12	0	0
Gena	8	0	0
Patrick	5	1	1

⋮

Training Set S

Person	Age	Male?	Height > 55"	
Alice	14	0	1	✓
Bob	10	1	1	✓
Carol	13	0	1	✓
Dave	8	1	0	✓
Erin	11	0	0	✗
Frank	9	1	1	✗
Gena	8	0	0	✓

y h(x)

Test Error:

$$\mathcal{L}(h) = E_{(x,y)\sim P(x,y)}[\, L(h(x),y)\,]$$

Recall: Test Error

- **Test Error:**

$$L_P(h) = E_{(x,y)\sim P(x,y)}\left[L(y,h(x))\right]$$

- **Treat h_S as random variable:**

$$h_S = \operatorname*{argmin}_h \sum_{(x_i,y_i)\in S} L\left(y_i,h(x)\right)$$

- **Expected Test Error:**

$$E_S\left[L_P(h_S)\right] = E_S\left[E_{(x,y)\sim P(x,y)}\left[L(y,h_S(x))\right]\right]$$

Recall: Bias-Variance Decomposition

$$E_S\left[L_P(h_S)\right] = E_S\left[E_{(x,y)\sim P(x,y)}\left[L(y,h_S(x))\right]\right]$$

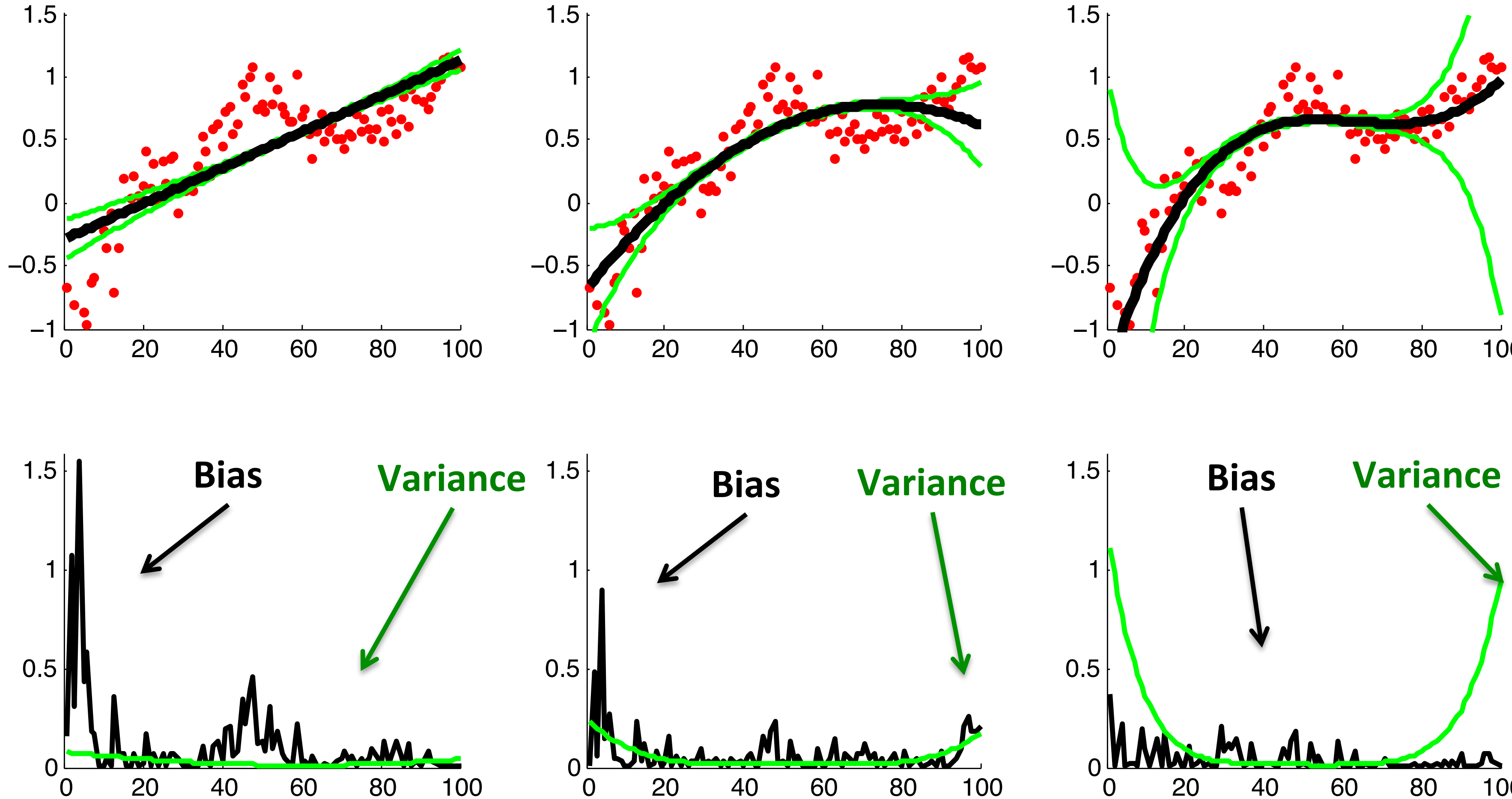
- For squared error:

$$E_S\left[L_P(h_S)\right] = E_{(x,y)\sim P(x,y)}\left[\underbrace{E_S\left[\left(h_S(x)-H(x)\right)^2\right]}_{\text{Variance Term}} + \underbrace{\left(H(x)-y\right)^2}_{\text{Bias Term}}\right]$$

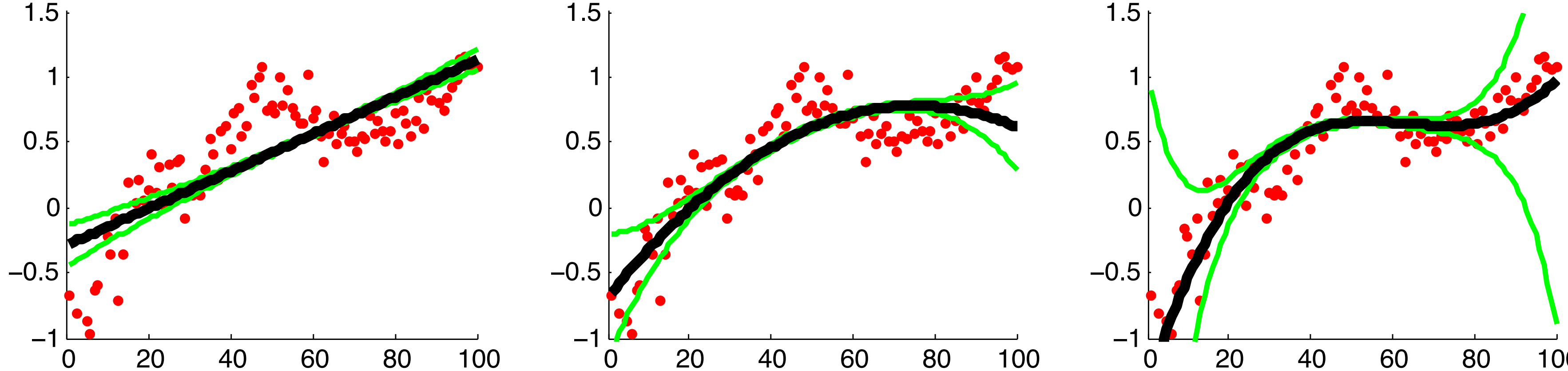
$$H(x) = E_S\left[h_S(x)\right]$$

“Average prediction on x”

Recall: Bias-Variance Decomposition



Recall: Bias-Variance Decomposition



Some models experience high test error due to high bias.
(Model class to simple to make accurate predictions.)

Some models experience high test error due to high variance.
(Model class unstable due to insufficient training data.)

General Concept: Ensemble Methods

- Combine multiple learning algorithms or models
  - Previous Lecture: Bagging
  - Today: Boosting & Ensemble Selection

Decision Trees, SVMs, etc.

- "Meta Learning" approach
  - Does not innovate on base learning algorithm/model
  - Innovates at higher level of abstraction:
    - creating training data and combining resulting base models
    - Bagging creates new training sets via bootstrapping, and then combines by averaging predictions

Intuition: Why Ensemble Methods Work

- **Bias-Variance Tradeoff!**
- **Bagging reduces variance of low-bias models**
  - Low-bias models are “complex” and unstable.
  - Bagging averages them together to create stability
- **Boosting reduces bias of low-variance models**
  - Low-variance models are simple with high bias
  - Boosting trains sequence of models on residual error ➔ sum of simple models is accurate

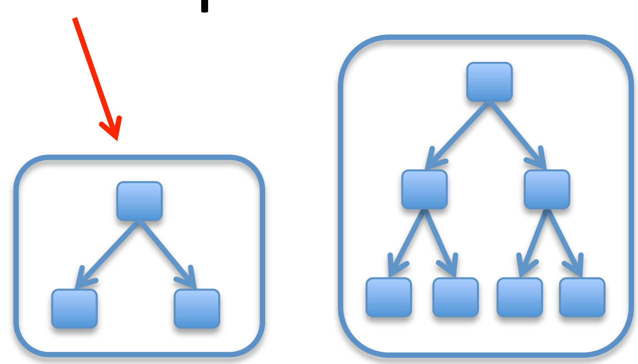
Boosting

“The Strength of Weak Classifiers”*

* http://www.cs.princeton.edu/~schapire/papers/strengthofweak.pdf

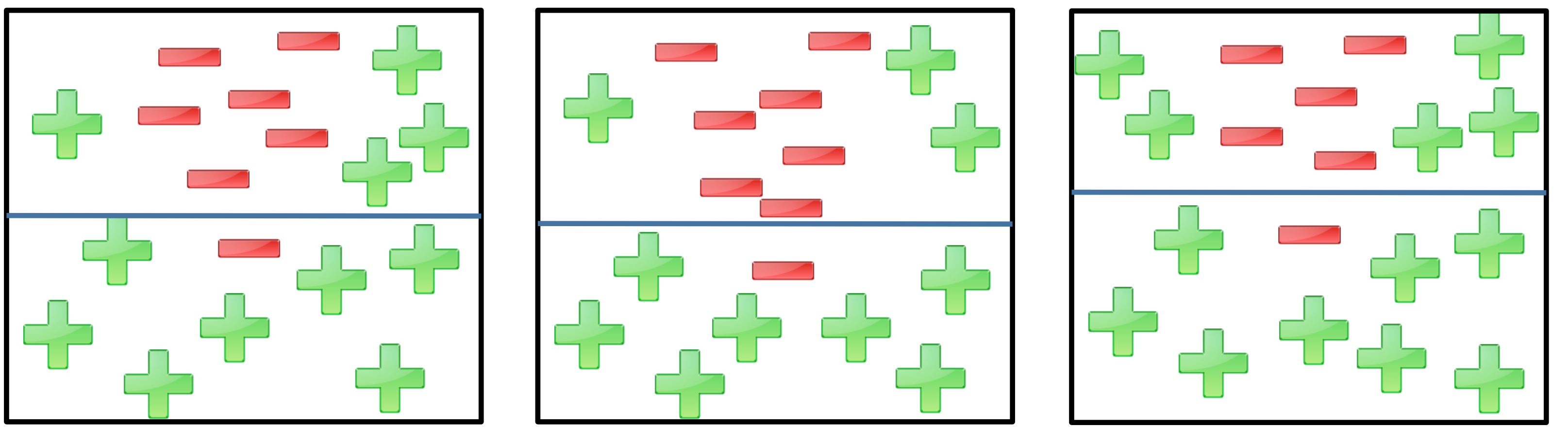
Terminology: Shallow Decision Trees

- Decision Trees with only a few nodes
- Very high bias & low variance
  - Different training sets lead to very similar trees
  - Error is high (barely better than static baseline)
- Extreme case: “Decision Stumps”
  - Trees with exactly 1 split



Stability of Shallow Trees

- Tends to learn more-or-less the same model.
- $h_S(x)$ has low variance
  - Over the randomness of training set S



Terminology: Weak Learning

- **Error rate:** $\varepsilon_{h,P} = E_{P(x,y)}\left[1_{[h(x)\neq y]}\right]$

- **Weak Classifier:** $\varepsilon_{h,P}$ slightly better than 0.5
  - Slightly better than random guessing

- **Weak Learner:** can learn a weak classifier

Terminology: Weak Learning

- **Error rate:** $\varepsilon_{h,P} = E_{P(x,y)}\left[1_{[h(x)\neq y]}\right]$
- **Weak Classifier:** $\varepsilon_{h,P}$ slightly better than 0.5
  - Slightly better than random guessing

Shallow Decision Trees are Weak Classifiers!

Weak Learners are Low Variance & High Bias!

How to “Boost” Performance of Weak Models?

$$\underbrace{E_S\left[L_P(h_S)\right]}_{\substack{\text{Expected Test Error} \\ \text{Over randomness of S} \\ \text{(Squared Loss)}}} = E_{(x,y)\sim P(x,y)}\left[\underbrace{E_S\left[\left(h_S(x)-H(x)\right)^2\right]}_{\text{Variance Term}} + \underbrace{\left(H(x)-y\right)^2}_{\text{Bias Term}}\right]$$

“Average prediction on x” $\longrightarrow$ $H(x) = E_S\left[h_S(x)\right]$

- Weak Models are High Bias & Low Variance
- Bagging would not work
  – Reduces variance, not bias

First Try (for Regression)

- 1 dimensional regression
- Learn Decision Stump
  - (single split, predict mean of two partitions)

"residual"

$y_t = y - h_{1:t-1}(x)$

$h_{1:t}(x) = h_1(x) + \ldots + h_t(x)$

S

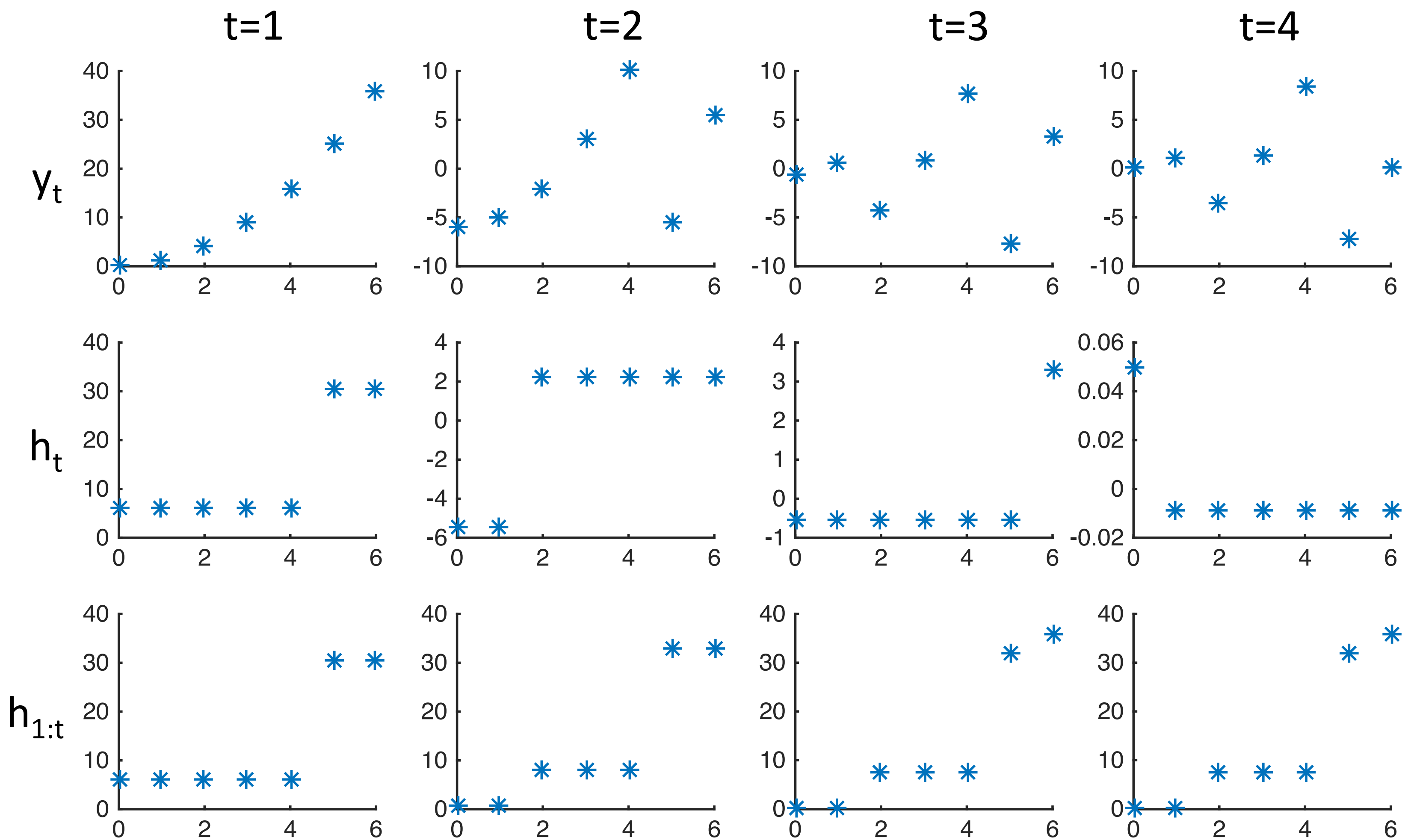
x	y
0	0
1	1
2	4
3	9
4	16
5	25
6	36

y_1	$h_1(x)$	y_2	$h_2(x)$	$h_{1:2}(x)$	y_3	$h_3(x)$	$h_{1:3}(x)$
0	6	-6	-5.5	0.5	-0.5	-0.55	-0.05
1	6	-5	-5.5	0.5	0.5	-0.55	-0.05
4	6	-2	2.2	8.2	-4.2	-0.55	7.65
9	6	-3	2.2	8.2	0.8	-0.55	7.65
16	6	10	2.2	8.2	7.8	-0.55	7.65
25	30.5	-5.5	2.2	32.7	-7.7	-0.55	32.15
36	30.5	5.5	2.2	32.7	3.3	3.3	36

First Try (for Regression)

$$h_{1:t}(x) = h_1(x) + \ldots + h_t(x)$$

$$y_t = y - h_{1:t-1}(x)$$



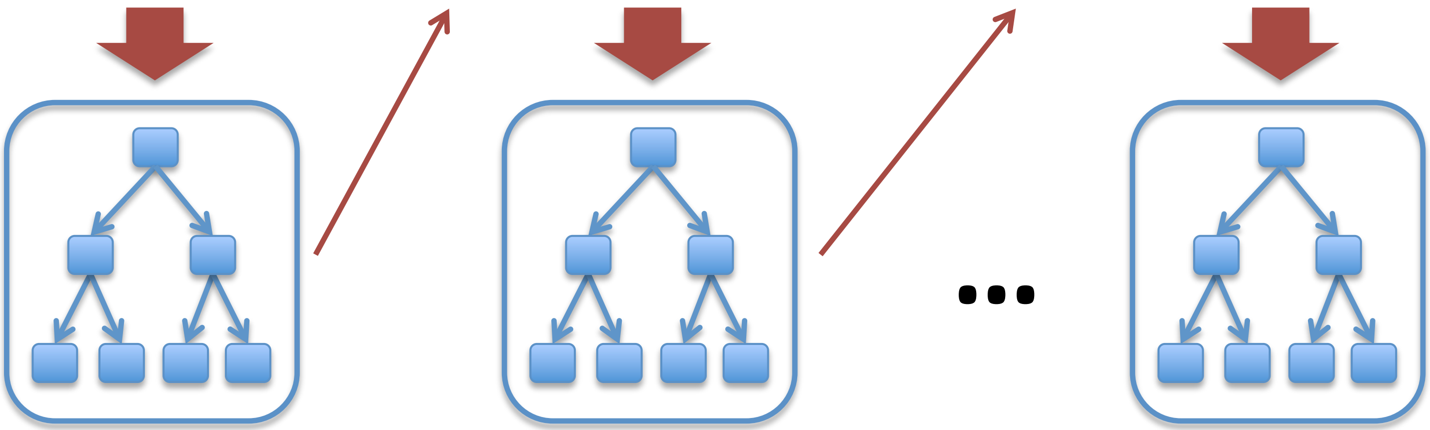
Gradient Boosting (Simple Version)

(Why is it called "gradient"?)
(Answer next slides.)

(For Regression Only)

$$S = \left\{(x_i, y_i)\right\}_{i=1}^{N} \qquad h(x) = h_1(x) + h_2(x) + \ldots + h_n(x)$$

$$S_1 = \left\{(x_i, y_i)\right\}_{i=1}^{N} \rightarrow S_2 = \left\{(x_i, y_i - h_1(x_i))\right\}_{i=1}^{N} \rightarrow S_n = \left\{(x_i, y_i - h_{1:n-1}(x_i))\right\}_{i=1}^{N}$$



$h_1(x)$ $\quad$ $h_2(x)$ $\quad$ … $\quad$ $h_n(x)$

http://statweb.stanford.edu/~jhf/ftp/trebst.pdf

Axis Aligned Gradient Descent

(For Linear Model)

- Linear Model: $h(x) = w^T x$
- Squared Loss: $L(y,y') = (y-y')^2$

Training Set

$$S = \{(x_i, y_i)\}_{i=1}^N$$

- Similar to Gradient Descent
  - But only allow axis-aligned update directions
  - Updates are of the form:

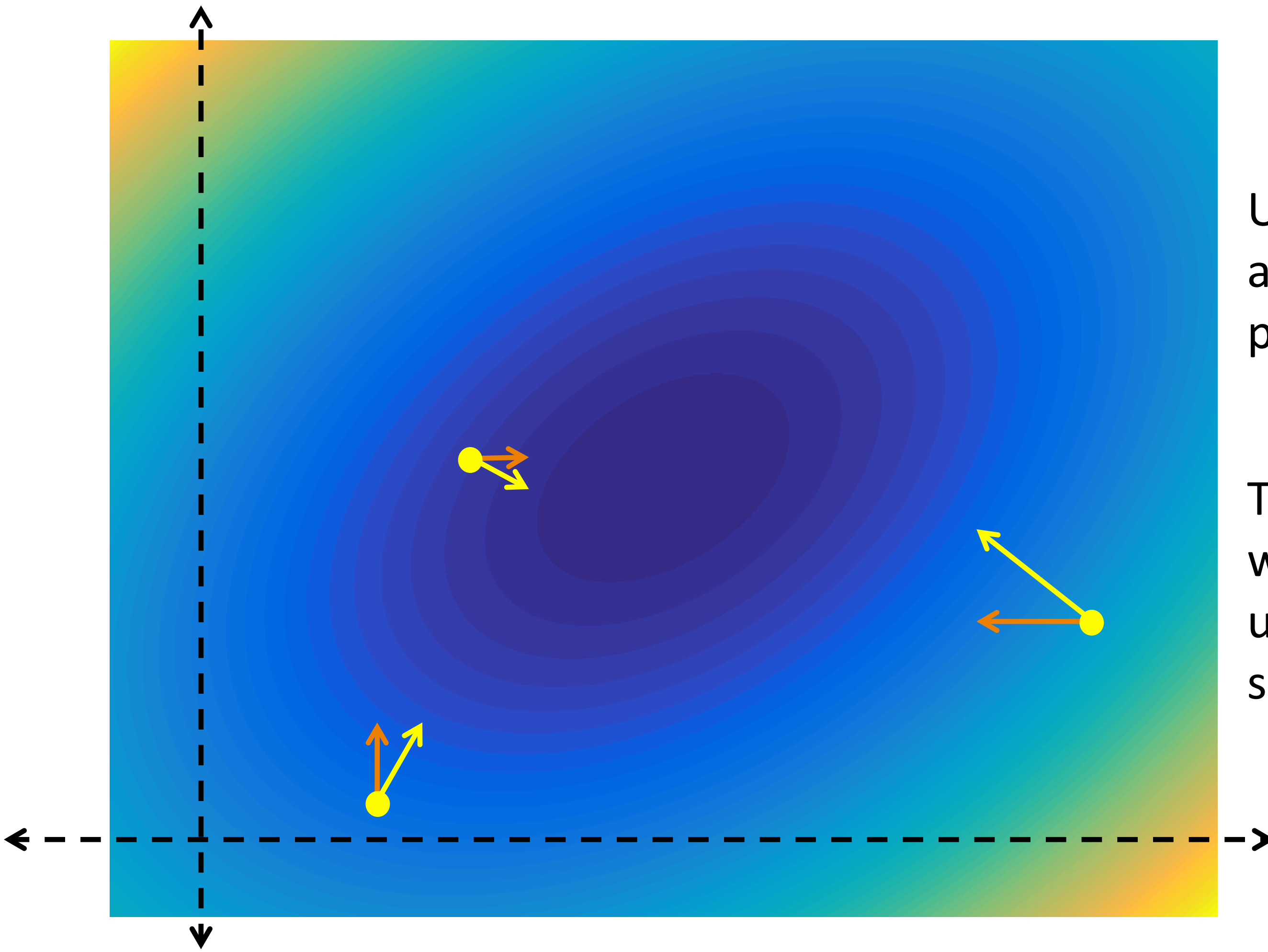
$$w = w - \eta g_d e_d \qquad g = \sum_i \nabla_w L(y_i, w^T x_i)$$

Unit vector along d-th Dimension $e_d = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

Projection of gradient along d-th dimension

Update along axis with greatest projection

Axis Aligned Gradient Descent



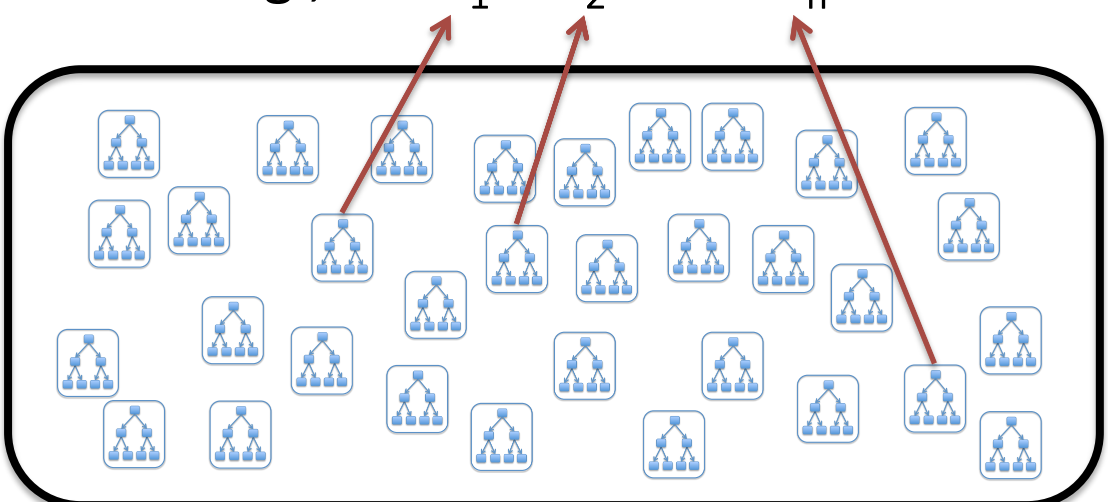
Update along axis with largest projection

This concept will become useful in ~5 slides

Function Space & Ensemble Methods

- Linear model = one coefficient per feature
  - Linear over the input feature space
- Ensemble methods = one coefficient per model
  - Linear over a function space
  - E.g., $h = h_1 + h_2 + \ldots + h_n$

Coefficient=1 for models used
Coefficient=0 for other models



"Function Space"
(All possible shallow trees)
(Potentially infinite)
(Most coefficients are 0)

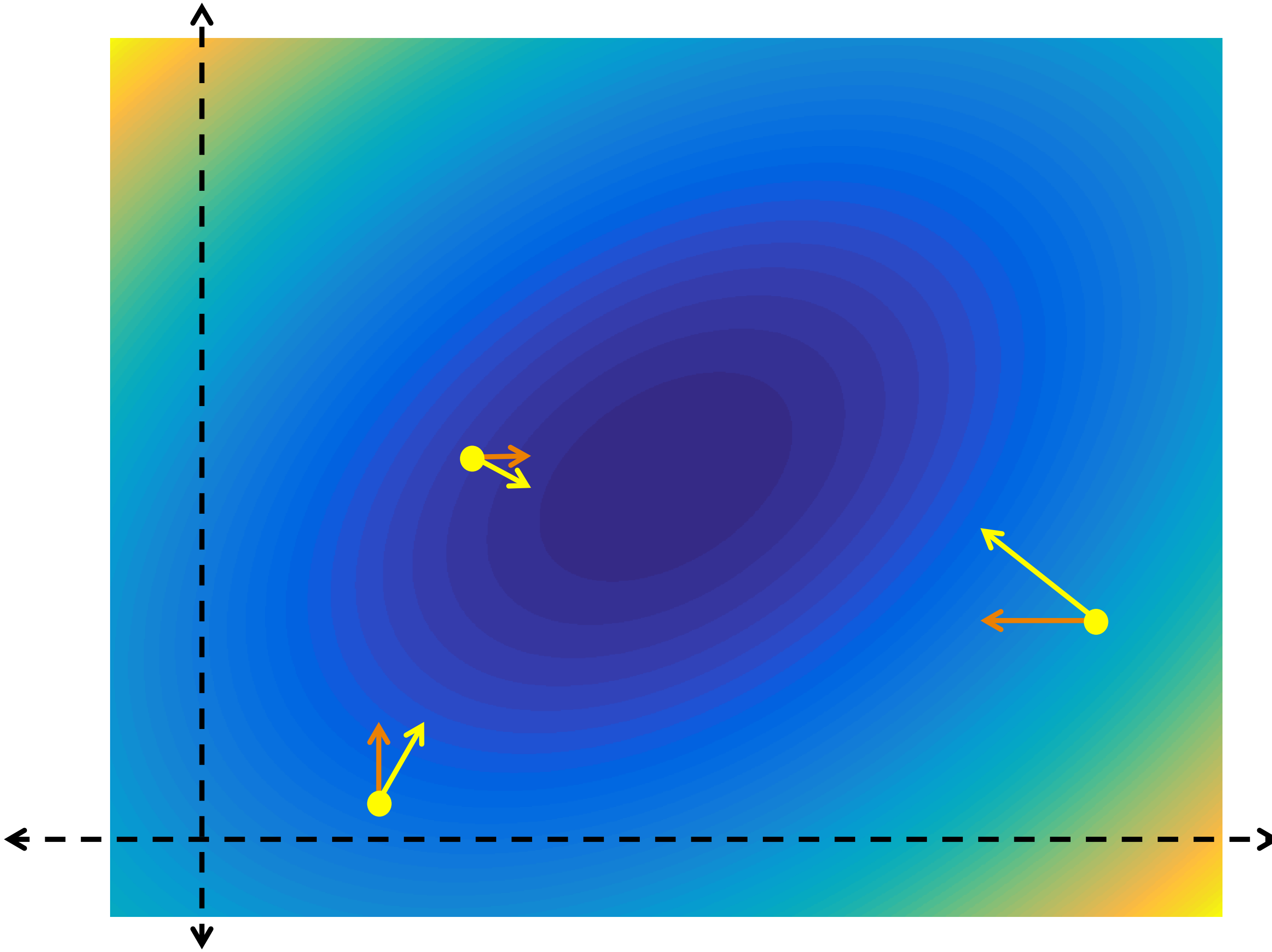
Properties of Function Space

- Generalization of a Vector Space
- Closed under Addition
  - Sum of two functions is a function
- Closed under Scalar Multiplication
  - Multiplying a function with a scalar is a function
- **Gradient descent:** adding a scaled function to an existing function

Function Space of Weak Models

- Every “axis” in the space is a weak model
  - Potentially infinite axes/dimensions
- Complex models are linear combinations of weak models
  - $h = \eta_1 h_1 + \eta_2 h_2 + \ldots + \eta_n h_n$
  - Equivalent to a point in function space
    - Defined by coefficients η

Recall: Axis Aligned Gradient Descent



Project to closest axis & update
(smallest squared dist)

Imagine each axis is a weak model.

Every point is a linear combination of weak models

Functional Gradient Descent

(Gradient Descent in Function Space)
(Derivation for Squared Loss)

- Init h(x) = 0
- Loop n=1,2,3,4,…

$$h = h - \eta \sum_i \nabla_h L(y_i, h(x_i))$$

$$= h + \eta \sum_i \frac{y_i - h(x_i)}{\partial h(x_i)}$$

$$= h + \eta \operatorname*{argmin}_{h_n} \sum_i (y_i - h(x_i) - h_i(x_i))^2$$

Direction of Steepest Descent (aka Gradient) is to add the function that outputs the residual error for each (x_i,y_i)

Projecting to closest weak model = training on the residual

$$\nabla_h L(y_i, h(x_i)) = -\frac{\partial (y_i - h(x_i))^2}{\partial h(x_i)} = -\frac{y_i - h(x_i)}{\partial h(x_i)}$$

$$S = \left\{(x_i, y_i)\right\}_{i=1}^N$$

Reduction to Vector Space

- Function space = axis-aligned unit vectors
  - Weak model = axis-aligned unit vector:

$$e_d = \begin{bmatrix} \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix}$$

- Linear model w has same functional form:
  - $w = \eta_1 e_1 + \eta_2 e_2 + \ldots + \eta_D e_D$
  - Point in space of D "axis-aligned functions"

- **Axis-Aligned Gradient Descent = Functional Gradient Descent on space of axis-aligned unit vector weak models.**

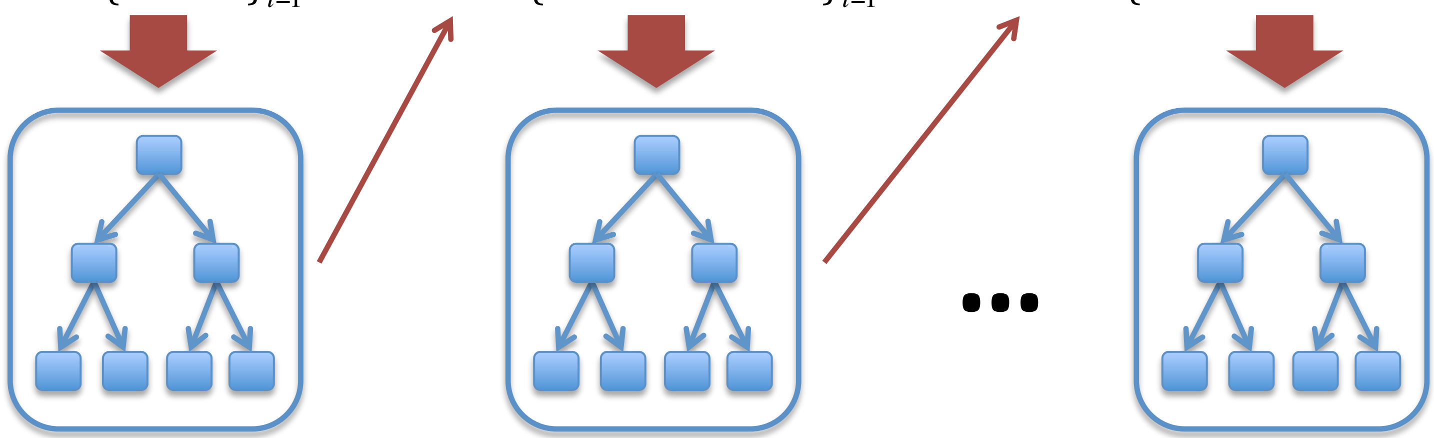
Gradient Boosting (Full Version)

(Instance of Functional Gradient Descent)　　　(For Regression Only)

$$S = \left\{ (x_i, y_i) \right\}_{i=1}^{N}$$

$$h_{1:n}(x) = h_1(x) + \eta_2 h_2(x) + \ldots + \eta_n h_n(x)$$

$$S_1 = \left\{ (x_i, y_i) \right\}_{i=1}^{N} \longrightarrow S_2 = \left\{ (x_i, y_i - h_1(x_i)) \right\}_{i=1}^{N} \longrightarrow S_n = \left\{ (x_i, y_i - h_{1:n-1}(x_i)) \right\}_{i=1}^{N}$$



$h_1(x)$　　$h_2(x)$　　$h_n(x)$

http://statweb.stanford.edu/~jhf/ftp/trebst.pdf ← See reference for how to set η

Recap: Basic Boosting

- Ensemble of many weak classifiers.
  - $h(x) = \eta_1 h_1(x) + \eta_2 h_2(x) + \ldots + \eta_n h_n(x)$
- **Goal:** reduce bias using low-variance models
- **Derivation:** via Gradient Descent in Function Space
  - Space of weak classifiers
- We've only seen the regression so far…

AdaBoost

Adaptive Boosting for Classification

http://www.yisongyue.com/courses/cs155/lectures/msri.pdf

Boosting for Classification

- Gradient Boosting was designed for regression

- Can we design one for classification?

- AdaBoost
  - Adaptive Boosting

AdaBoost = Functional Gradient Descent

- AdaBoost is also instance of functional gradient descent:
  - $h(x) = \text{sign}(\ a_1h_1(x) + a_2h_2(x) + \ldots + a_3h_n(x)\ )$

- E.g., weak models $h_i(x)$ are classification trees
  - Always predict 0 or 1
  - (Gradient Boosting used regression trees)

Combining Multiple Classifiers

Aggregate Scoring Function:

$$f(x) = 0.1*h_1(x) + 1.5*h_2(x) + 0.4*h_3(x) + 1.1*h_4(x)$$

Aggregate Classifier:

$$h(x) = \text{sign}(f(x))$$

Data Point	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$	$f(x)$	$h(x)$
x_1	+1	+1	+1	-1	0.1 + 1.5 + 0.4 - 1.1 = 0.9	+1
x_2	+1	+1	+1	+1	0.1 + 1.5 + 0.4 + 1.1 = 3.1	+1
x_3	-1	+1	-1	-1	-0.1 + 1.5 – 0.3 – 1.1 = -0.1	-1
x_4	-1	-1	+1	-1	-0.1 – 1.5 + 0.3 – 1.1 = -2.4	-1

Also Creates New Training Sets

For Regression

- Gradients in Function Space
  - Weak model that outputs residual of loss function
    - Squared loss = $y-h(x)$
  - **Algorithmically equivalent to training weak model on modified training set**
    - Gradient Boosting = train on $(x_i, y_i–h(x_i))$
- **What about AdaBoost?**
  - **Classification problem.**

Reweighting Training Data

- Define weighting D over S: $S = \{(x_i, y_i)\}_{i=1}^{N}$
  - Sums to 1: $\sum_i D(i) = 1$
- Examples:

Data Point	D(i)
(x_1,y_1)	1/3
(x_2,y_2)	1/3
(x_3,y_3)	1/3

Data Point	D(i)
(x_1,y_1)	0
(x_2,y_2)	1/2
(x_3,y_3)	1/2

Data Point	D(i)
(x_1,y_1)	1/6
(x_2,y_2)	1/3
(x_3,y_3)	1/2

- Weighted loss function:

$$L_D(h) = \sum_i D(i) L(y_i, h(x_i))$$

Training Decision Trees with Weighted Training Data

- Slight modification of splitting criterion.

- Example: Bernoulli Variance:

$$L(S') = |S'| p_{S'}(1 - p_{S'}) = \frac{\#pos * \#neg}{|S'|}$$

- Estimate fraction of positives as:

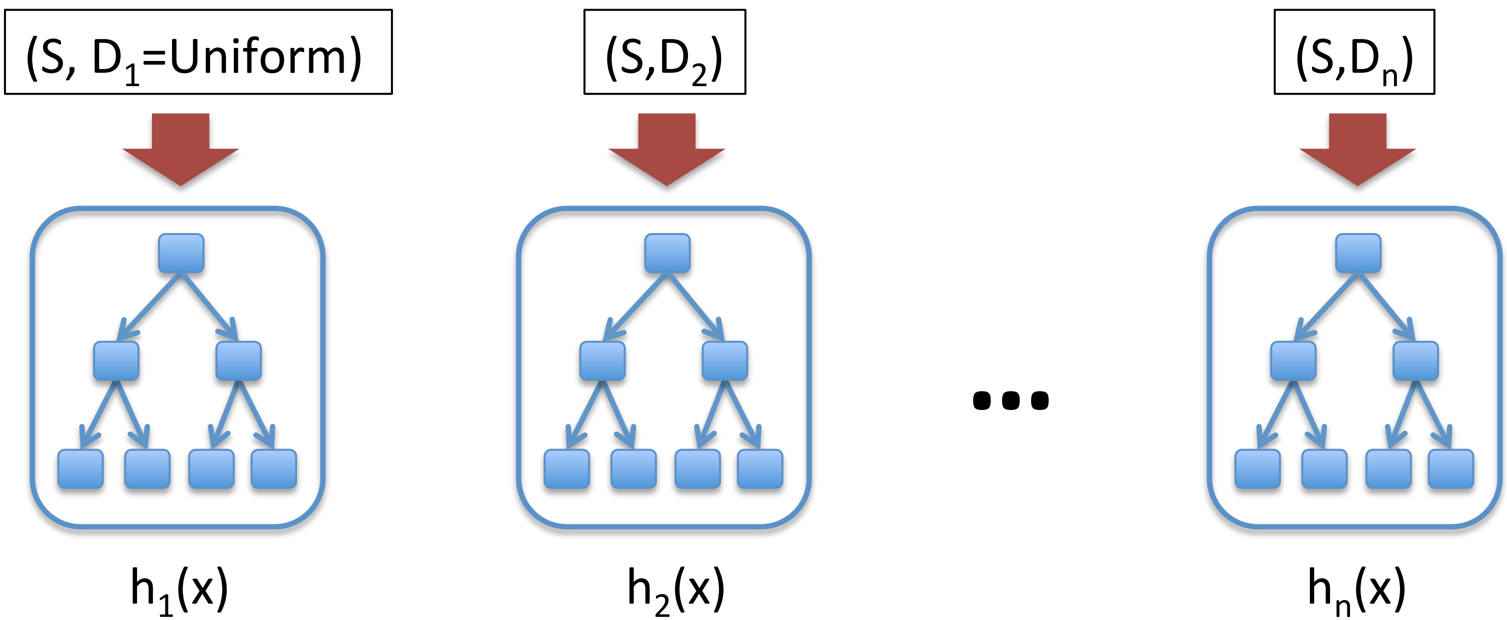
$$p_{S'} = \frac{\sum_{(x_i, y_i) \in S'} D(i) 1_{[y_i = 1]}}{|S'|} \qquad |S'| \equiv \sum_{(x_i, y_i) \in S'} D(i)$$

AdaBoost Outline

$$S = \{(x_i, y_i)\}_{i=1}^{N}$$

$$h(x) = \mathrm{sign}(a_1 h_1(x) + a_2 h_2(x) + \ldots + a_n h_n(x))$$

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



D_t – weighting on data points
a_t – weight of linear combination

Stop when validation performance plateaus (will discuss later)

http://www.yisongyue.com/courses/cs155/lectures/msri.pdf

Intuition

Aggregate Scoring Function:

$$f(x) = 0.1*h_1(x) + 1.5*h_2(x) + 0.4*h_3(x) + 1.1*h_4(x)$$

Aggregate Classifier:

$$h(x) = \text{sign}(f(x))$$

Somewhat close to Decision Boundary

Violates Decision Boundary

Safely Far from Decision Boundary

Data Point	Label	f(x)	h(x)
x_1	y_1=+1	0.9	+1
x_2	y_2=+1	3.1	+1
x_3	y_3=+1	-0.1	-1
x_4	y_4=-1	-2.4	-1

Intuition

Thought Experiment:

When we train new $h_5(x)$ to add to $f(x)$…

… what happens when h_5 mispredicts on everything?

Somewhat close to Decision Boundary

Violates Decision Boundary

Safely Far from Decision Boundary

Data Point	Label	f(x)	h(x)
x_1	y_1=+1	0.9	+1
x_2	y_2=+1	3.1	+1
x_3	y_3=+1	-0.1	-1
x_4	y_4=-1	-2.4	-1

Intuition

Aggregate Scoring Function:

$$f_{1:5}(x) = f_{1:4}(x) + 0.5 * h_5(x)$$

Aggregate Classifier:

$$h_{1:5}(x) = \text{sign}(f_{1:5}(x))$$

Suppose $a_5 = 0.5$

Data Point	Label	$f_{1:4}(x)$	$h_{1:4}(x)$	Worst case $h_5(x)$	Worst case $f_{1:5}(x)$	Impact of $h_5(x)$
x_1	y_1=+1	0.9	+1	-1	0.4	Kind of Bad
x_2	y_2=+1	3.1	+1	-1	2.6	Irrelevant
x_3	y_3=+1	-0.1	-1	-1	-0.6	Very Bad
x_4	y_4=-1	-2.4	-1	+1	-1.9	Irrelevant

$h_5(x)$ that mispredicts on everything

Intuition

$h_5(x)$ should definitely classify (x_3,y_3) correctly!
$h_5(x)$ should probably classify (x_1,y_1) correctly.
Don't care about (x_2,y_2) & (x_4,y_4)
Implies a weighting over training examples

Data Point	Label	$f_{1:4}(x)$	$h_{1:4}(x)$	Worst case $h_5(x)$	Worst case $f_{1:5}(x)$	Impact of $h_5(x)$
x_1	y_1=+1	0.9	+1	-1	0.4	Kind of Bad
x_2	y_2=+1	3.1	+1	-1	2.6	Irrelevant
x_3	y_3=+1	-0.1	-1	-1	-0.6	Very Bad
x_4	y_4=-1	-2.4	-1	+1	-1.9	Irrelevant

$h_5(x)$ that mispredicts on everything

Intuition

Aggregate Scoring Function:

$$f_{1:4}(x) = 0.1*h_1(x) + 1.5*h_2(x) + 0.4*h_3(x) + 1.1*h_4(x)$$

Aggregate Classifier:

$$h_{1:4}(x) = \text{sign}(f_{1:4}(x))$$

Data Point	Label	$f_{1:4}(x)$	$h_{1:4}(x)$	Desired D_5
x_1	y_1=+1	0.9	+1	**Medium**
x_2	y_2=+1	3.1	+1	**Low**
x_3	y_3=+1	-0.1	-1	**High**
x_4	y_4=-1	-2.4	-1	**Low**

AdaBoost

$$S = \{(x_i, y_i)\}_{i=1}^{N}$$

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

- Init $D_1(x) = 1/N$
- Loop $t = 1 \ldots n$:
  - Train classifier $h_t(x)$ using (S, D_t)
  - Compute error on (S, D_t): $\varepsilon_t \equiv L_{D_t}(h_t) = \sum_i D_t(i) L(y_i, h_t(x_i))$
  - Define step size a_t: $a_t = \frac{1}{2}\log\left\{\frac{1-\varepsilon_t}{\varepsilon_t}\right\}$
  - Update Weighting: $D_{t+1}(i) = \frac{D_t(i)\exp\{-a_t y_i h_t(x_i)\}}{Z_t}$
- **Return:** $h(x) = \text{sign}(a_1 h_1(x) + \ldots + a_n h_n(x))$

Normalization Factor s.t. D_{t+1} sums to 1.

http://www.yisongyue.com/courses/cs155/lectures/msri.pdf

Example

$y_i h_t(x_i)$ = -1 or +1

$$\varepsilon_t \equiv L_{D_t}(h_t) = \sum_i D_t(i) L(y_i, h_t(x_i))$$

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

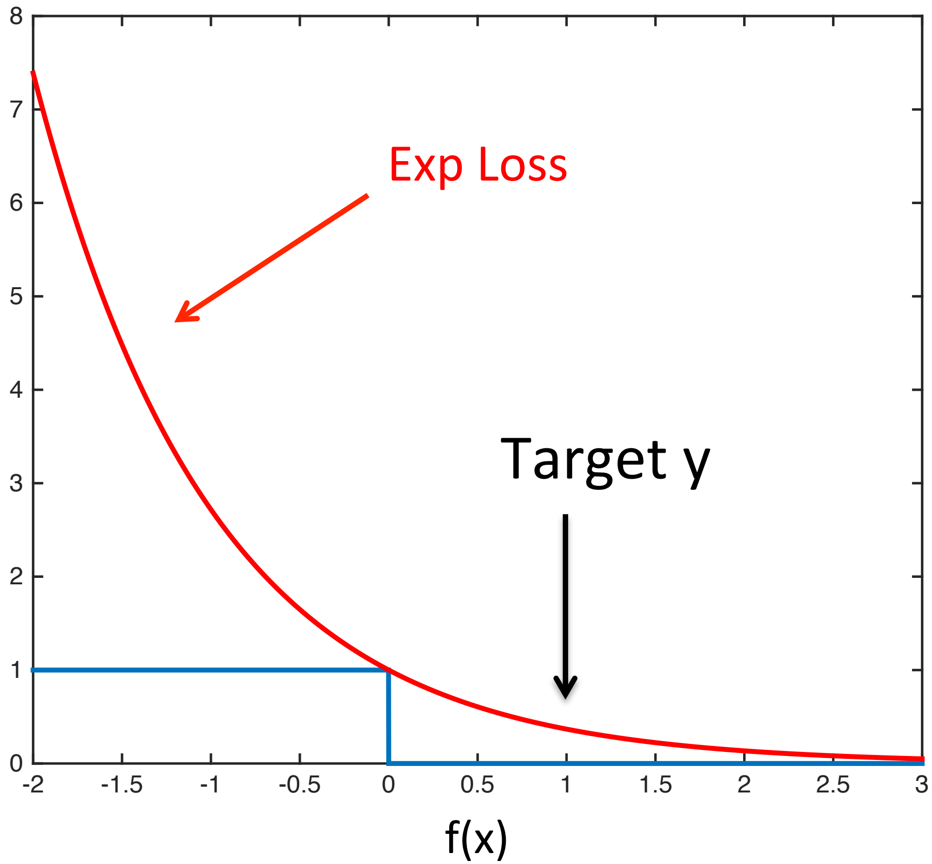
$$a_t = \frac{1}{2}\log\left\{\frac{1-\varepsilon_t}{\varepsilon_t}\right\}$$

Normalization Factor s.t. D_{t+1} sums to 1.

			ε_1=0.4 a_1=0.2		ε_2=0.45 a_2=0.1		ε_3=0.35 a_3=0.31
Data Point	**Label**	**D_1**	**$h_1(x)$**	**D_2**	**$h_2(x)$**	**D3**	**$h_3(x)$**
x_1	y_1=+1	0.01	+1	0.008	+1	0.007	-1
x_2	y_2=+1	0.01	-1	0.012	+1	0.011	+1
x_3	y_3=+1	0.01	-1	0.012	-1	0.013	+1
x_4	y_4=-1	0.01	-1	0.008	+1	0.009	-1
⋮			⋮		⋮		⋮

Exponential Loss

$$L(y, f(x)) = \exp\{-yf(x)\}$$



Upper Bounds 0/1 Loss!

Can prove that AdaBoost minimizes Exp Loss
(Homework Question)

Decomposing Exp Loss

$$L(y, f(x)) = \exp\{-yf(x)\}$$

$$= \exp\left\{-y\left(\sum_{t=1}^{n} a_t h_t(x)\right)\right\}$$

$$= \prod_{t=1}^{n} \underbrace{\exp\{-y a_t h_t(x)\}}_{\textbf{Distribution Update Rule!}}$$

http://www.yisongyue.com/courses/cs155/lectures/msri.pdf

Intuition

$$L(y, f(x)) = \exp\left\{-y\sum_{t=1}^{n} a_t h_t(x)\right\} = \prod_{t=1}^{n} \exp\left\{-y a_t h_t(x)\right\}$$

- Exp Loss operates in exponent space
- Additive update to f(x) = multiplicative update to Exp Loss of f(x)
- Reweighting Scheme in AdaBoost can be derived via residual Exp Loss

http://www.yisongyue.com/courses/cs155/lectures/msri.pdf

AdaBoost = Minimizing Exp Loss

$$S = \{(x_i, y_i)\}_{i=1}^{N}$$

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

- Init $D_1(x) = 1/N$
- Loop $t = 1 \ldots n$:
  - Train classifier $h_t(x)$ using (S, D_t)
  - Compute error on (S, D_t): $\varepsilon_t \equiv L_{D_t}(h_t) = \sum_i D_t(i) L(y_i, h_t(x_i))$
  - Define step size a_t: $a_t = \frac{1}{2} \log \left\{ \frac{1-\varepsilon_t}{\varepsilon_t} \right\}$
  - Update Weighting: $D_{t+1}(i) = \frac{D_t(i) \exp\{-a_t y_i h_t(x_i)\}}{Z_t}$
- **Return:** $h(x) = \text{sign}(a_1 h_1(x) + \ldots + a_n h_n(x))$

Data points reweighted according to Exp Loss!

Normalization Factor s.t. D_{t+1} sums to 1.

http://www.yisongyue.com/courses/cs155/lectures/msri.pdf

Story So Far: AdaBoost

- AdaBoost iteratively finds weak classifier to minimize residual Exp Loss
  – Trains weak classifier on reweighted data (S,D_t).

- **Homework: Rigorously prove it!**

The proof is in earlier slides.

1. Formally prove Exp Loss ≥ 0/1 Loss

2. Relate Exp Loss to Z_t:

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

3. Justify choice of a_t:
   - Gives largest decrease in Z_t

$$a_t = \frac{1}{2}\log\left\{\frac{1-\varepsilon_t}{\varepsilon_t}\right\}$$

http://www.yisongyue.com/courses/cs155/lectures/msri.pdf

Recap: AdaBoost

- Gradient Descent in Function Space
  - Space of weak classifiers
- Final model = linear combination of weak classifiers
  - $h(x) = \text{sign}(a_1 h_1(x) + \ldots + a_n h_n(x))$
  - I.e., a point in Function Space
- Iteratively creates new training sets via reweighting
  - Trains weak classifier on reweighted training set
  - Derived via minimizing residual Exp Loss

Ensemble Selection

Recall: Bias-Variance Decomposition

$$E_S\left[L_P(h_S)\right] = E_S\left[E_{(x,y)\sim P(x,y)}\left[L(y,h_S(x))\right]\right]$$

- For squared error:

$$E_S\left[L_P(h_S)\right] = E_{(x,y)\sim P(x,y)}\left[\underbrace{E_S\left[\left(h_S(x)-H(x)\right)^2\right]}_{\text{Variance Term}} + \underbrace{\left(H(x)-y\right)^2}_{\text{Bias Term}}\right]$$

$$H(x) = E_S\left[h_S(x)\right]$$

"Average prediction on x"

Ensemble Methods

- Combine base models to improve performance
- **Bagging:** averages high variance, low bias models
  - Reduces variance
  - Indirectly deals with bias via low bias base models
- **Boosting:** carefully combines simple models
  - Reduces bias
  - Indirectly deals with variance via low variance base models
- **Can we get best of both worlds?**

Insight: Use Validation Set

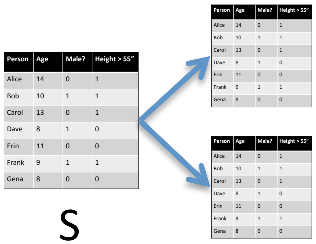
- Evaluate error on validation set V:

$$L_V(h_S) = E_{(x,y)\sim V}\left[L(y, h_S(x))\right]$$

- Proxy for test error:

$$\underbrace{E_V\left[L_V(h_S)\right]}_{\text{Expected Validation Error}} = \underbrace{L_P(h_S)}_{\text{Test Error}}$$

Ensemble Selection



S

Training S'

Validation V'

H = {2000 models trained using S'}

Maintain ensemble model as combination of H:

$h(x) = h_1(x) + h_2(x) + \ldots + h_n(x) \; + h_{n+1}(x)$

Denote as h_{n+1}

Add model from H that maximizes performance on V'

Repeat

Models are trained on S'
Ensemble built to optimize V'

"Ensemble Selection from Libraries of Models"
Caruana, Niculescu-Mizil, Crew & Ksikes, ICML 2004

Reduces Both Bias & Variance

- Expected Test Error = Bias + Variance
- **Bagging:** reduce variance of low-bias models
- **Boosting:** reduce bias of low-variance models
- **Ensemble Selection:** who cares!
  - Use validation error to approximate test error
  - Directly minimize validation error
  - Don't worry about the bias/variance decomposition

What's the Catch?

- Relies heavily on validation set
  - Bagging & Boosting: uses training set to select next model
  - Ensemble Selection: uses validation set to select next model
- Requires validation set be sufficiently large
- **In practice:** implies smaller training sets
  - Training & validation = partitioning of finite data
- Often works very well in practice

MODEL	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	CAL	SAR	MEAN
ENS. SEL.	**0.956**	**0.944**	**0.992**	**0.997**	**0.985**	**0.979**	**0.980**	**0.981**	0.906	**0.996**	**0.969**
BAYESAVG	0.926	0.891	0.979	0.985	0.977	0.956	0.950	0.959	0.907	0.941	0.948
BEST	0.928	0.919	0.975	0.988	0.959	0.958	0.919	0.944	**0.924**	0.924	0.946
AVG_ALL	0.836	0.801	0.982	0.988	0.972	0.961	0.827	0.809	0.832	0.916	0.890
STACK_LR	0.275	0.777	0.835	0.799	0.786	0.847	0.332	-0.990	-0.011	0.705	0.406
SVM	0.813	**0.909**	0.948	0.962	0.933	0.938	**0.877**	0.878	0.889	**0.905**	**0.905**
ANN	0.877	0.875	0.949	0.955	0.917	0.914	0.853	0.863	**0.916**	0.896	0.902
BAG-DT	0.811	0.861	0.947	0.967	0.942	0.922	0.859	**0.894**	0.786	0.904	0.888
KNN	0.756	0.846	0.909	0.937	0.885	0.889	0.761	0.735	0.876	0.847	0.844
BST-DT	**0.890**	0.899	**0.957**	**0.978**	**0.960**	**0.943**	0.607	0.611	0.413	0.871	0.806
DT	0.526	0.789	0.850	0.868	0.767	0.795	0.556	0.624	0.720	0.745	0.722
BST-STMP	0.732	0.790	0.906	0.919	0.861	0.834	0.304	0.286	0.389	0.659	0.669

Ensemble Selection often outperforms a more homogenous sets of models.
Reduces overfitting by building model using validation set.

Ensemble Selection won KDD Cup 2009
http://www.niculescu-mizil.org/papers/KDDCup09.pdf

"Ensemble Selection from Libraries of Models"
Caruana, Niculescu-Mizil, Crew & Ksikes, ICML 2004

References & Further Reading

"An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants" Bauer & Kohavi, Machine Learning, 36, 105–139 (1999)

"Bagging Predictors" Leo Breiman, Tech Report #421, UC Berkeley, 1994, http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf

"An Empirical Comparison of Supervised Learning Algorithms" Caruana & Niculescu-Mizil, ICML 2006

"An Empirical Evaluation of Supervised Learning in High Dimensions" Caruana, Karampatziakis & Yessenalina, ICML 2008

"Ensemble Methods in Machine Learning" Thomas Dietterich*, Multiple Classifier Systems*, 2000

"Ensemble Selection from Libraries of Models" Caruana, Niculescu-Mizil, Crew & Ksikes, ICML 2004

"Getting the Most Out of Ensemble Selection" Caruana, Munson, & Niculescu-Mizil, ICDM 2006

"Explaining AdaBoost" Rob Schapire, https://www.cs.princeton.edu/~schapire/papers/explaining-adaboost.pdf

"Greedy Function Approximation: A Gradient Boosting Machine", Jerome Friedman, 2001, http://statweb.stanford.edu/~jhf/ftp/trebst.pdf

"Random Forests – Random Features" Leo Breiman, Tech Report #567, UC Berkeley, 1999,

"Structured Random Forests for Fast Edge Detection" Dollár & Zitnick, ICCV 2013

"ABC-Boost: Adaptive Base Class Boost for Multi-class Classification" Ping Li, ICML 2009

"Additive Groves of Regression Trees" Sorokina, Caruana & Riedewald, ECML 2007, http://additivegroves.net/

"Winning the KDD Cup Orange Challenge with Ensemble Selection", Niculescu-Mizil et al., KDD 2009

"Lessons from the Netflix Prize Challenge" Bell & Koren, SIGKDD Exporations 9(2), 75—79, 2007

Next Week

- Office Hours Today:
  - Finding group members for mini-project

- Next Week:
  - Extensions of Decision Trees
  - Learning Reductions
    - How to combine binary classifiers to solve more complicated prediction tasks