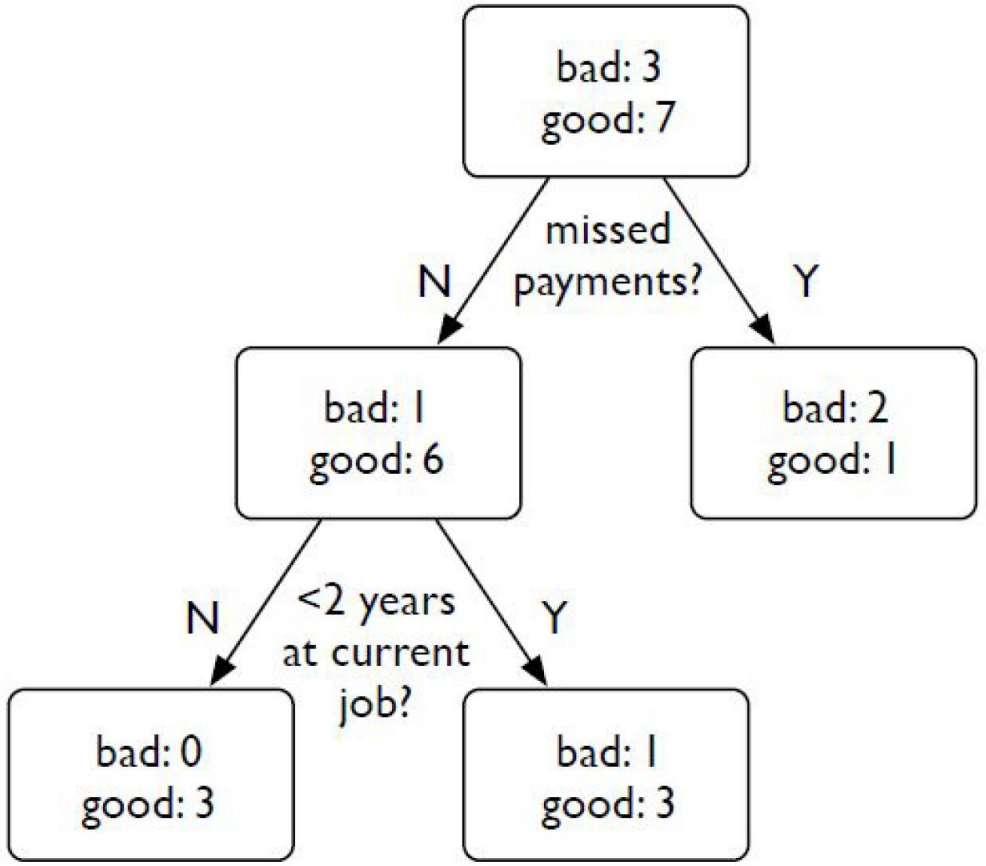
Decision Tree & Ensemble Methods

WEEK 5

Decision Tree: Example

Predicting credit risk

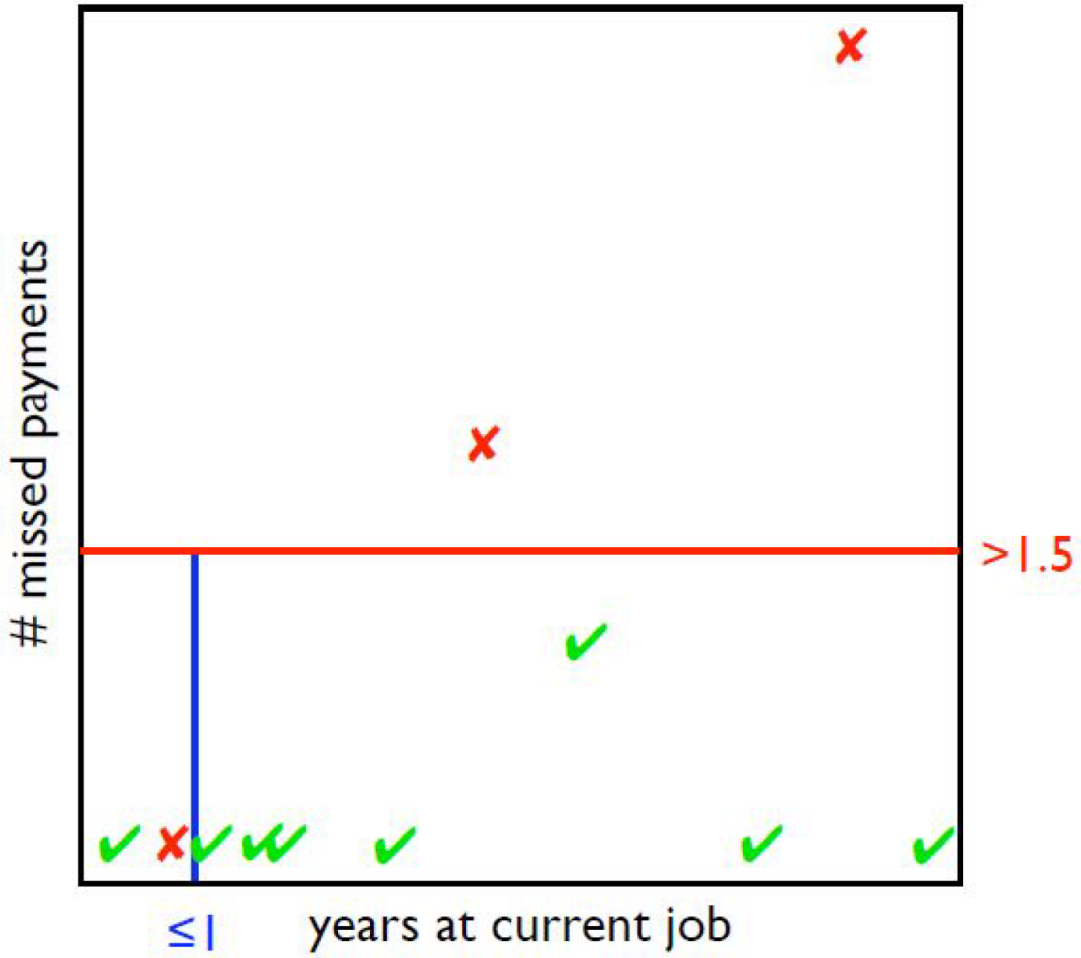
<2 years at current job?	missed payments?	defaulted?
N	N	N
Y	N	Y
N	N	N
N	N	N
N	Y	Y
Y	N	N
N	Y	N
N	Y	Y
Y	N	N
Y	N	N



Decision Tree: Example

Predicting credit risk

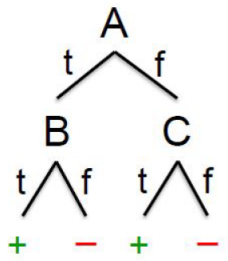
years at current job	# missed payments	defaulted?
7	0	N
0.75	0	Y
3	0	N
9	0	N
4	2	Y
0.25	0	N
5	1	N
8	4	Y
1.0	0	N
1.75	0	N

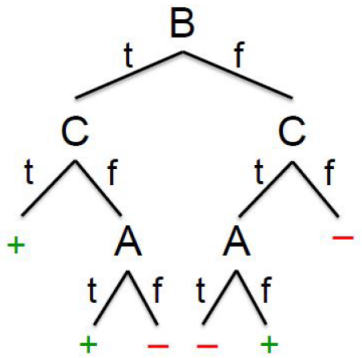


Decision Tree

Many ways to split one dataset, but not all trees will have the same size.

Eg: (A ∧ B) ∨(¬A∧ C)





A	B	C	Y
T	T	T	T
T	T	F	T
T	F	T	F
T	F	F	F
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F

- Learning the simplest (smallest) decision tree is an NP---complete problem [Hyafil & Rivest '76]
- Resort to a greedy heuristic.
  - Start from empty decision tree.
  - Iteratively split on next best feature.

Measures for Selecting the Best Split

How to find the “best split” (best attributes and best cut) ?

- Measure the Impurity I and evaluate the gain Δ

$$\Delta = I(\text{parent}) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

Common metric to measure Impurity: Entropy

$$H(X) \equiv -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x)$$

$$h(X) = -\int_S f(x) \log f(x)\, dx$$

Entropy

For discrete random variable x, the entropy is (unit: bit)

$$H(X) \equiv -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x) = \mathbb{E} \log_2 \left[\frac{1}{p(X)} \right]$$

For a continuous random variable, the differential entropy is

$$h(X) = -\int_S f(x) \log f(x) \, dx \text{ where } \quad S_X = \{x : f(x) > 0\}$$

Intuitively, the entropy gives a measure of the uncertainty of the random variable.

Entropy : Example

For binary variables:

A one-sided coin has entropy 0.

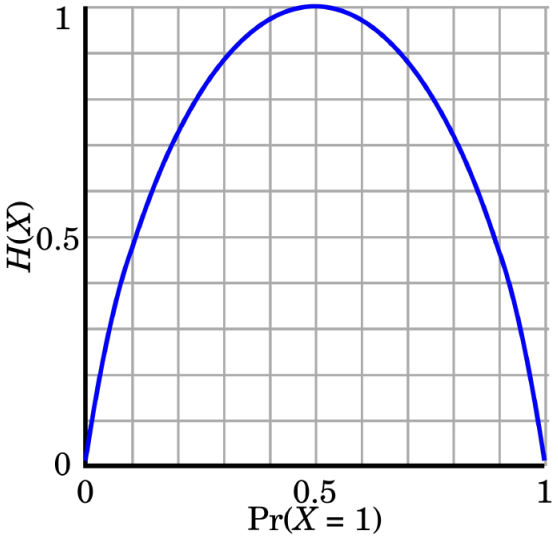
- 1 * log(1) = 0

An unfair coin with probability [0.1, 0.9] of each side has entropy 0.469 bit.

- 0.9 * log(0.9) - 0.1 * log(0.1) = 0.469

A fair coin has two values with equal probability. Its entropy is 1 bit.

- 2 * 0.5 * log(0.5) = 1



Measures for Selecting the Best Split

- Splitting binary categorical attributes
  - Calculate the entropy at each child node (split).
  - $H_{child} = \sum_j (N_j / N) H_j$
  - Measure information gain H_{parent} - H_{child} .

- Splitting multi-class categorical attributes
  - Multiway splits - calculate the weighted sum of all values in the attributes.
  - Could combine child node to reduce the number of splits.
  - Could also apply binary split by grouping attributes.

- Splitting continuous attributes
  - Look for split position s that leads to the most gain in $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$
  - At each attributes: O(N^2) for brute force method, O(NlogN) with cleverer methods.

Measures for Selecting the Best Split

- Alternatives for measuring "gain"
  - Gini index $\iota_{Gini}(p_1, p_2) = 1 - ({p_1}^2 + {p_2}^2) = 2 \cdot p_1 \cdot p_2$
  - Gain Ratio $\text{Gain ratio} = \frac{\Delta_{\text{info}}}{\text{Split Info}}$, where $\text{Split Info} = -\sum_{i=1}^{k} P(v_i) \log_2 P(v_i)$ and $P(v_i) = (N_i / N)$

Ensemble Learning on Trees

Acknowledgement (Also good reading material)

Alex Ihler
http://sli.ics.uci.edu/Classes/2015W-273a?action=download&upname=09-ensembles.pdf

Cheng Li
http://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf

Chris McCormick
http://mccormickml.com/2013/12/13/adaboost-tutorial/

Overview of ensemble methods on trees (slides):

http://jessica2.msri.org/attachments/10778/10778-boost.pdf

Robust Real-Time Face Detection

http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf

A Gentle Introduction to Gradient Boosting (slides):

http://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf

Ensemble Methods

- Why learn one classifier when you can learn many?
  - "Committee": learn K classifiers, average their predictions
- "Bagging" = bootstrap aggregation
  - Learn many classifiers, each with only part of the data
  - Combine through model averaging
- Remember overfitting: "memorize" the data
  - Used test data to see if we had gone too far
  - Cross-validation
    - Make many splits of the data for train & test
    - Each of these defines a classifier
    - Typically, we use these to check for overfitting
    - Could we instead combine them to produce a better classifier?

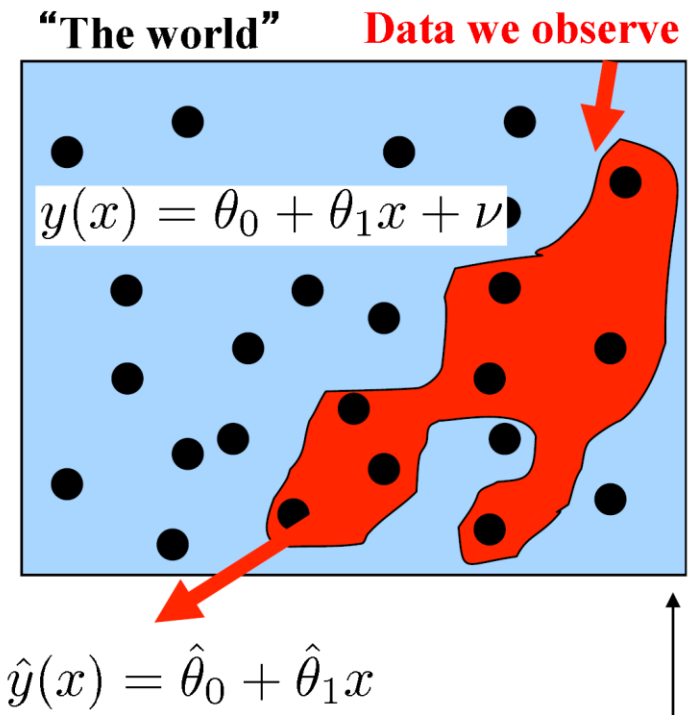
Ensemble Methods

- Committees
  - Regression : Average （prediction结果是numerical / continuous)
  - Classification: Majority Vote (结果是Categorical（特别注意做成numerical 的categorical结果))
- Weighted
  - Up-weight better results.
- Predictor of predictors
  - 把所以classifier得到的结果当做新的Features，用他们再train另一个classifier F_e。
  - 注意，如果ensemble的classifier F_e选择的是linear的classifier，那等于weighted vote.
  - 为了避免overfitting，Ensemble 的classifier F_e最好用在validation data上

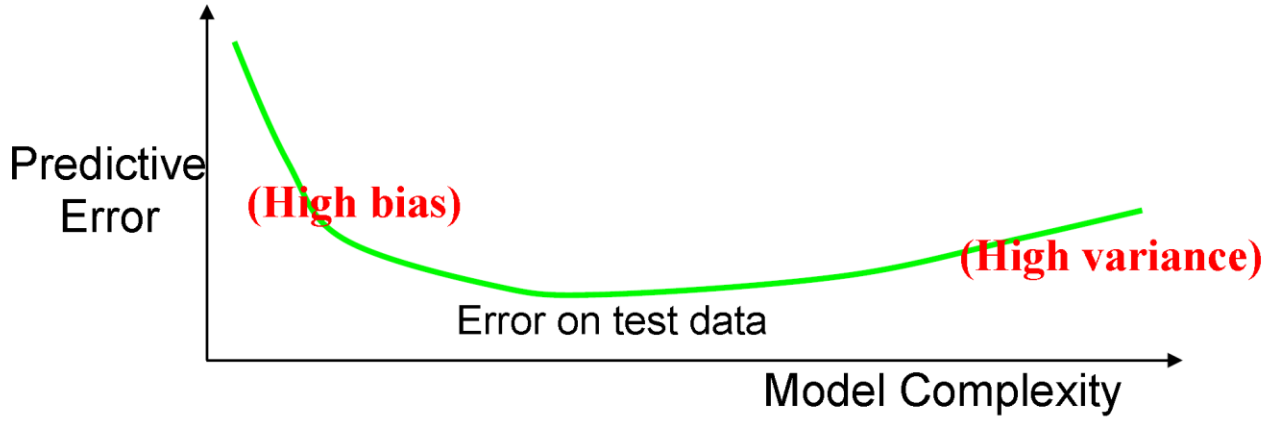
Bagging

- Bootstrap
  - Create a random subset of data by sampling
  - Draw N’ of the N samples with replacement (sometimes w/o)
- Bagging
  - Repeat K times
    - Create a training set of N’ < N examples
    - Train a classifier on the random training set
  - To test, run each trained classifier
    - Each classifier votes on the output, take majority
    - For regression: each regressor predicts, take average
- Notes:
  - Some complexity control: harder for each to memorize data
    - Doesn’t work for linear models (e.g. linear regression)
    - Perceptrons OK (linear + threshold = nonlinear)

Bias/Variance



- We only see a little bit of data
- Can decompose error into two parts
  - Bias – error due to model choice
    - Can our model represent the true best predictor?
    - Gets better with more complexity
  - Variance – randomness due to data size
    - Better w/ more data, worse w/ complexity



Random Forest

- Bagging applied to decision trees
- Problem
  - With lots of data, we usually learn the same classifier
  - Averaging over these doesn't help!
- Introduce extra variation in learner
  - At each step of training, only allow a subset of features
  - Enforces diversity ("best" feature not available)
  - Average over these learners (majority vote)

```
In decisionTreeSplitData2(X,Y):
  For each of a subset of features
    For each possible split
      Score the split  (e.g. information gain)
  Pick the feature & split with the best score
  Recurse on each subset
```

Ensemble

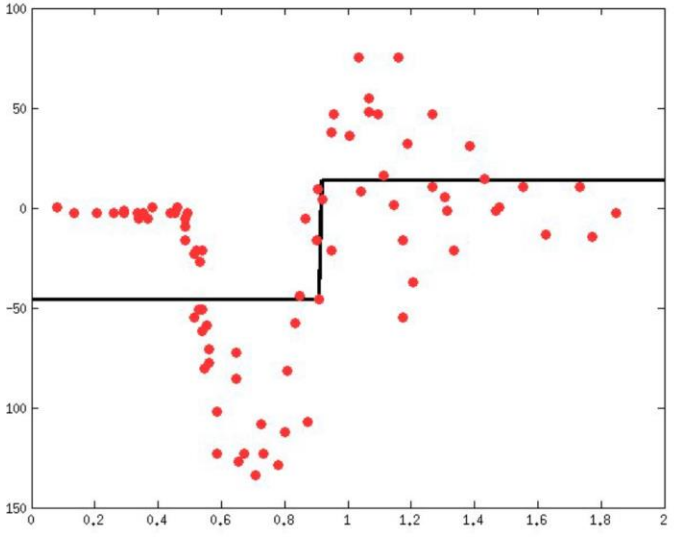
- Weighted combinations of predictors
- “Committee” decisions
  - Trivial example
  - Equal weights (majority vote / unweighted average)
  - Might want to weight unevenly – up-weight better predictors

- Boosting
  - Focus new learners on examples that others get wrong
  - Train learners sequentially
  - Errors of early predictions indicate the “hard” examples
  - Focus later predictions on getting these examples right
  - Combine the whole set in the end
  - Convert many “weak” learners into a complex predictor

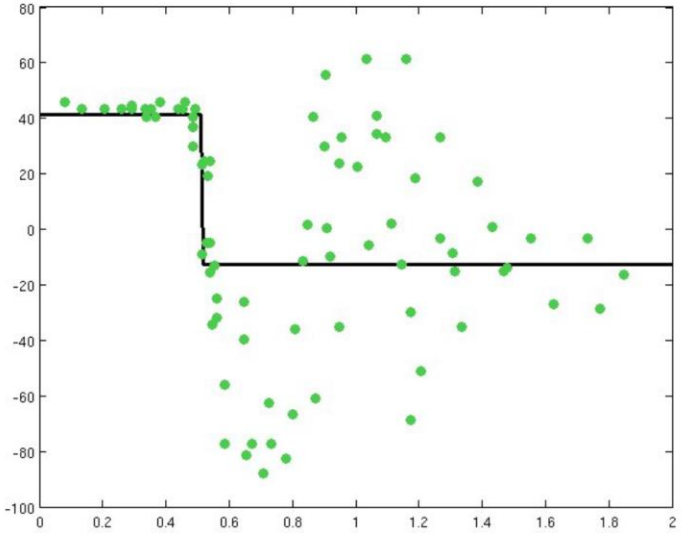
Gradient Boosting

- Learn a regression predictor
- Compute the error residual
- Learn to predict the residual

Learn a simple predictor…

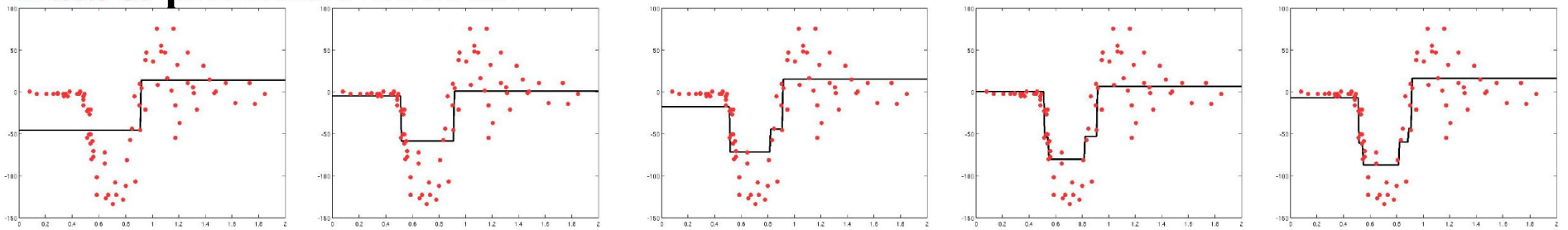


Then try to correct its errors

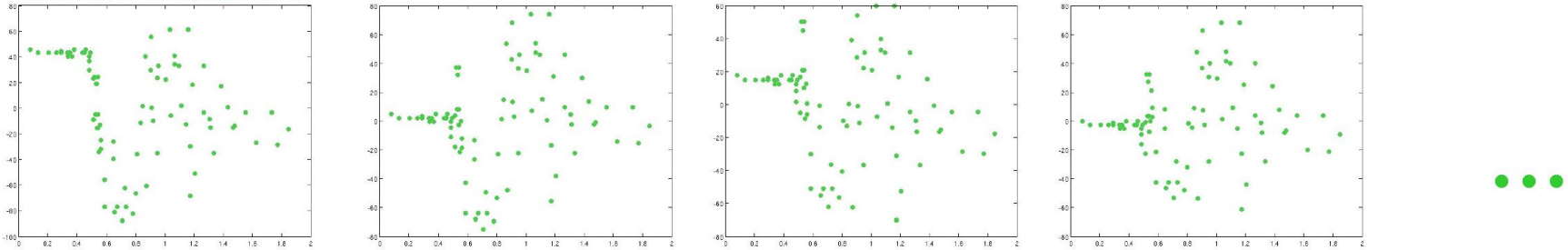


- Learn sequence of predictors
- Sum of predictions is increasingly accurate
- Predictive function is increasingly complex

Data & prediction function



Error residual



A Gradient Descent View

Loss function $L(y, F(x)) = (y - F(x))^2/2$
We want to minimize $J = \sum_i L(y_i, F(x_i))$ by adjusting $F(x_1), F(x_2), ..., F(x_n)$.
Notice that $F(x_1), F(x_2), ..., F(x_n)$ are just some numbers. We can treat $F(x_i)$ as parameters and take derivatives

$$\frac{\partial J}{\partial F(x_i)} = \frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)} = \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = F(x_i) - y_i$$

So we can interpret residuals as negative gradients.

$$y_i - F(x_i) = -\frac{\partial J}{\partial F(x_i)}$$

A Gradient Descent View

$$F(x_i) := F(x_i) + h(x_i)$$
$$F(x_i) := F(x_i) + y_i - F(x_i)$$
$$F(x_i) := F(x_i) - 1\frac{\partial J}{\partial F(x_i)}$$
$$\theta_i := \theta_i - \rho\frac{\partial J}{\partial \theta_i}$$

A Gradient Descent View

For regression with **square loss**,

$$\textit{residual} \Leftrightarrow \textit{negative gradient}$$

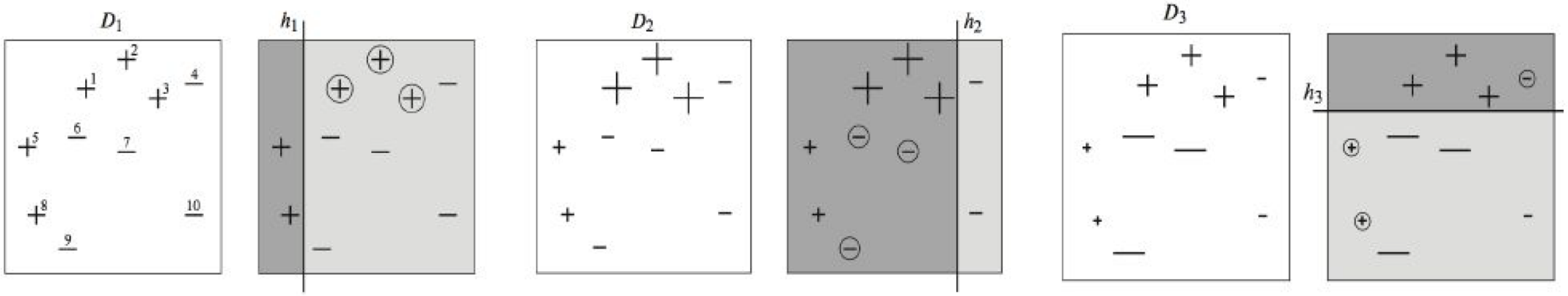
$$\textit{fit } h \textit{ to residual} \Leftrightarrow \textit{fit } h \textit{ to negative gradient}$$

$$\textit{update } F \textit{ based on residual} \Leftrightarrow \textit{update } F \textit{ based on negative gradient}$$

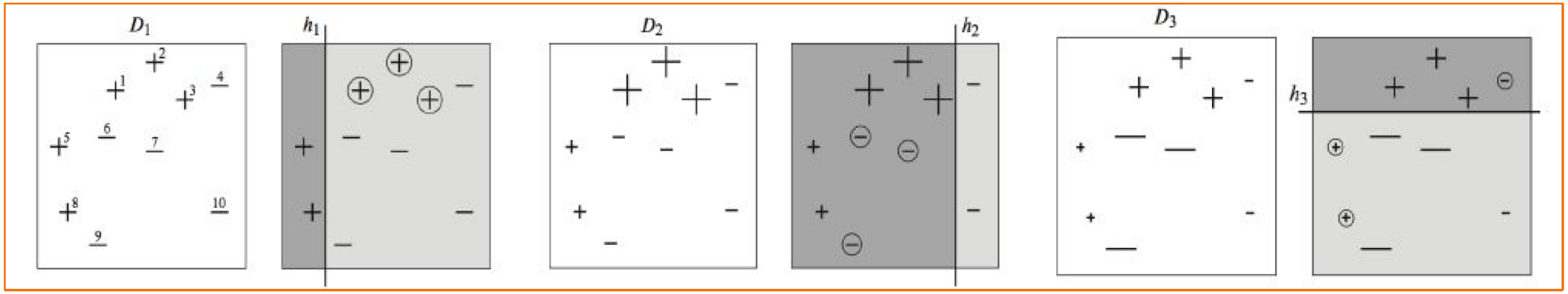
So we are actually updating our model using **gradient descent**!
It turns out that the concept of **gradients** is more general and useful than the concept of **residuals**. So from now on, let's stick with gradients. The reason will be explained later.

AdaBoost

- Short for Adaptive Boosting
- Ensemble many weak classifiers in serial
- Iteratively adjust the weight of each point to minimize the overall ensemble error.



AdaBoost

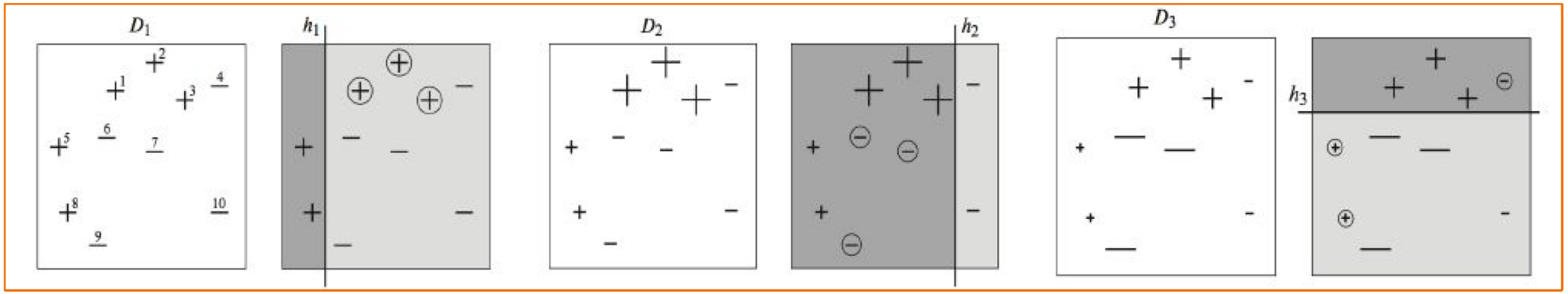


Steps:

1. Initialize the distribution D_1, let $D_1(i) = \frac{1}{N}$
2. Pick a classifier h_t that minimize the error $\epsilon_t = \sum_{\text{wrong } i} D_t(i)$
3. Pick classifier weight α_t
4. Calculate D_{t+1}

   Repeat step 2 - 4
5. The overall Classifier is $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

AdaBoost

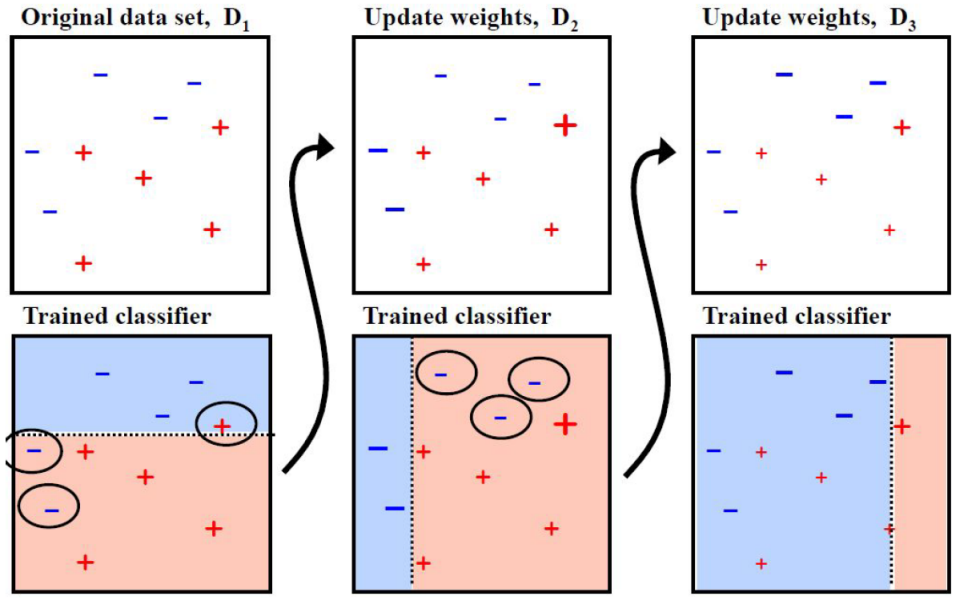


Overall Classifier $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$,

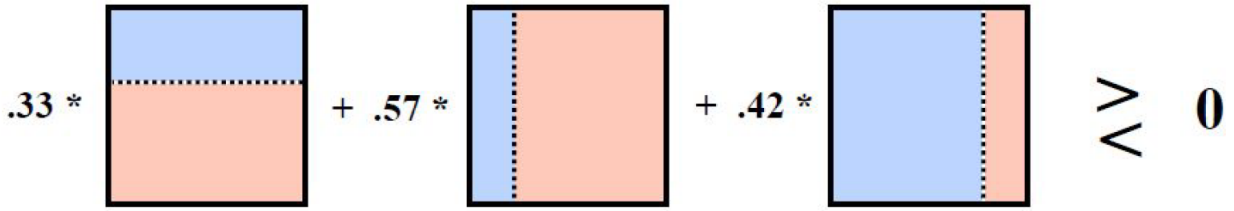
As stated in the original paper (http://rob.schapire.net/papers/explaining-adaboost.pdf)
The weight "distribution" of each point is at time T:

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

AdaBoost: Example

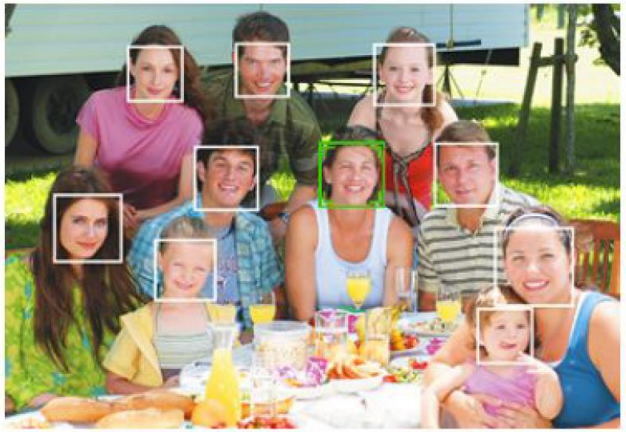


Weight each classifier and combine them:

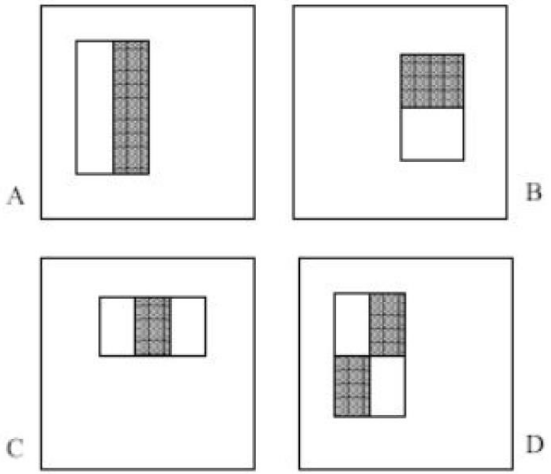


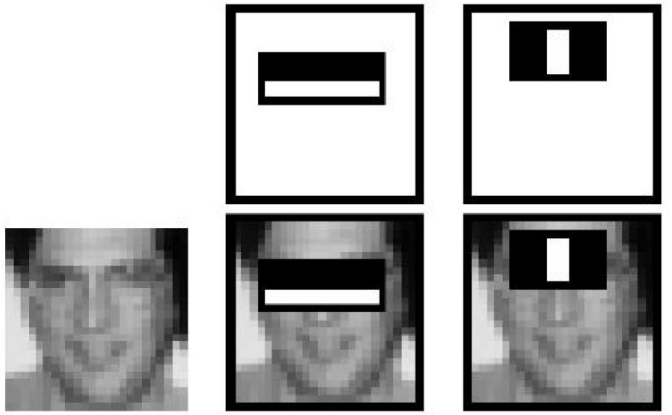
AdaBoost Example: Face Detection

- Viola-Jones face detection algorithm
- Combine lots of very weak classifiers
  - Decision stumps = threshold on a single feature
- Define lots and lots of features
- Use AdaBoost to find good features
  - And weights for combining as well



- Four basic types.
  - They are easy to calculate.
  - The white areas are subtracted from the black ones.
  - A special representation of the sample called the **integral image** makes feature extraction faster.





- Wavelets give ~100k features
- Each feature is one possible classifier
- To train: iterate from 1:T
  - Train a classifier on each feature using weights
  - Choose the best one, find errors and re-weight

- This can take a long time… (lots of classifiers)
  - One way to speed up is to not train very well…
  - Rely on adaboost to fix "even weaker" classifier

- Lots of other tricks in "real" Viola-Jones
  - Cascade of decisions instead of weighted combo
  - Apply at multiple image scales
  - Work to make computationally efficient