CSE 519: Data Science
Steven Skiena
Stony Brook University

Lecture 21: Introduction to Machine Learning

The Machine Learning Zoo

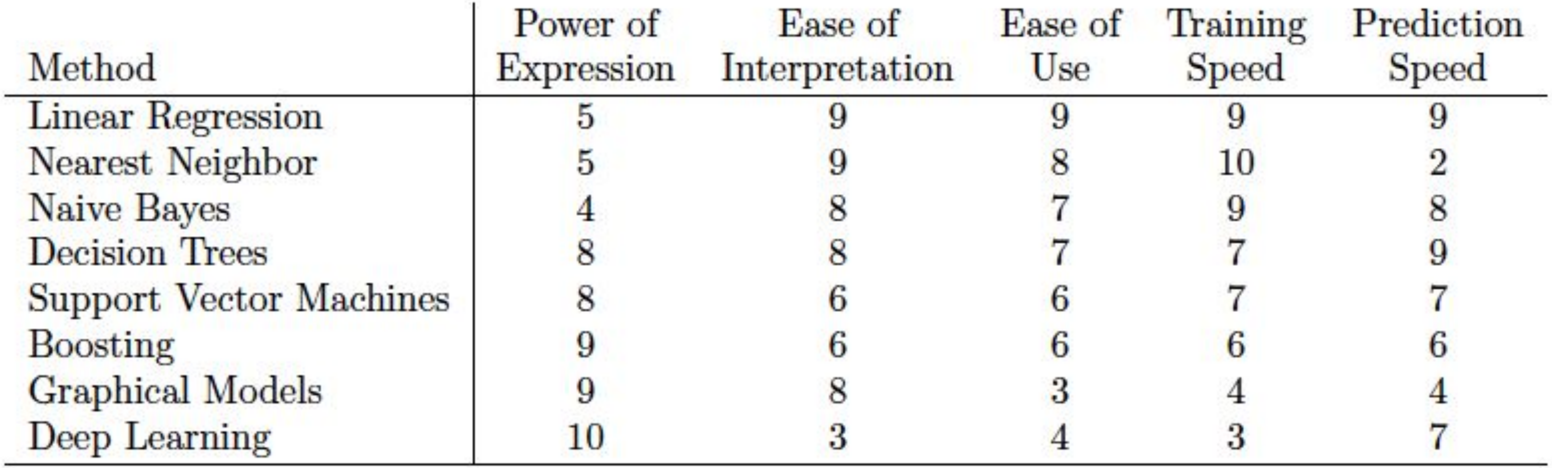
The variety of clever machine learning algorithms makes it easy to lose sight of why?

Evaluation dimensions include:

- Power and expressibility
- Interpretability
- Ease of Use
- Training speed
- Prediction speed

Subjective Rankings

I asked knowledgeable people to compare these ML methods by these dimensions:



Method	Power of Expression	Ease of Interpretation	Ease of Use	Training Speed	Prediction Speed
Linear Regression	5	9	9	9	9
Nearest Neighbor	5	9	8	10	2
Naive Bayes	4	8	7	9	8
Decision Trees	8	8	7	7	9
Support Vector Machines	8	6	6	7	7
Boosting	9	6	6	6	6
Graphical Models	9	8	3	4	4
Deep Learning	10	3	4	3	7

Take Home Lessons

- Linear/logistic regression is pretty good, except in expressibility.
- No single method dominates all the others.
- The comparative advantages of most methods are relatively murky.

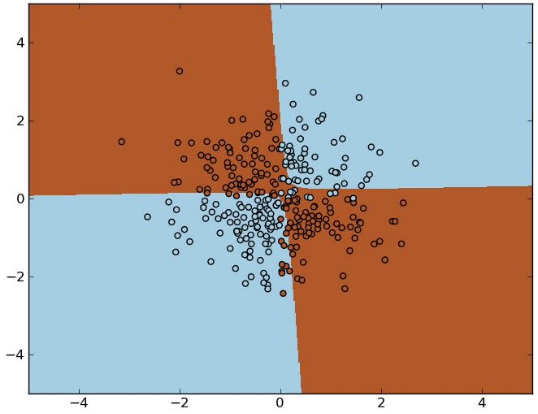
The no free lunch theorem states that no single method works best in all cases.

XOR and Linear Classifiers

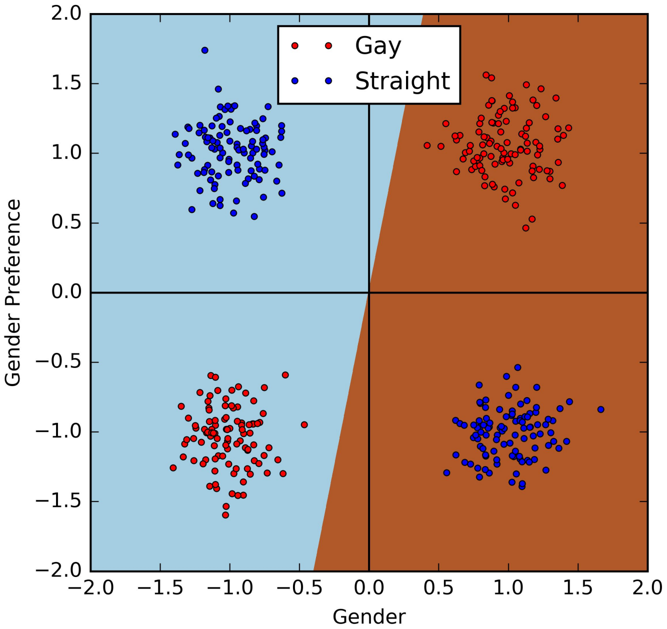
Linear classifiers cannot be used to fit simple non-linear functions like eXclusive OR.

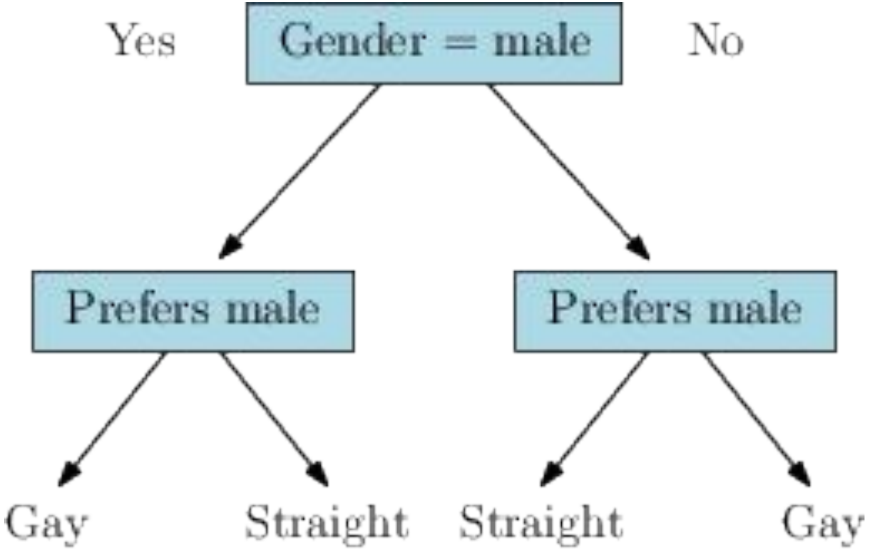
More powerful methods are needed, including:

- Decision trees
- Nearest neighbor methods
- Support vector machines



Gay vs. Straight?





Yes

Gender = male

No

Prefers male

Prefers male

Gay

Straight

Straight

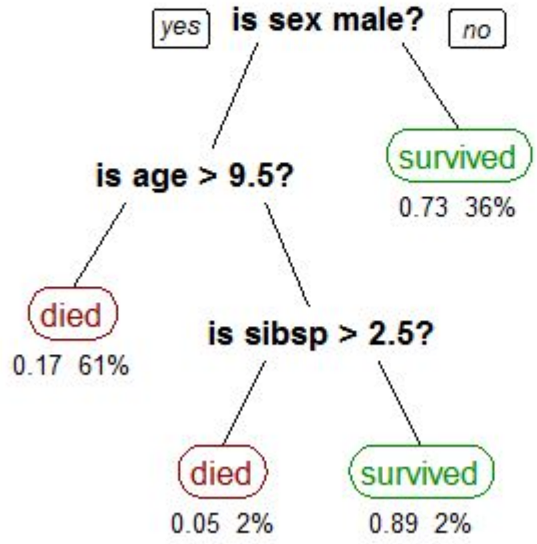
Gay

Decision Tree Classifiers

Each row/instance travels a unique root-to-leaf path to classification.

The tree partitions the training examples into groups of relatively uniform composition, where the decision becomes easy.

Reducing each training example to its own leaf node implies over training.



Class Logic-Based Explanations? (Projects)

- Miss Universe?
- Movie gross?
- Baby weight?
- Art auction price?
- Snow on Christmas?
- Super Bowl / College Champion?
- Ghoul Pool?
- Future Gold / Oil Price?

Advantages of Decision Trees

- Non-linearity
- Support for categorical variables (hair=red)
- Interpretability -- people can read the tree
- Robustness -- we can construct ensembles of different trees and vote (CART)
- Applicability to regression within leaf subset

Biggest disadvantage is lack of elegance/*Math*.

Constructing Decision Trees

Trees are constructed in a top-down manner.

Seek a split along one feature/dimension to purify the information over m classes.

- Pure splits create nodes of one single class
- Balanced splits partition the items into equal-sized groups.

Thus tradeoffs between speed and robustness.

Information-Theoretic Entropy

Entropy measures the amount of class confusion:

$$H(S) = -\sum_{i=1}^{m} f_i \log_2 f_i$$

- Complete disorder means *f_i= 1/m*, so *H(S)=log(m)*.
- Perfect order means *f_1=1* and *f_2=0*, so *H(S)=0*.

Split Criteria

The value of a potential split *S* is how much it reduces the entropy of the system:

- Information gain

$$IG_p(S) = H(S) - \sum_{j=1}^{2} \frac{|S_i|}{|S|} H(S_i)$$

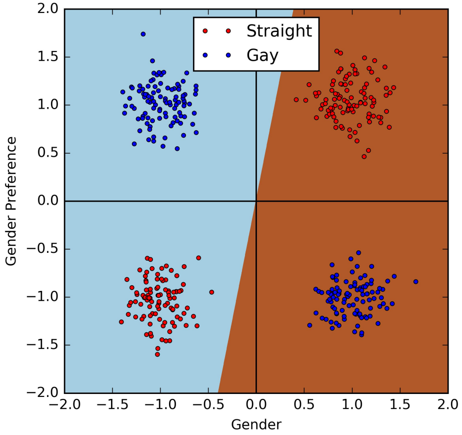
- Gini impurity

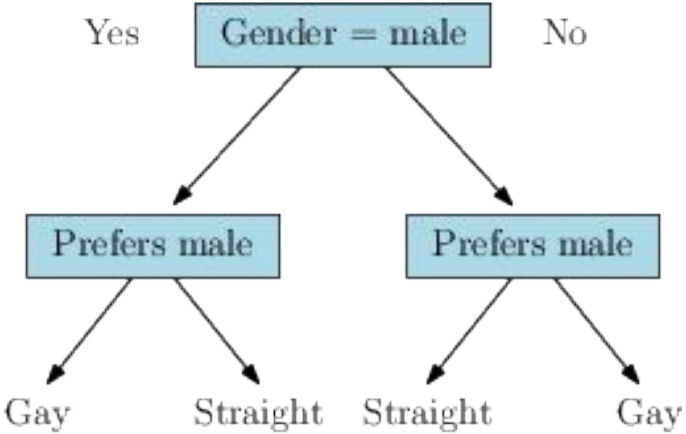
$$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i) = \sum_{i=1}^{m} (f_i - f_i^2)$$

$$= \sum_{i=1}^{m} f_i - \sum_{i=1}^{m} f_i^2 = 1 - \sum_{i=1}^{m} f_i^2$$

Look Ahead

Finding the best XOR tree requires look ahead, as it cannot be found by greedy:





Yes

Gender = male

No

Prefers male

Prefers male

Gay

Straight

Straight

Gay

Stopping Criteria

Building a tree until each leaf is pure (f=1) likely ends with singleton elements, and is overfit.

- Better is to stop when the information gain is small, instead of zero.
- An alternate strategy is to build the full tree, then prune way low value nodes.

Decision tree construction is hacking, not science.

Ensembles of Decision Trees

We can construct hundreds of decision trees by randomly selecting the feature to split on.

- Voting among multiple classifiers increases robustness and let us score our trust level.
- Bagging picks randomly selected subsets of items to train each tree on.

But should all trees get equal votes?

Voting Classifiers

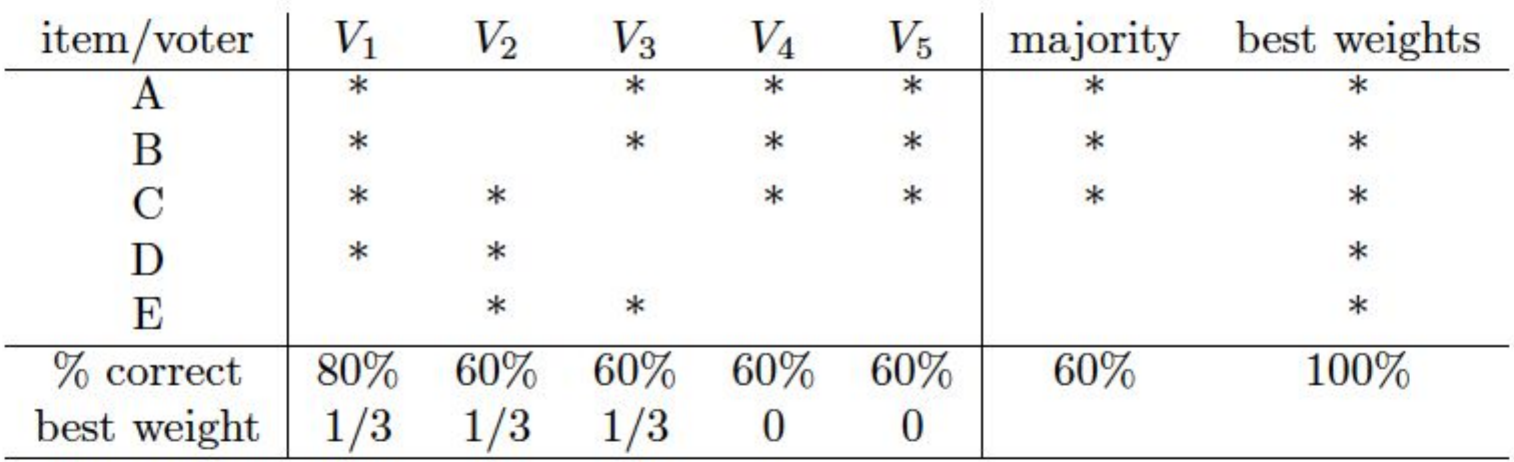
The natural way to use multiple classifiers gives each a vote, and takes the majority label.

Such aggregation is typically done for random sets of decision trees, or even single features.

But should each classifier get the same vote?

Epicurus’ Principle: ``Keep all theories that are consistent with the data’’. But not equal votes..

Weighing Better Classifiers More



item/voter	V_1	V_2	V_3	V_4	V_5	majority	best weights
A	*		*	*	*	*	*
B	*		*	*	*	*	*
C	*	*		*	*	*	*
D	*	*					*
E		*	*				*
% correct	80%	60%	60%	60%	60%	60%	100%
best weight	1/3	1/3	1/3	0	0		

Might weigh by accuracy or linear regression.

But *better* means getting the hard cases right...

Boosting

Boost weak (but >0.5 accuracy) classifiers in a strong classifier.

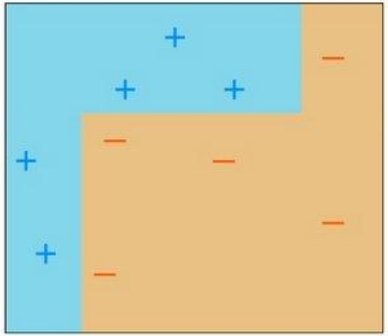
To set the weights of the classifier, we will adjust the weights of the training examples.

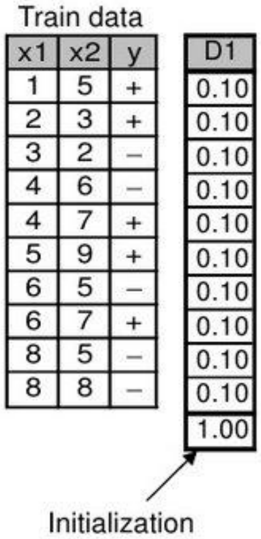
Easy training examples will be properly classified by most classifiers: we reward classifiers more for getting the hard cases right.

Adaboost in Action

We seek non-linear classifiers using thresholded features as classifiers (e.g x>1)

Initially all points are c equal weight.

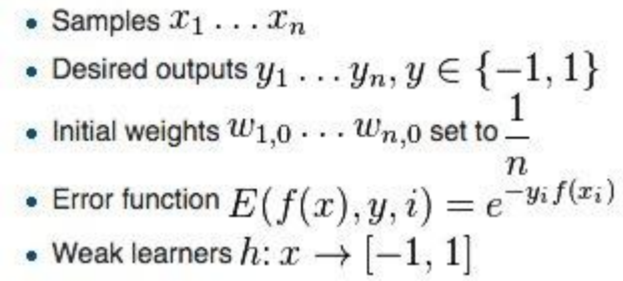




Train data

x1	x2	y	D1
1	5	+	0.10
2	3	+	0.10
3	2	–	0.10
4	6	–	0.10
4	7	+	0.10
5	9	+	0.10
6	5	–	0.10
6	7	+	0.10
8	5	–	0.10
8	8	–	0.10
			1.00

Initialization



- Samples $x_1 \ldots x_n$
- Desired outputs $y_1 \ldots y_n, y \in \{-1, 1\}$
- Initial weights $w_{1,0} \ldots w_{n,0}$ set to $\frac{1}{n}$
- Error function $E(f(x), y, i) = e^{-y_i f(x_i)}$
- Weak learners $h: x \to [-1, 1]$

Boosting Misclassified Points

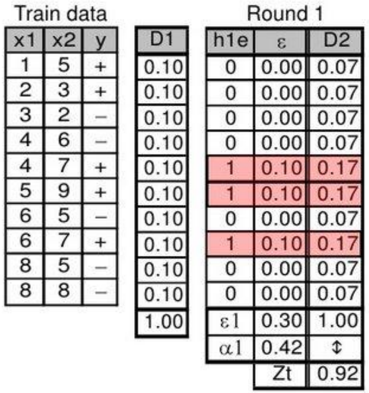
The weight of the misclassified points is boosted to make them more important in the next round.

$$w'_{i,t+1} = w_{i,t}e^{-y_i\alpha_t h_t(x_i)}$$

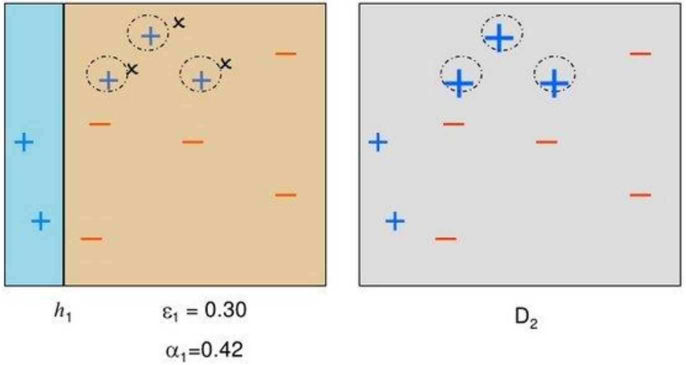
Normalize total weights to sum to 1.

The weight of the new classifier depends upon how accurate it is:

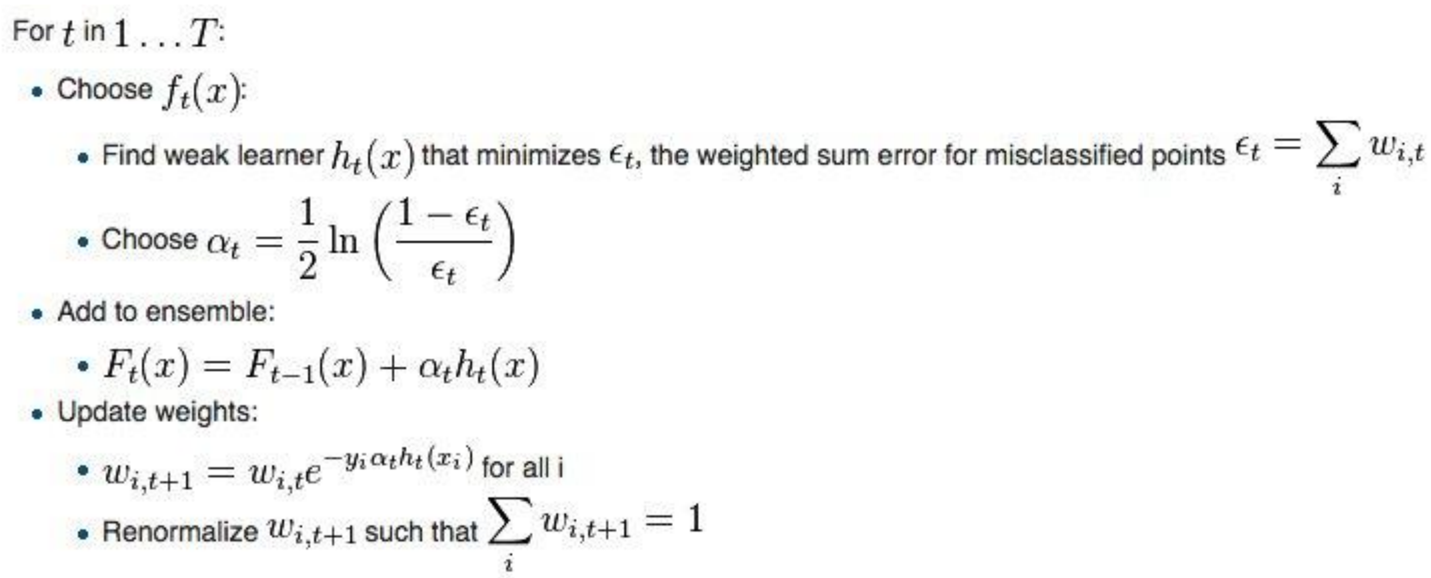
$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$



Train data				Round 1		
x1	x2	y	D1	h1e	ε	D2
1	5	+	0.10	0	0.00	0.07
2	3	+	0.10	0	0.00	0.07
3	2	–	0.10	0	0.00	0.07
4	6	–	0.10	0	0.00	0.07
4	7	+	0.10	1	0.10	0.17
5	9	+	0.10	1	0.10	0.17
6	5	–	0.10	0	0.00	0.07
6	7	+	0.10	1	0.10	0.17
8	5	–	0.10	0	0.00	0.07
8	8	–	0.10	0	0.00	0.07
			1.00	ε1	0.30	1.00
				α1	0.42	↕
					Zt	0.92



AdaBoost Algorithm

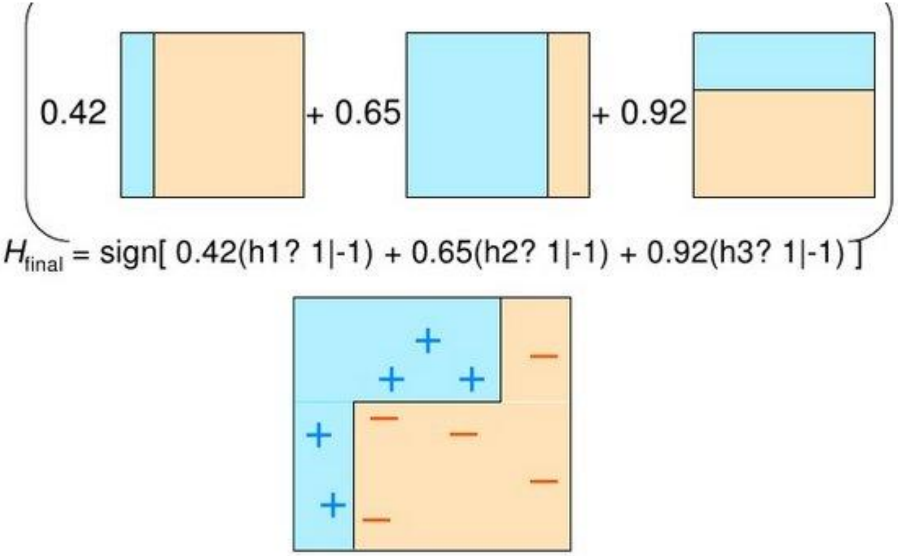


For t in $1 \ldots T$:

- Choose $f_t(x)$:
    - Find weak learner $h_t(x)$ that minimizes ϵ_t, the weighted sum error for misclassified points $\epsilon_t = \sum_i w_{i,t}$
    - Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$
- Add to ensemble:
    - $F_t(x) = F_{t-1}(x) + \alpha_t h_t(x)$
- Update weights:
    - $w_{i,t+1} = w_{i,t} e^{-y_i \alpha_t h_t(x_i)}$ for all i
    - Renormalize $w_{i,t+1}$ such that $\sum_i w_{i,t+1} = 1$

Final Classifier

This weighted ensemble correctly classifies all points.


0.42
+ 0.65
+ 0.92
H_final = sign[0.42(h1? 1|-1) + 0.65(h2? 1|-1) + 0.92(h3? 1|-1)]
+
−

Boosting: Pro and Con

Boosting provides a way to take advantage of weak classifiers (small correlation features) in an effective way.

Boosting tries to fit every example, thus it will overfit noisy data. “Hard cases make bad law.”

Gradient boosted decision trees are the most common method in Kaggle competitions.