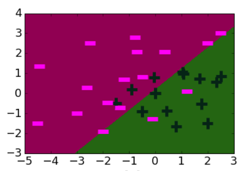
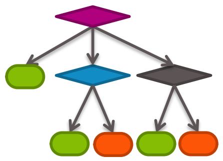
Boosting

CS229: Machine Learning
Carlos Guestrin
Stanford University
Slides include content developed by and co-developed with Emily Fox

Simple (weak) classifiers are good!



Logistic regression
w. simple features



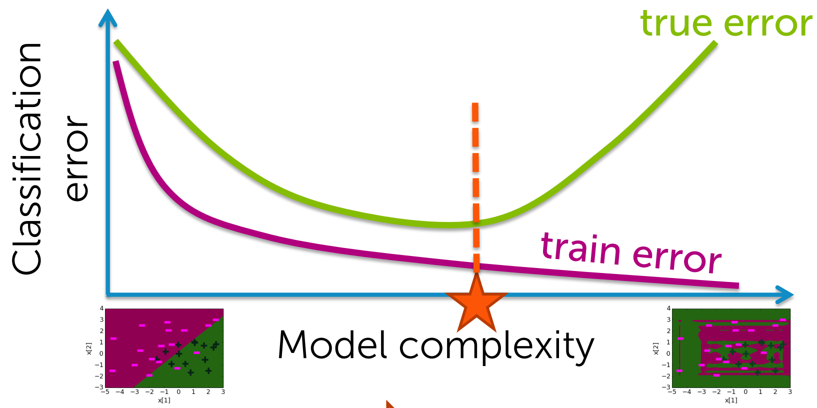
Shallow
decision trees

Decision
stumps

Low variance. Learning is fast!

But high bias…

Finding a classifier that's just right



Weak learner → Need stronger learner

Option 1: add more features or depth
Option 2: ?????

Boosting question

"Can a set of weak learners be combined to create a stronger learner?" *Kearns and Valiant (1988)*

↓

Yes! *Schapire (1990)*

↓

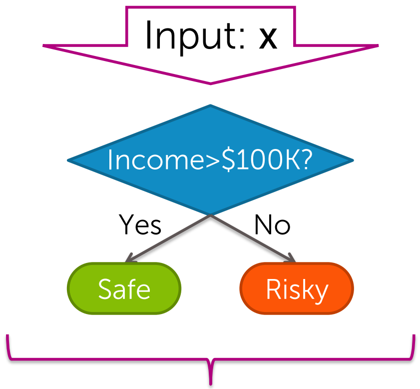
Boosting

↓

Amazing impact: • simple approach • widely used in industry • wins most Kaggle competitions • great systems (e.g., XGBoost)

Ensemble classifier

A single classifier



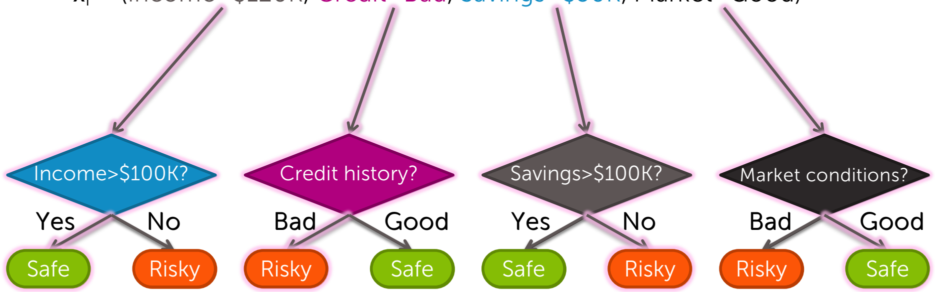
Output: $\hat{y} = f(\mathbf{x})$

– Either +1 or -1

Classifier

Ensemble methods: Each classifier "votes" on prediction

x_i = (Income=\$120K, Credit=Bad, Savings=\$50K, Market=Good)



$f_1(x_i) = +1$ $\quad f_2(x_i) = -1$ $\quad f_3(x_i) = -1$ $\quad f_4(x_i) = +1$

Combine?

Ensemble model

Learn coefficients

$$F(x_i) = \text{sign}(w_1 f_1(x_i) + w_2 f_2(x_i) + w_3 f_3(x_i) + w_4 f_4(x_i))$$

Ensemble classifier in general

- Goal:
  - Predict output y
    - Either +1 or -1
  - From input **x**
- Learn ensemble model:
  - Classifiers: $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_T(\mathbf{x})$
  - Coefficients: $\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_T$
- Prediction:

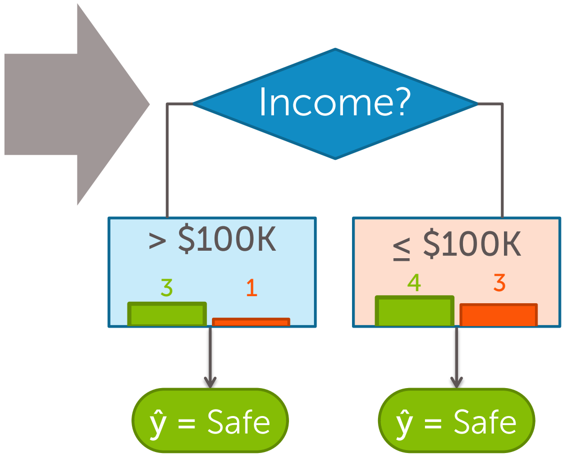
$$\hat{y} = sign\left(\sum_{t=1}^{T} \hat{\mathbf{w}}_t f_t(\mathbf{x})\right)$$

Boosting

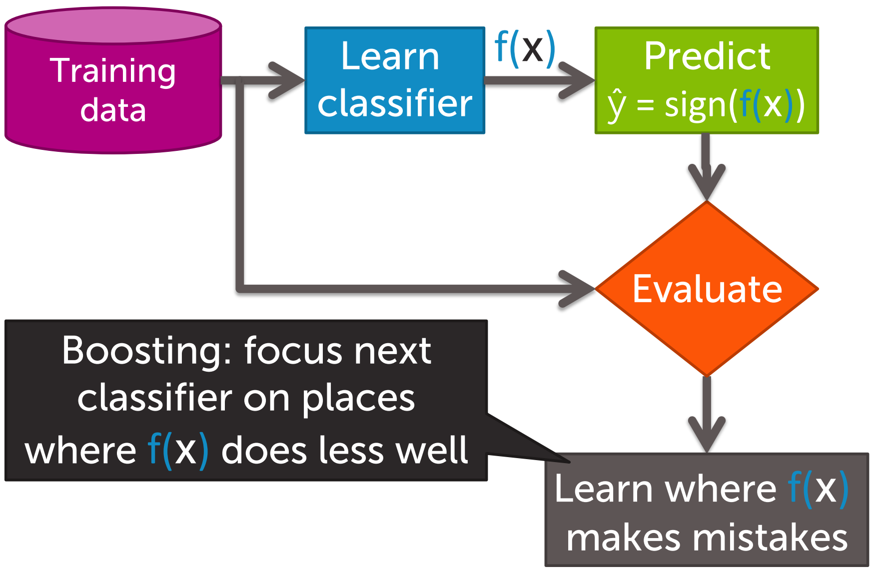
Training a classifier

Learning decision stump

Credit	Income	y
A	$130K	Safe
B	$80K	Risky
C	$110K	Risky
A	$110K	Safe
A	$90K	Safe
B	$120K	Safe
C	$30K	Risky
C	$60K	Risky
B	$95K	Safe
A	$60K	Safe
A	$98K	Safe



Boosting = Focus learning on "hard" points



Learning on weighted data:
More weight on "hard" or more important points

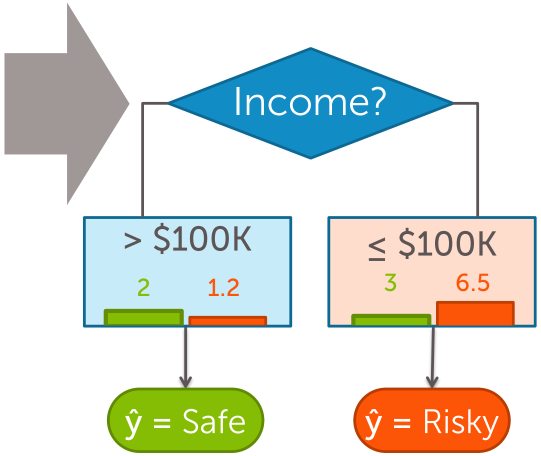
- Weighted dataset:
  - Each $\mathbf{x}_i, y_i$ weighted by α_i
    - More important point = higher weight α_i

- Learning:
  - Data point i counts as α_i data points
    - E.g., $\alpha_i = 2$ ➔ count point twice

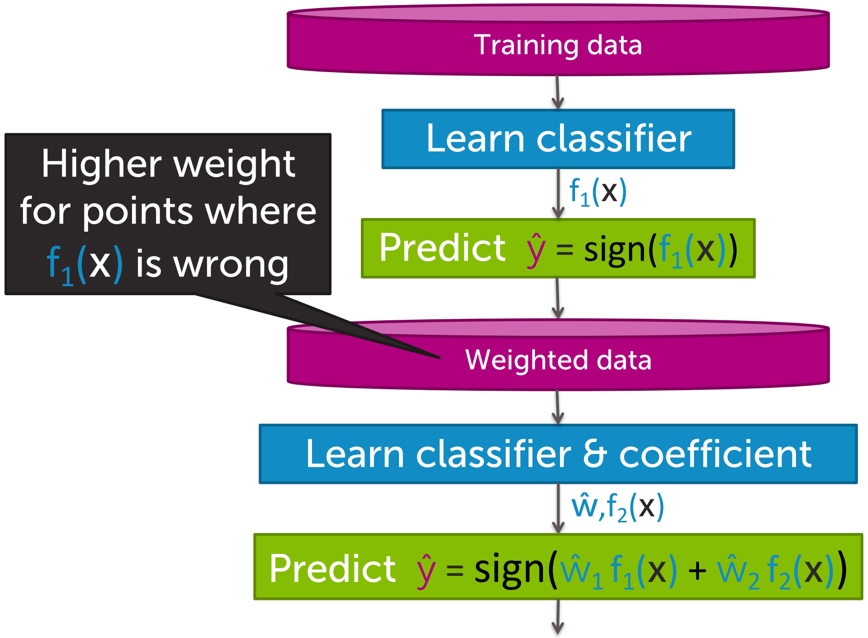
Learning a decision stump on weighted data

Increase weight α of harder/misclassified points

Credit	Income	y	Weight α
A	$130K	Safe	0.5
B	$80K	Risky	1.5
C	$110K	Risky	1.2
A	$110K	Safe	0.8
A	$90K	Safe	0.6
B	$120K	Safe	0.7
C	$30K	Risky	3
C	$60K	Risky	2
B	$95K	Safe	0.8
A	$60K	Safe	0.7
A	$98K	Safe	0.9



Boosting = Greedy learning ensembles from data



AdaBoost algorithm

AdaBoost: learning ensemble

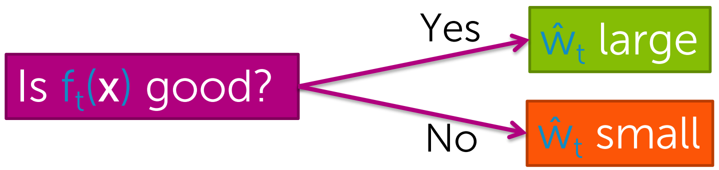
[Freund & Schapire 1999]

- Start with same weight for all points: $\alpha_i = 1/N$
- For t = 1,…,T
  - Learn $f_t(\mathbf{x})$ with data weights α_i
  - Compute coefficient $\hat{w}_t$
  - Recompute weights α_i
- Final model predicts by:

$$\hat{y} = sign\left(\sum_{t=1}^{T} \hat{\mathbf{w}}_t f_t(\mathbf{x})\right)$$

Computing coefficient $\hat{w}_t$

AdaBoost: Computing coefficient $\hat{w}_t$ of classifier $f_t(x)$



- $f_t(\mathbf{x})$ is good ➔ f_t has low training error
- Measuring error in weighted data?
  – Just weighted # of misclassified points

AdaBoost: Formula for computing coefficient $\hat{w}_t$ of classifier $f_t(x)$

$$\hat{\mathbf{w}}_t = \frac{1}{2} \ln \left(\frac{1 - weighted\_error(f_t)}{weighted\_error(f_t)} \right)$$

Is $f_t(\mathbf{x})$ good?

Yes

No

weighted_error(f_t) on training data	$\frac{1 - weighted\_error(f_t)}{weighted\_error(f_t)}$	$\hat{w}_t$

AdaBoost: learning ensemble

- Start with same weight for all points: $\alpha_i = 1/N$
- For t = 1,…,T
  - Learn $f_t(\mathbf{x})$ with data weights α_i
  - Compute coefficient $\hat{w}_t$
  - Recompute weights α_i

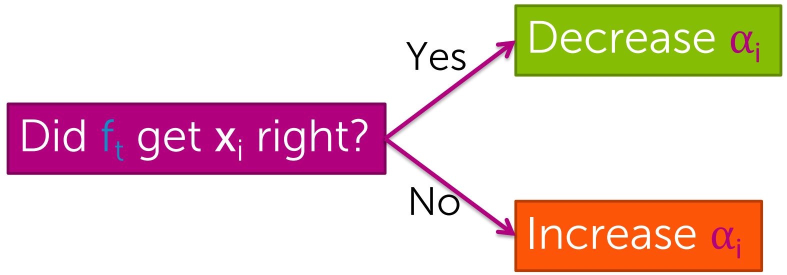
$$\hat{\mathbf{w}}_t = \frac{1}{2} \ln \left(\frac{1 - weighted\_error(f_t)}{weighted\_error(f_t)} \right)$$

- Final model predicts by:

$$\hat{y} = sign\left(\sum_{t=1}^{T} \hat{\mathbf{w}}_t f_t(\mathbf{x}) \right)$$

Recompute weights α_i

AdaBoost: Updating weights α_i based on where classifier $f_t(x)$ makes mistakes



AdaBoost: Formula for updating weights α_i

$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t}, & \text{if } f_t(x_i)=y_i \\ \alpha_i e^{\hat{w}_t}, & \text{if } f_t(x_i)\neq y_i \end{cases}$$

$f_t(x_i)=y_i$?	$\hat{w}_t$	**Multiply α_i by**	**Implication**

Did f_t get $\mathbf{x}_i$ right?

Yes

No

AdaBoost: learning ensemble

- Start with same weight for all points: $\alpha_i = 1/N$
- For t = 1,…,T
  - Learn $f_t(\mathbf{x})$ with data weights α_i
  - $\boxed{\text{Compute coefficient } \hat{w}_t}$
  - $\boxed{\text{Recompute weights } \alpha_i}$
- Final model predicts by:

$$\hat{y} = sign\left(\sum_{t=1}^{T} \hat{\mathbf{w}}_t f_t(\mathbf{x})\right)$$

$$\boxed{\hat{\mathbf{w}}_t = \frac{1}{2} \ln\left(\frac{1 - weighted\_error(f_t)}{weighted\_error(f_t)}\right)}$$

$$\alpha_i \leftarrow \begin{cases} \alpha_i \, e^{-\hat{w}_t}, & \text{if } f_t(x_i) = y_i \\ \alpha_i \, e^{\hat{w}_t}, & \text{if } f_t(x_i) \neq y_i \end{cases}$$

AdaBoost: Normalizing weights α_i

If $\mathbf{x}_i$ often mistake, weight α_i gets very **large**

If $\mathbf{x}_i$ often correct, weight α_i gets very **small**

Can cause numerical instability after many iterations

Normalize weights to add up to 1 after every iteration

$$\alpha_i \leftarrow \frac{\alpha_i}{\sum_{j=1}^{N} \alpha_j}$$

AdaBoost: learning ensemble

- Start with same weight for all points: $\alpha_i = 1/N$
- For t = 1,…,T
  - Learn $f_t(\mathbf{x})$ with data weights α_i
  - Compute coefficient $\hat{w}_t$
  - Recompute weights α_i
  - Normalize weights α_i
- Final model predicts by:

$$\hat{y} = sign\left(\sum_{t=1}^{T} \hat{\mathbf{w}}_t f_t(\mathbf{x})\right)$$

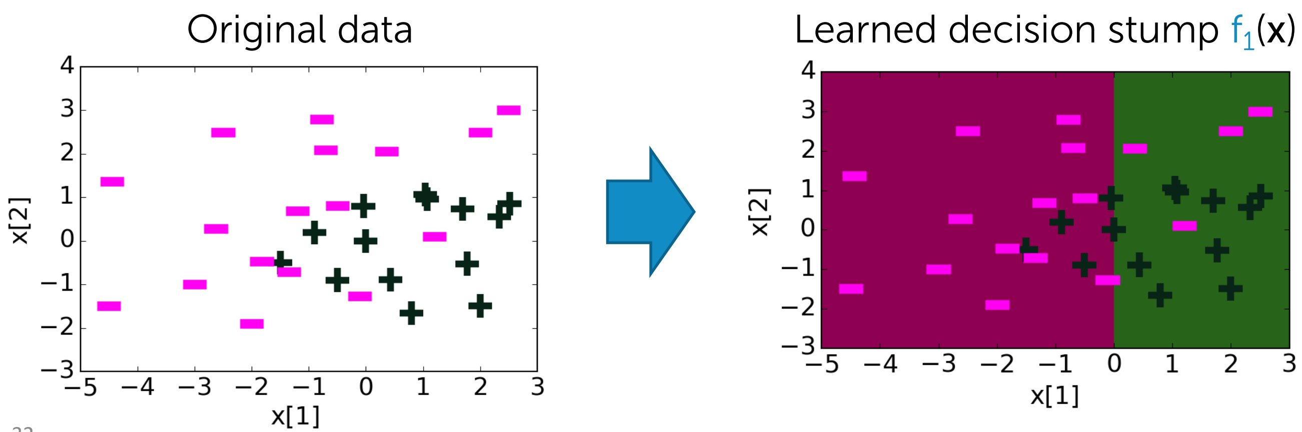
$$\hat{\mathbf{w}}_t = \frac{1}{2}\ln\left(\frac{1 - weighted\_error(f_t)}{weighted\_error(f_t)}\right)$$

$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i) = y_i \\ \alpha_i e^{\hat{w}_t}, & \text{if } f_t(\mathbf{x}_i) \neq y_i \end{cases}$$

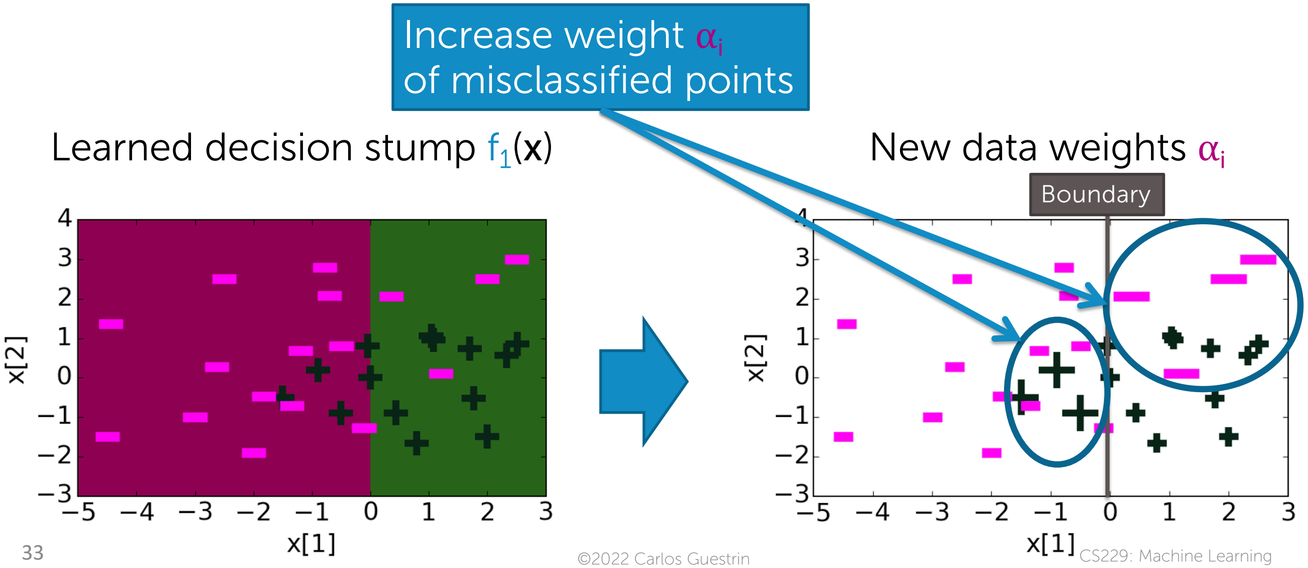
$$\alpha_i \leftarrow \frac{\alpha_i}{\sum_{j=1}^{N} \alpha_j}$$

AdaBoost example: A visualization

t=1: Just learn a classifier on original data

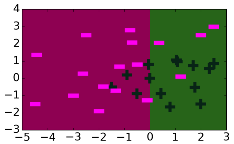


Updating weights α_i

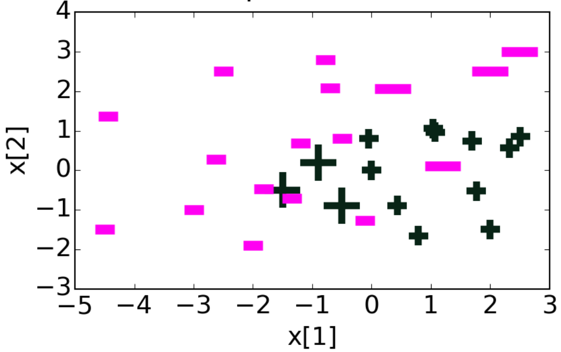


t=2: Learn classifier on weighted data

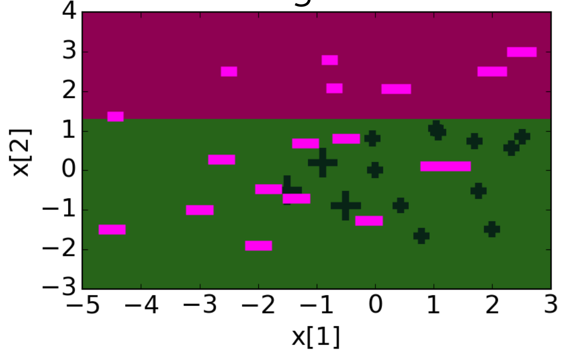
$f_1(\mathbf{x})$



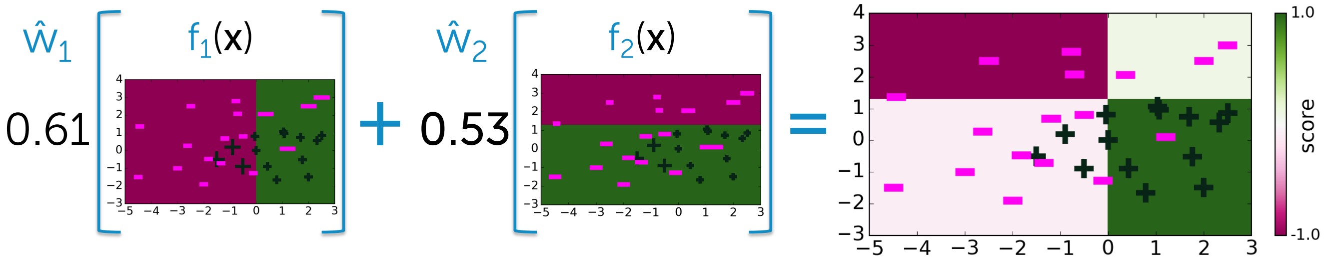
Weighted data: using α_i chosen in previous iteration



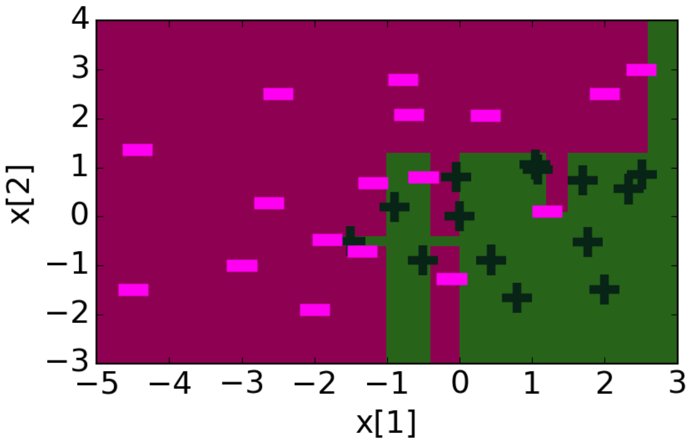
Learned decision stump $f_2(\mathbf{x})$ on weighted data



Ensemble becomes weighted sum of learned classifiers



Decision boundary of ensemble classifier after 30 iterations



training_error = 0

Boosting convergence & overfitting

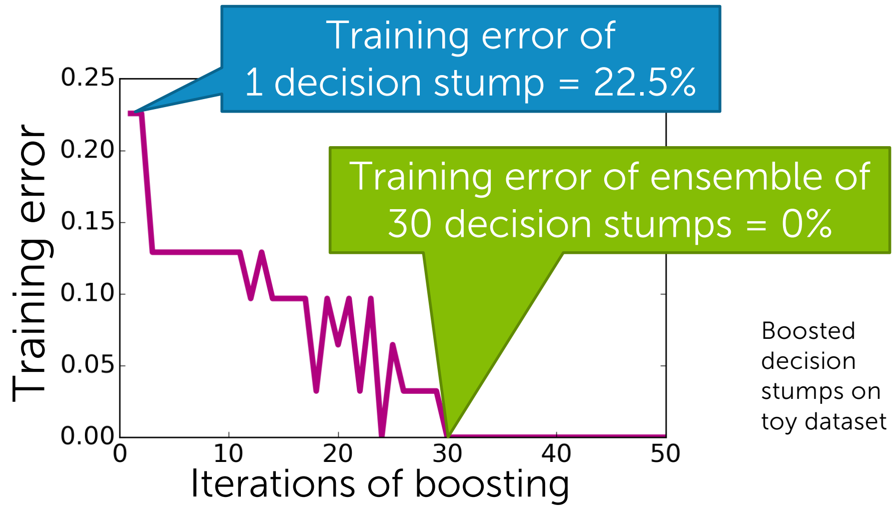
Boosting question revisited

"Can a set of weak learners be combined to create a stronger learner?" *Kearns and Valiant (1988)*

Yes! *Schapire (1990)*

Boosting

After some iterations, training error of boosting goes to zero!!!

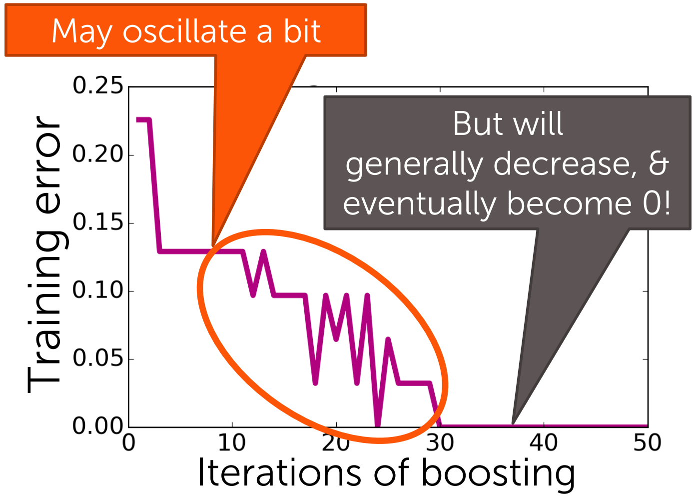


Boosted decision stumps on toy dataset

AdaBoost Theorem

Under some technical conditions…

Training error of boosted classifier → 0 as T→∞



Condition of AdaBoost Theorem

Under some technical conditions…

Training error of boosted classifier → 0 as T→∞

Condition = At every t, can find a weak learner with weighted_error(f_t) < 0.5

Not always possible

Extreme example: No classifier can separate a +1 on top of -1

Nonetheless, boosting often yields great training error

AdaBoost Theorem more formally

Training error of final classifier is bounded by:

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[F(x_i) \neq y_i] \leq \frac{1}{N}\sum_{i=1}^{N}\exp(-y_i\text{score}(x_i))$$

Where $\text{score}(x) = \sum_t \hat{w}_t f_t(x);\ F(x) = sign(\text{score}(x))$

AdaBoost Theorem more formally

Training error of final classifier is bounded by:

$$Z_t = \sum_{i=1}^{N} \alpha_{i,t} \exp(-\hat{w}_t y_i f_t(x_i))$$

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[F(x_i) \neq y_i] \leq \frac{1}{N}\sum_{i=1}^{N} \exp(-y_i \text{score}(x_i)) = \prod_{t=1}^{T} Z_t$$

$$\alpha_i \leftarrow \frac{\alpha_i}{\sum_{j=1}^{N} \alpha_j}$$

Where $\text{score}(x) = \sum_t \hat{w}_t f_t(x);\ F(x) = sign(\text{score}(x))$

AdaBoost Theorem more formally

If we minimize $\prod_{t=1}^{T} Z_t$, we minimize our training error

We can tighten this bound greedily by choosing $\hat{w}_t$, f_t on each iteration to minimize:

$$Z_t = \sum_{i=1}^{N} \alpha_{i,t} \exp(-\hat{w}_t y_i f_t(x_i))$$

For boolean target function, this is accomplished by [Freund & Schapire '97]:

$$\hat{w}_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$$

AdaBoost Theorem more formally

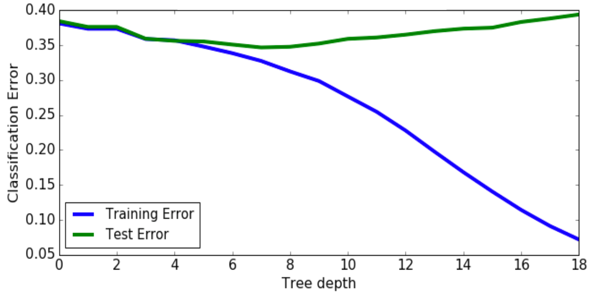
If each classifier is (at least slightly) better than random

$$weighted\_error(f_t) = \epsilon_t < 0.5$$

AdaBoost will achieve zero training error (exponentially fast):

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[F(x_i) \neq y_i] \leq \prod_{t=1}^{T} Z_t \leq \exp\left(-2\sum_{t=1}^{T}(1/2-\epsilon_t)^2\right)$$

Decision trees on loan data

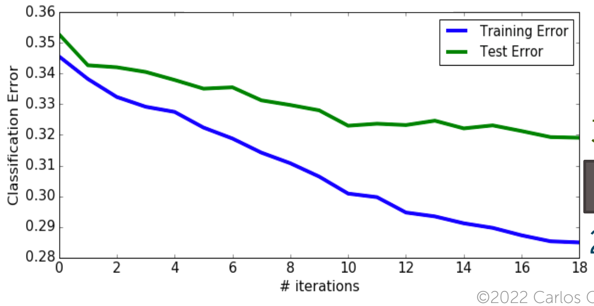


39% test error

Overfitting

8% training error

Boosted decision stumps on loan data

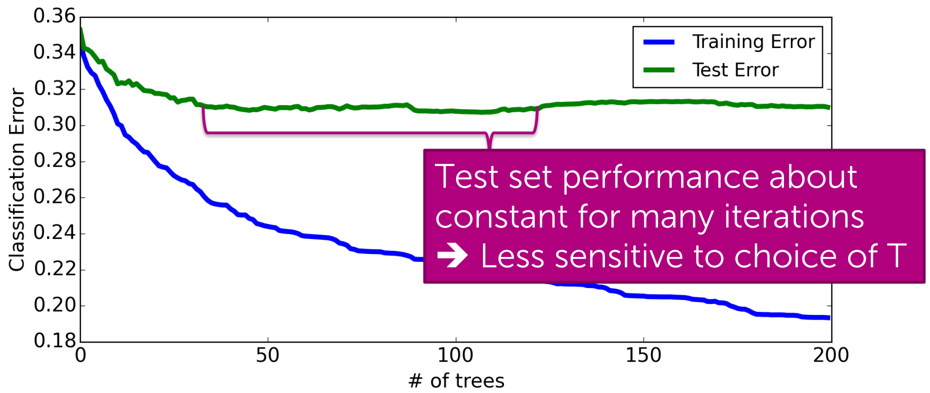


32% test error

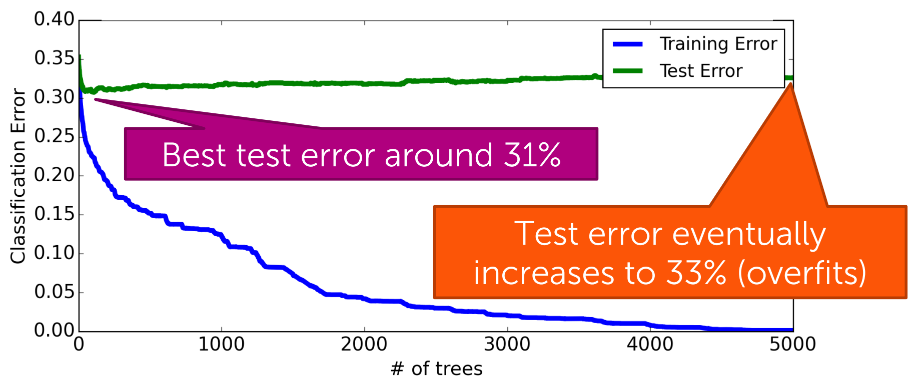
Better fit & lower test error

28.5% training error

Boosting tends to be robust to overfitting



But boosting will eventually overfit, so must choose max number of components T



Summary of boosting

Variants of boosting and related algorithms

There are hundreds of variants of boosting, most important:

Gradient boosting

- Like AdaBoost, but useful beyond basic classification
- Great implementations available (e.g., XGBoost)

Many other approaches to learn ensembles, most important:

Random forests

- **Bagging**: Pick random subsets of the data
  - Learn a tree in each subset
  - Average predictions
- Simpler than boosting & easier to parallelize
- Typically higher error than boosting for same # of trees (# iterations T)

Impact of boosting (*spoiler alert... HUGE IMPACT*)

Amongst most useful ML methods ever created

Extremely useful in computer vision	• Standard approach for face detection, for example
Used by **most winners** of ML competitions (Kaggle, KDD Cup,...)	• Malware classification, credit fraud detection, ads click through rate estimation, sales forecasting, ranking webpages for search, Higgs boson detection,...
Most deployed ML systems use model ensembles	• Coefficients chosen manually, with boosting, with bagging, or others

What you can do now…

- Identify notion ensemble classifiers
- Formalize ensembles as weighted combination of simpler classifiers
- Outline the boosting framework – sequentially learn classifiers on weighted data
- Describe the AdaBoost algorithm
  - Learn each classifier on weighted data
  - Compute coefficient of classifier
  - Recompute data weights
  - Normalize weights
- Implement AdaBoost to create an ensemble of decision stumps