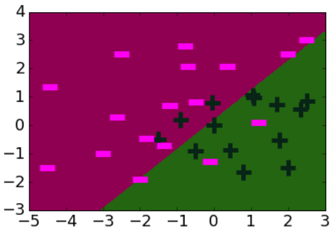
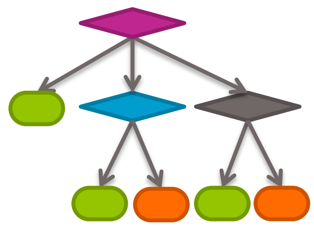
Boosting

Emily Fox & Carlos Guestrin
Machine Learning Specialization
University of Washington

Simple (weak) classifiers are good!



Logistic regression w. simple features



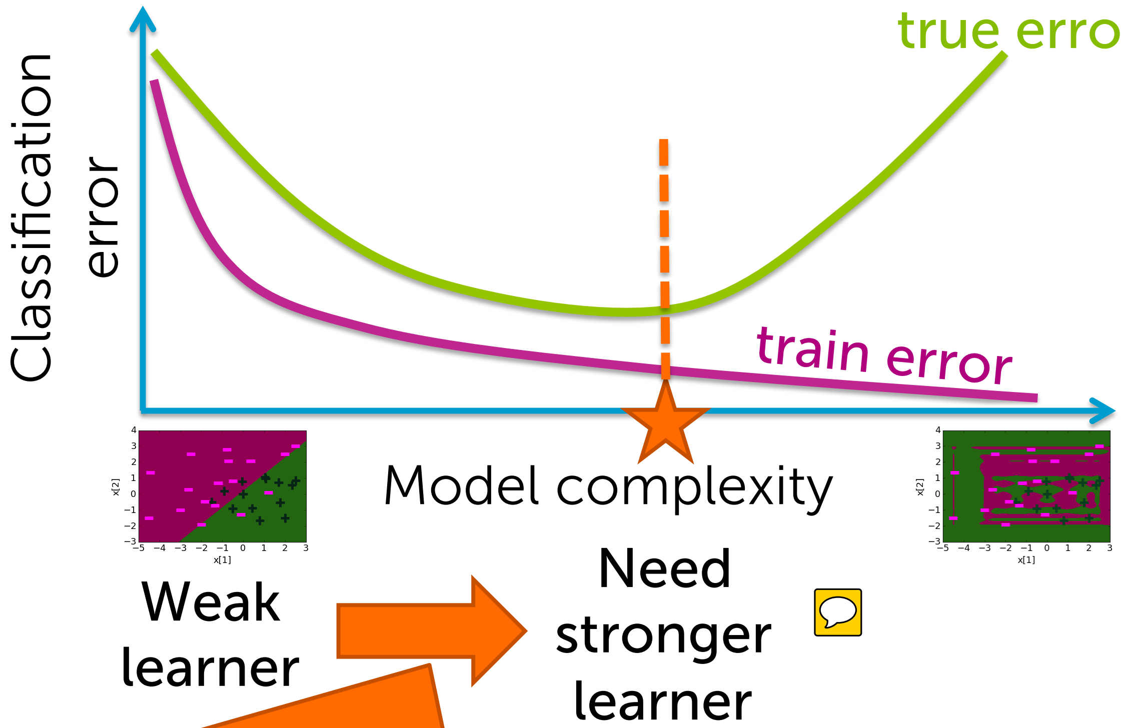
Shallow decision trees

Decision stumps

Low variance. Learning is fast!

But high bias…

Finding a classifier that's just right



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)*

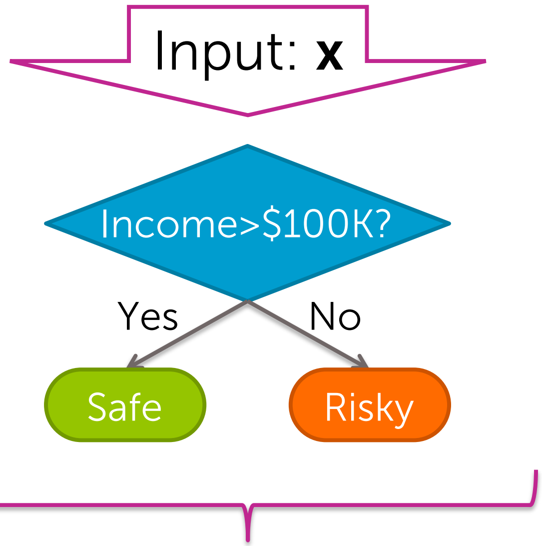
Yes! *Schapire (1990)*

Boosting

Amazing impact: • simple approach • widely used in industry • wins most Kaggle competitions

Ensemble classifier

A single classifier



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

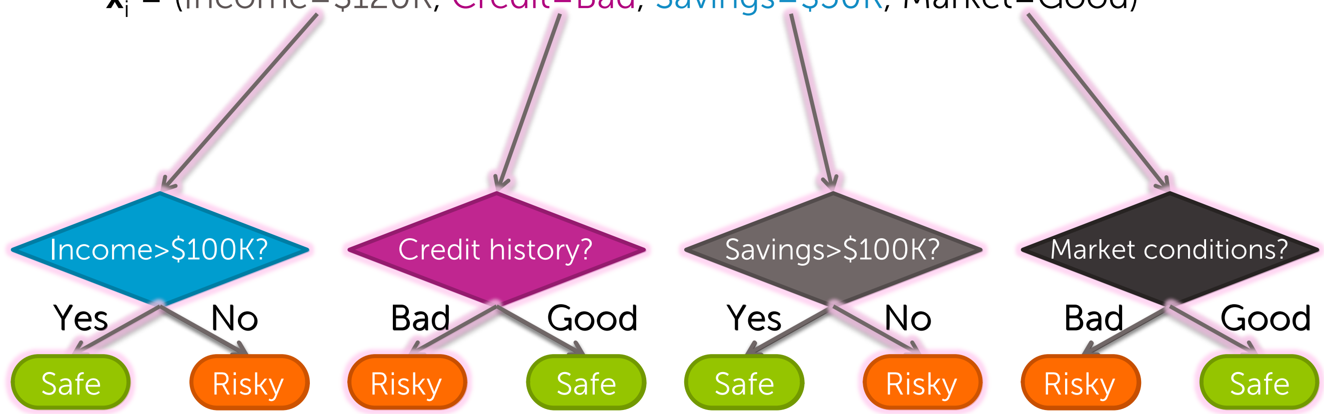
– Either +1 or -1

Classifier


Income>$100K?
Yes
No
Safe
Risky

Ensemble methods: Each classifier "votes" on prediction

$\mathbf{x}_i$ = (Income=\$120K, Credit=Bad, Savings=\$50K, Market=Good)



$f_1(\mathbf{x}_i) = +1$ $\quad f_2(\mathbf{x}_i) = -1$ $\quad f_3(\mathbf{x}_i) = -1$ $\quad f_4(\mathbf{x}_i) = +1$

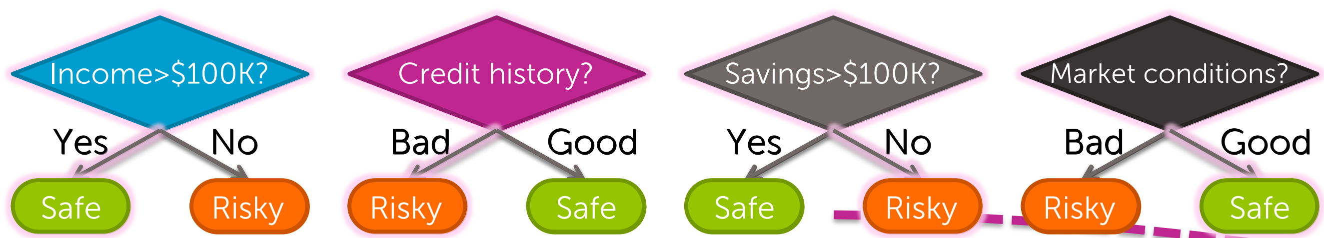
Combine?

Ensemble model

Learn coefficients

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

Prediction with ensemble



$f_1(\mathbf{x}_i) = +1$ $f_2(\mathbf{x}_i) = -1$ $f_3(\mathbf{x}_i) = -1$ $f_4(\mathbf{x}_i) = +1$

Combine?

Ensemble model

Learn coefficients

$= \text{sign}(2 \times +1 + 1.5 \times -1 + 1.5 \times -1 + 0.5 \times +1)$

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

$= \text{sign}(-0.5) \Rightarrow \hat{y}_i = -1$

w_1	2
w_2	1.5
w_3	1.5
w_4	0.5

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:

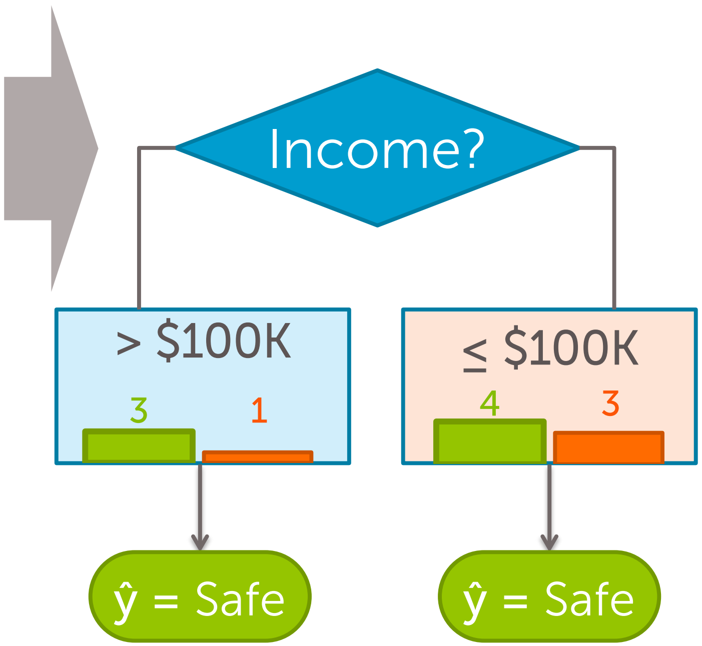
$$\hat{y} = sign\left(\sum_{t=1}^{T} \hat{\mathbf{w}}_t f_t(\mathbf{x})\right)$$

Boosting

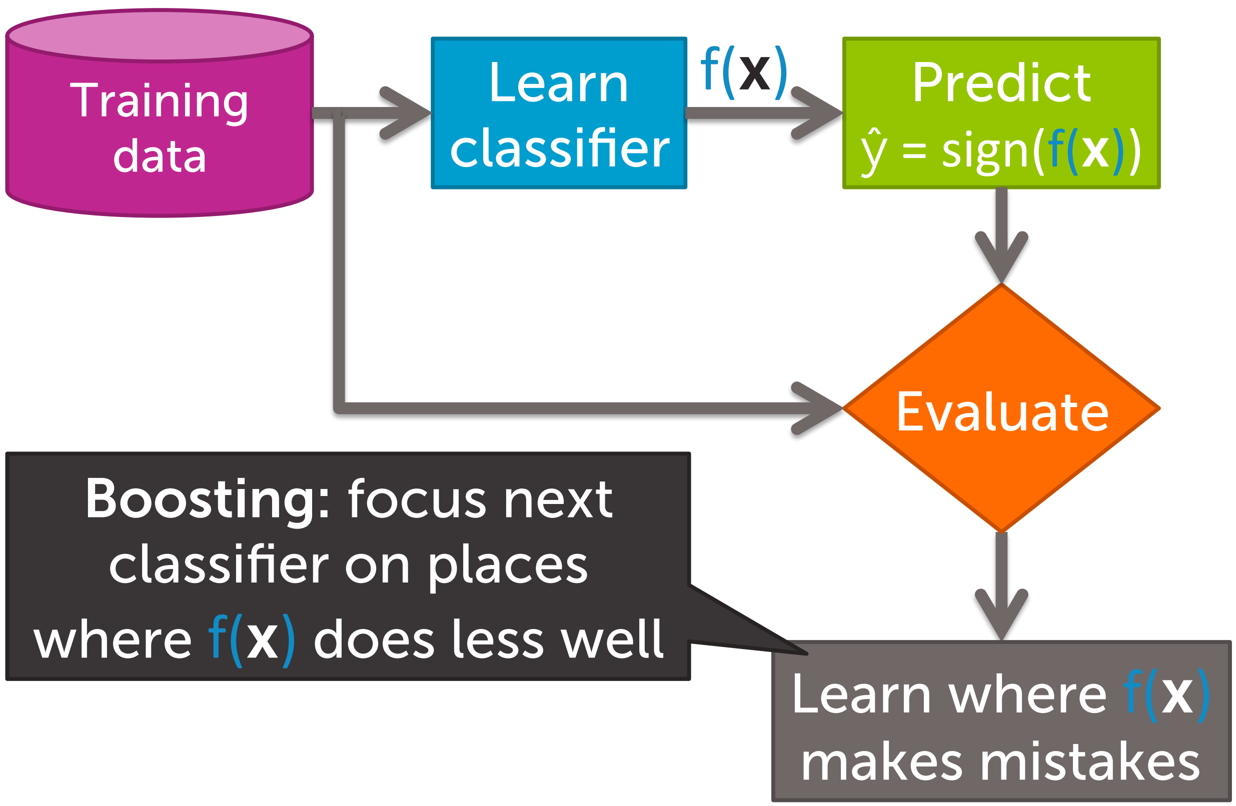
Training a classifier

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



Boosting: focus next classifier on places where f(**x**) does less well

Learning on weighted data:

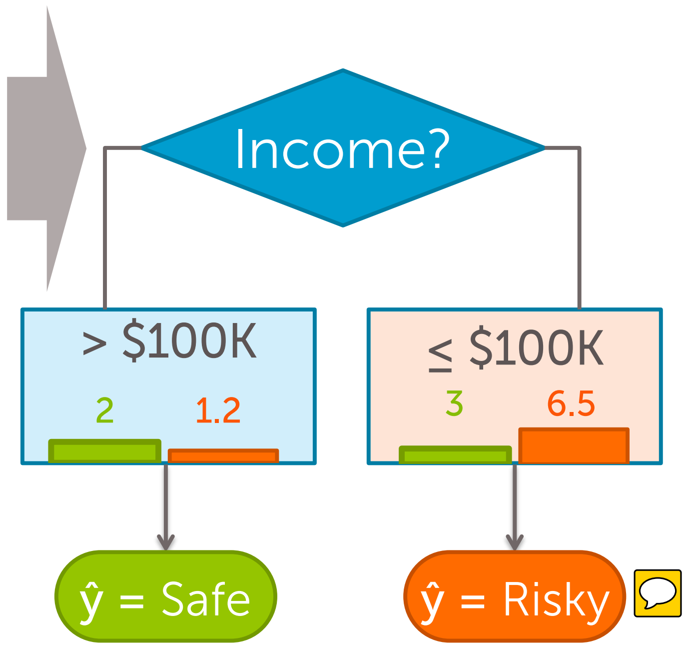
More weight on "hard" or more important points

- Weighted dataset:
  - Each $\mathbf{x}_i, y_i$ weighted by α_i
    - More important point = higher weight α_i
- Learning:
  - Data point j counts as α_i data points
    - E.g., $\alpha_i = 2 \rightarrow$ 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



Learning from weighted data in general

- Usually, learning from weighted data
  - Data point i counts as α_i data points

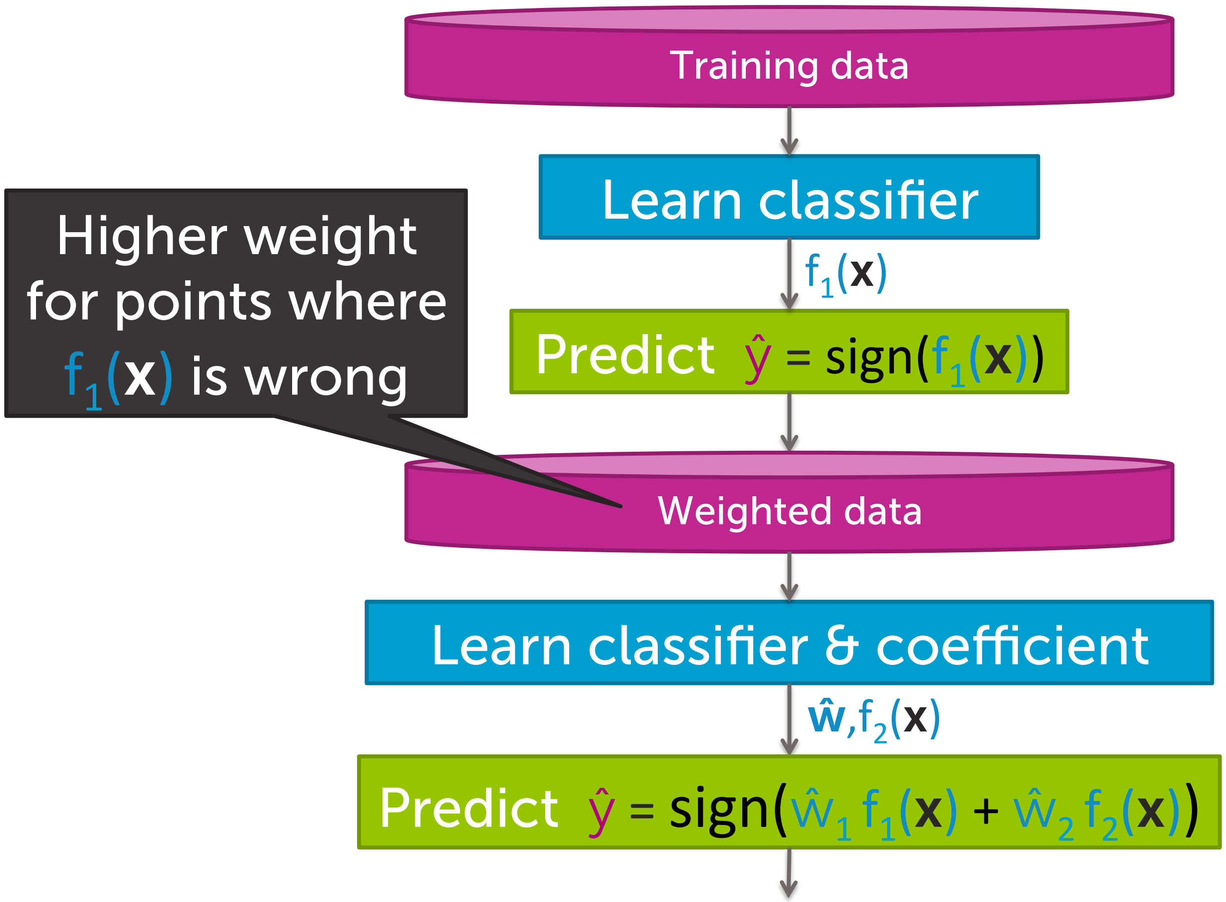
- E.g., gradient ascent for logistic regression:

Sum over data points

$$\mathbf{w}_j^{(t+1)} \leftarrow \mathbf{w}_j^{(t)} + \eta \sum_{i=1}^{N} \alpha_i h_j(\mathbf{x}_i)\Big(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w}^{(t)})\Big)$$

Weigh each point by α_i

Boosting = Greedy learning ensembles from data



AdaBoost

AdaBoost: learning ensemble

[Freund & Schapire 1999]

- Start 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

Problem 1: How much do I trust f_t?

Problem 2: weigh mistakes more?

- Final model predicts by:

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

Computing coefficient $\hat{w}_t$

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

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

Yes → $\hat{w}_t$ large

No → $\hat{w}_t$ small

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

Weighted classification error

Learned classifier

$\hat{y}$ = +

Data point

(Sushi was great, +, α=1.2)
(Food was OK, −, α=0.5)

Correct!
Mistake!

Weight of correct	1.2
Weight of mistakes	0.5

Hide label

Weighted classification error

- Total weight of mistakes:

$$= \sum_{i=1}^{N} \alpha_i \underbrace{\mathbb{1}(\hat{y}_i \neq y_i)}_{\text{Mistake?}}$$

- Total weight of all points:

$$= \sum_{i=1}^{N} \alpha_i$$

- Weighted error measures fraction of weight of mistakes:

$$\boxed{\text{weighted\_error} = \frac{\text{Total weight of mistakes}}{\text{Total weight of all data points}}}$$

  – Best possible value is 0.0 → worst → 1.0 → Random classifier = 0.5

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

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

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

weighted_error(f_t) on training data	$\frac{1 - weighted\_error(f_t)}{weighted\_error(f_t)}$	$\hat{\mathbf{w}}_t$
Yes: 0.01	$\frac{1-0.01}{0.01} = 99$	$\frac{1}{2}\ln 99 = 2.3$
No: 0.5	$\frac{1-0.5}{0.5} = 1$	0
No: 0.99	$\frac{1-0.99}{0.99} = 0.01$	-2.3

Terrible classifier, but $1-f_t$ is awesome!!

AdaBoost: learning ensemble

- Start 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

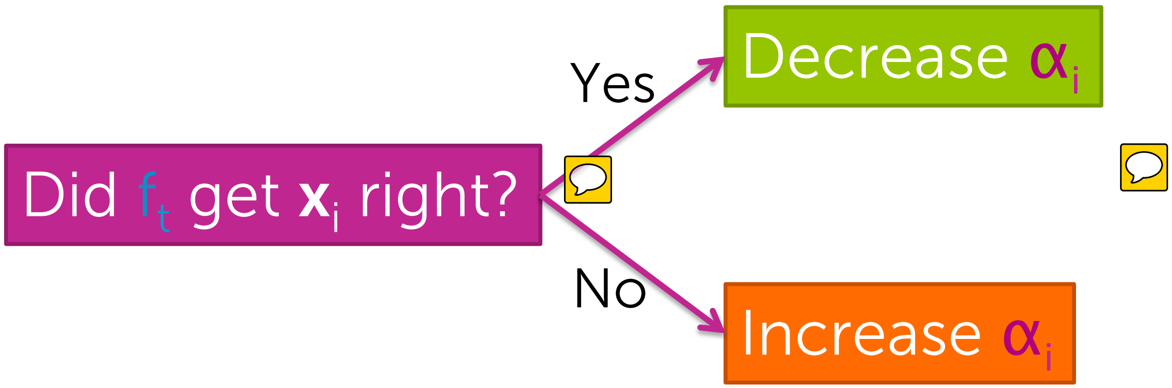
$$\hat{\mathbf{w}}_t = \frac{1}{2} \ln \left(\frac{1 - weighted\_error(f_t)}{weighted\_error(f_t)} \right)$$

- 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 $f_t(\mathbf{x})$ makes mistakes



AdaBoost: Formula for updating weights α_i

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

Did f_t get $\mathbf{x}_i$ right? Yes / No

$f_t(\mathbf{x}_i)=y_i$?	$\hat{w}_t$	Multiply α_i by	Implication
Correct	2.3	$e^{-2.3} = 0.1$	Decrease importance of x_i, y_i
Correct	0	$e^{-0} = 1$	Keep importance the same
Mistake	2.3	$e^{2.3} = 9.98$	Increasing importance of x_i, y_i
Mistake	0	$e^{0} = 1$	Keep importance the same

AdaBoost: learning ensemble

- Start 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:

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

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

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

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 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:

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

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

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

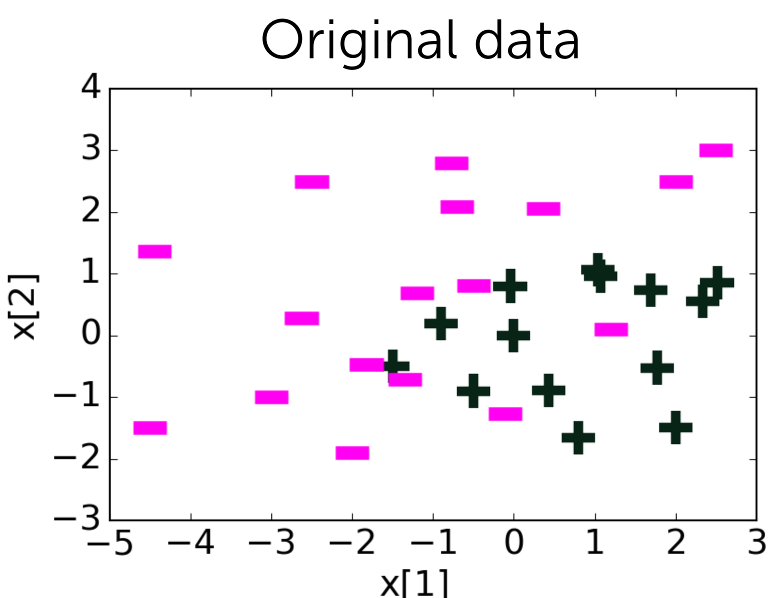
$$\alpha_i \leftarrow \frac{\alpha_i}{\sum_{j=1}^{N} \alpha_j}$$

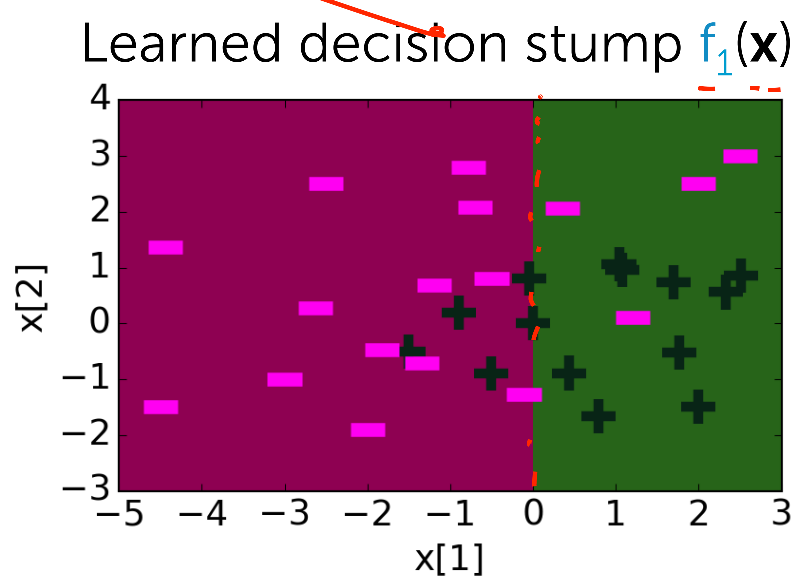
AdaBoost example

t=1: Just learn a classifier on original data

all points have same weight

standard learning

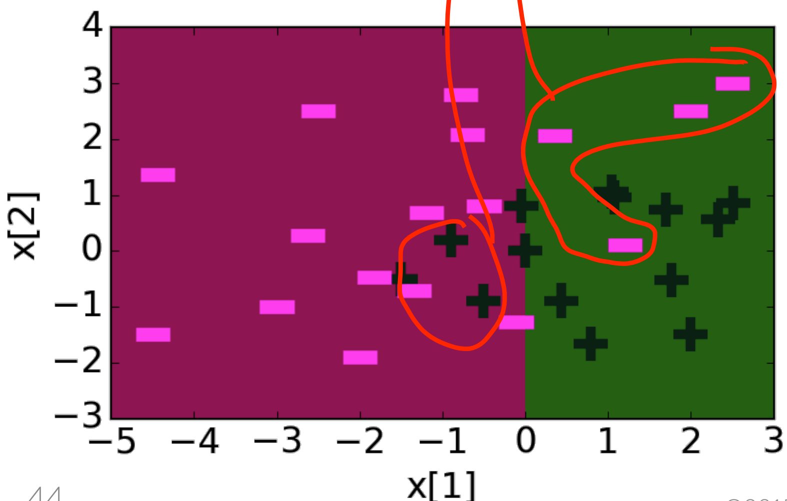




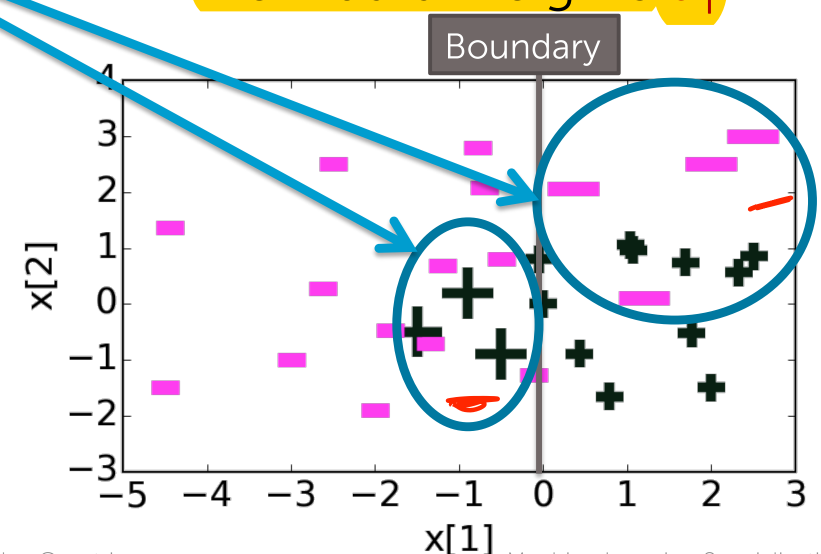
Updating weights α_i

Increase weight α_i of misclassified points

Learned decision stump $f_1(\mathbf{x})$

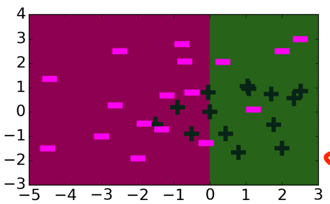


New data weights α_i

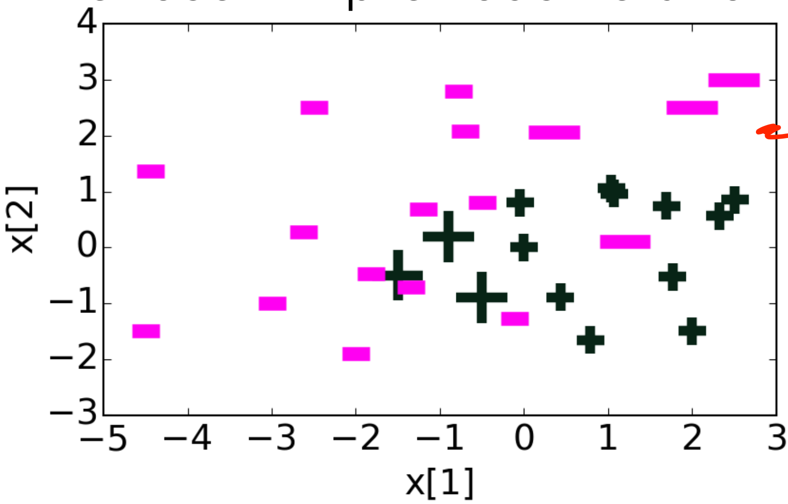


t=2: Learn classifier on weighted data

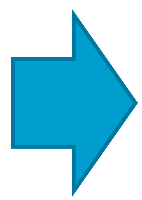
$f_1(\mathbf{x})$



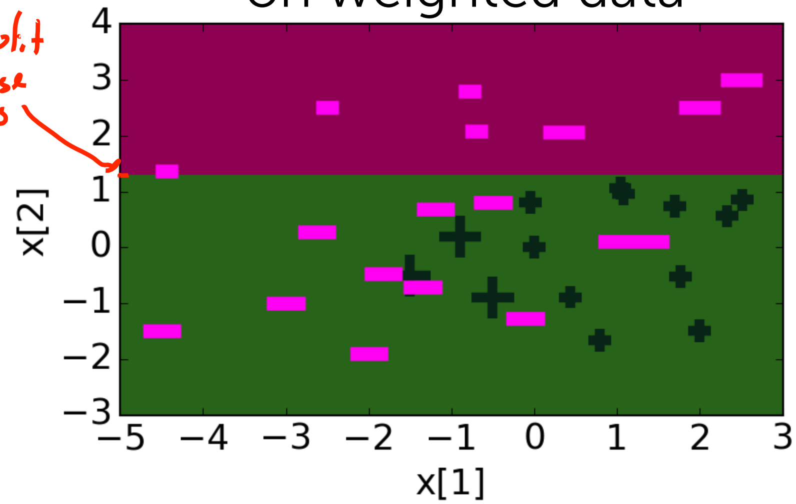
Weighted data: using α_i chosen in previous iteration



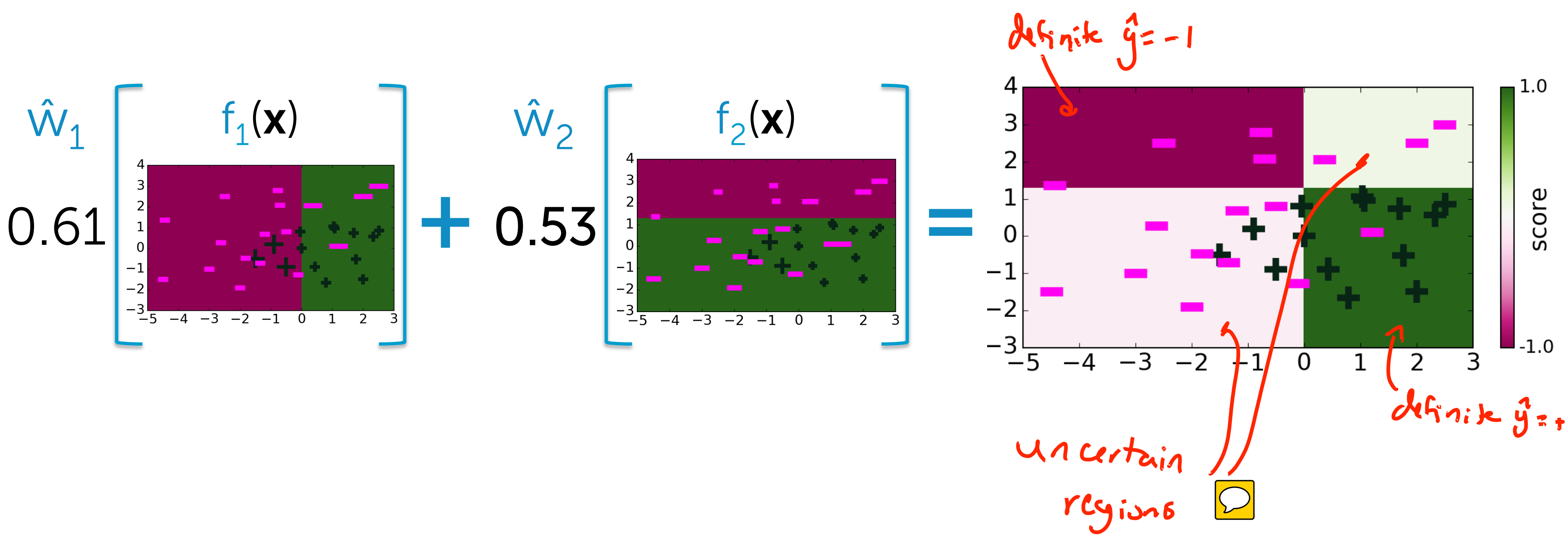
better split for these weights



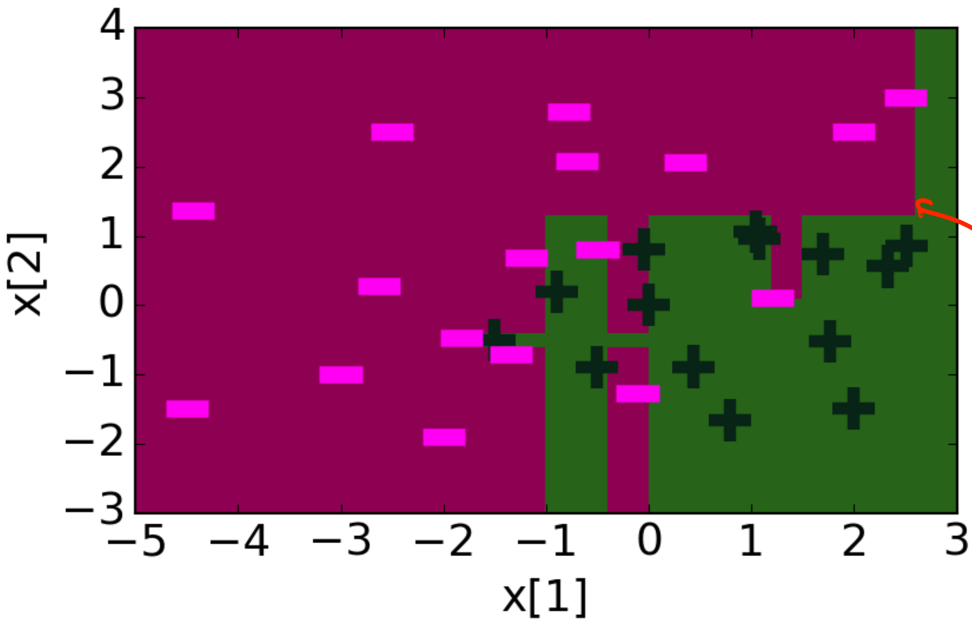
Learned decision stump $f_2(\mathbf{x})$ on weighted data



Ensemble becomes weighted sum of learned classifiers



Decision boundary of ensemble classifier after 30 iterations



Decision boundary is crazy!!

training_error = 0

Probably overfitting

AdaBoost summary

AdaBoost: learning ensemble

- Start same weight for all points: $\alpha_i = 1/N$

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

- For t = 1,…,T
  - Learn $f_t(\mathbf{x})$ with data weights α_i
  - $\boxed{\text{Compute coefficient } \hat{w}_t}$
  - $\boxed{\text{Recompute weights } \alpha_i}$
  - $\boxed{\text{Normalize weights } \alpha_i}$

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

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

- Final model predicts by:

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

Boosted decision stumps

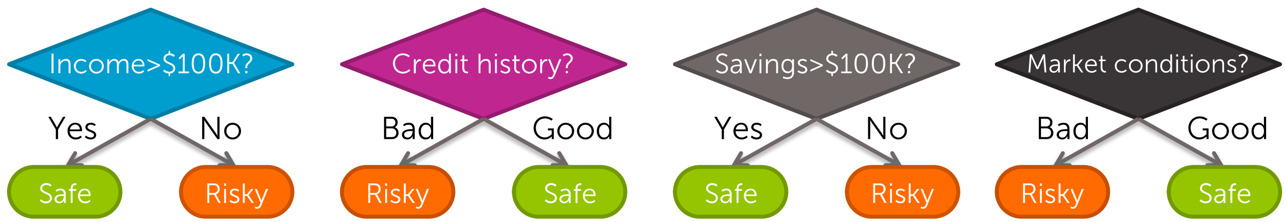
Boosted decision stumps

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

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

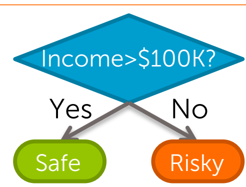
Finding best next decision stump $f_t(\mathbf{x})$

Consider splitting on each feature:



weighted_error = 0.2	weighted_error = 0.35	weighted_error = 0.3	weighted_error = 0.4

f_t =



$$\hat{W}_t = \frac{1}{2}\ln\left(\frac{1 - weighted\_error(f_t)}{weighted\_error(f_t)}\right) = 0.69$$

Boosted decision stumps

- Start same weight for all points: $\alpha_i = 1/N$
- For t = 1,…,T
  - Learn $f_t(\mathbf{x})$: pick decision stump with lowest weighted training error according to α_i
  - Compute coefficient $\hat{w}_t$
  - $\boxed{\text{Recompute weights } \alpha_i}$
  - Normalize weights α_i
- Final model predicts by:

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

Updating weights α_i

$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t} = \alpha_i e^{-0.69} = \boxed{\alpha_i/2}, & \text{if } f_t(\mathbf{x}_i) = y_i \\ \alpha_i e^{\hat{w}_t} = \alpha_i e^{0.69} = \boxed{2\,\alpha_i}, & \text{if } f_t(\mathbf{x}_i) \neq y_i \end{cases}$$

Credit	Income	y	ŷ	Previous weight α	New weight α
A	$130K	Safe	Safe	0.5	0.5/2 = 0.25
B	$80K	Risky	Risky	1.5	0.75
C	$110K	Risky	Safe	1.5	2 * 1.5 = 3
A	$110K	Safe	Safe	2	1
A	$90K	Safe	Risky	1	2
B	$120K	Safe	Safe	2.5	1.25
C	$30K	Risky	Risky	3	1.5
C	$60K	Risky	Risky	2	1
B	$95K	Safe	Risky	0.5	1
A	$60K	Safe	Risky	1	2
A	$98K	Safe	Risky	0.5	1

Boosting convergence & overfitting

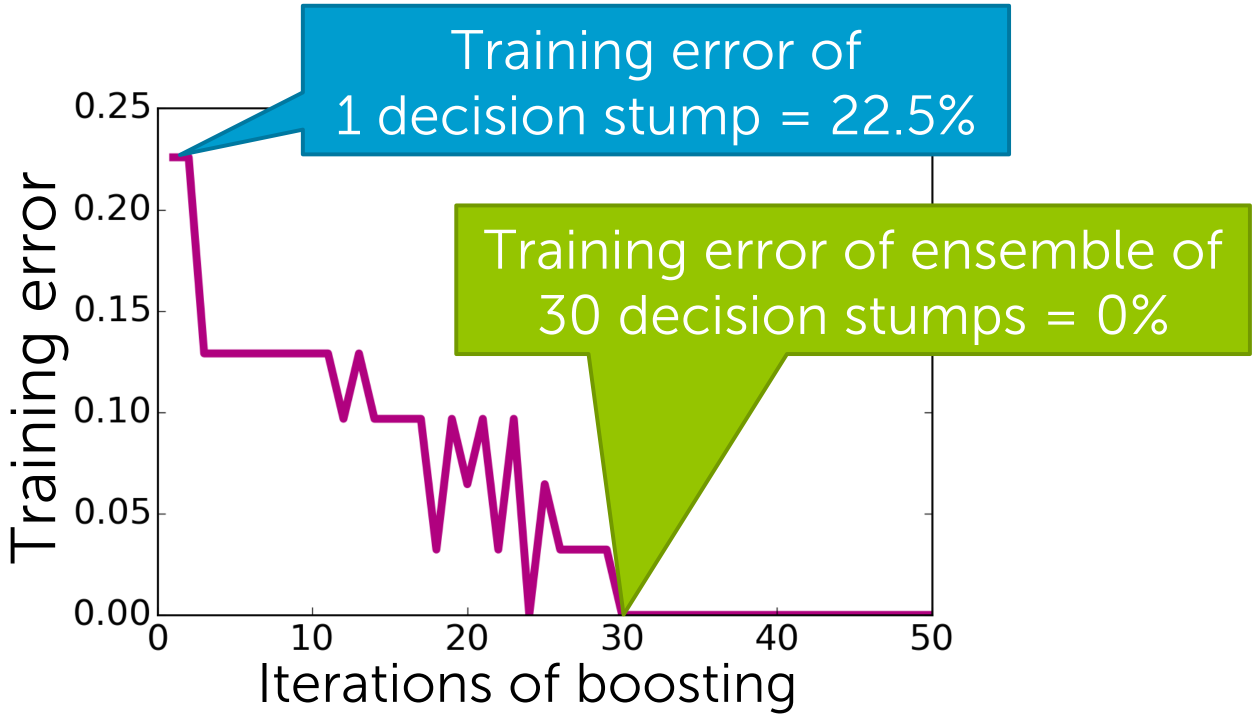
Boosting question revisited

"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!!!

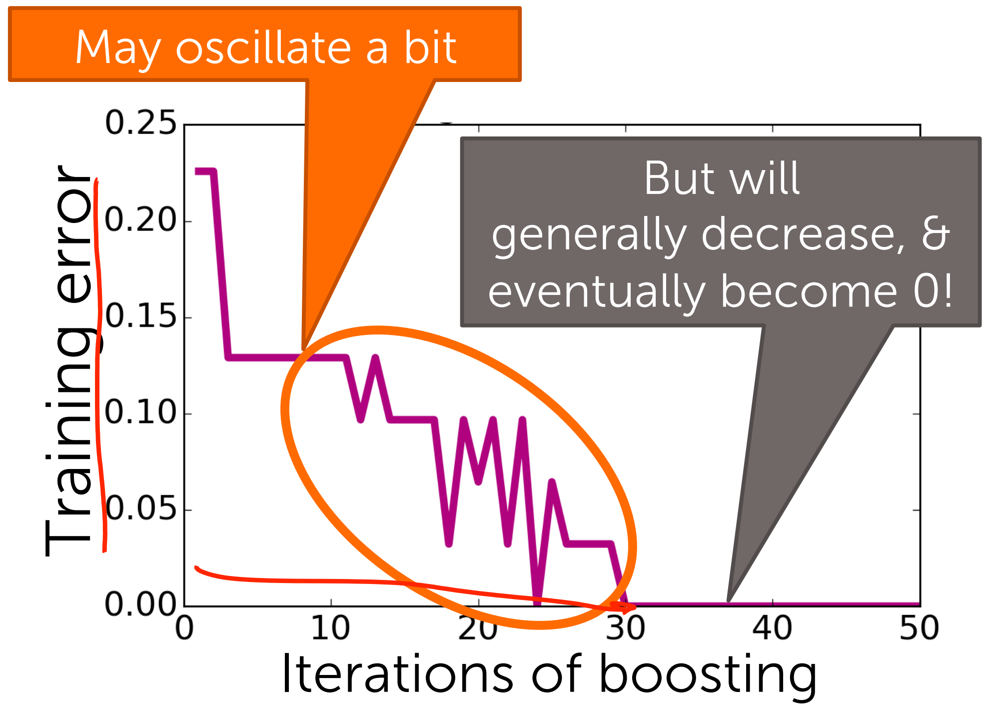


Boosted decision stumps on toy dataset

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→∞

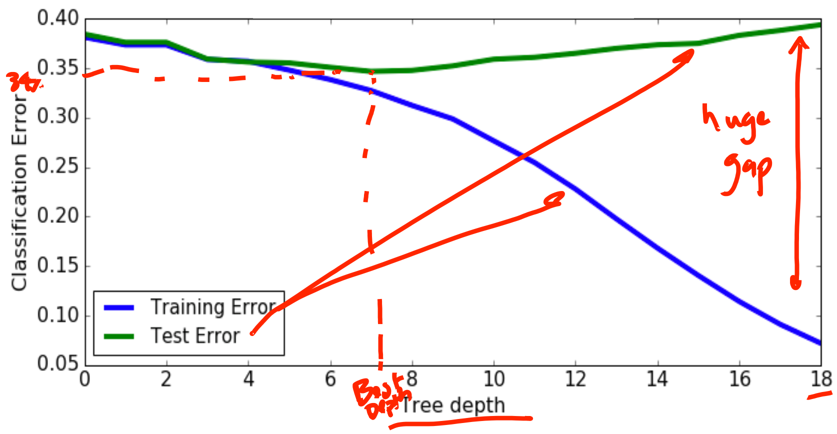
Condition = At every t, can find a weak learner with weighted_error(f_t) < 0.5

Not always possible

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

Nonetheless, boosting often yields great training error

Decision trees on loan data

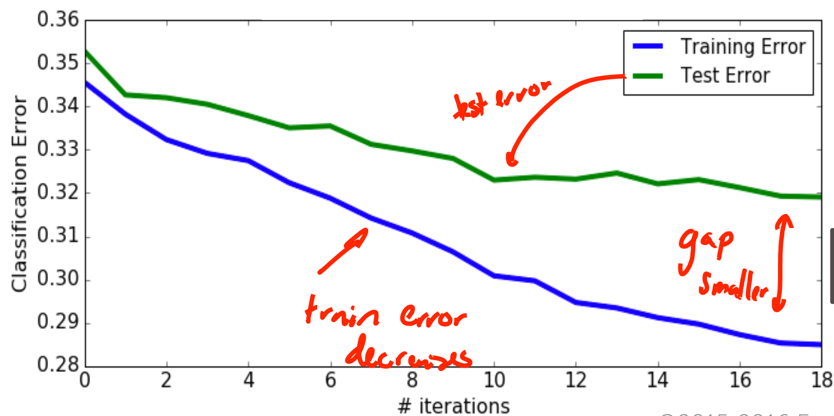


39% test error

8% training error

Boosted decision stumps on loan data

better

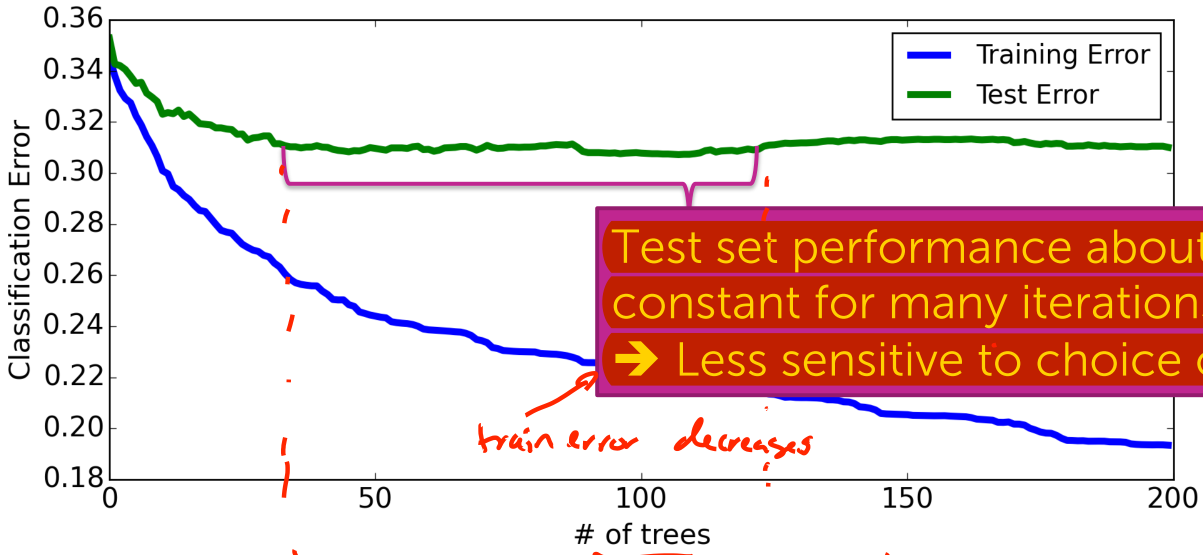


32% test error

Better fit & lower test error

28.5% training error

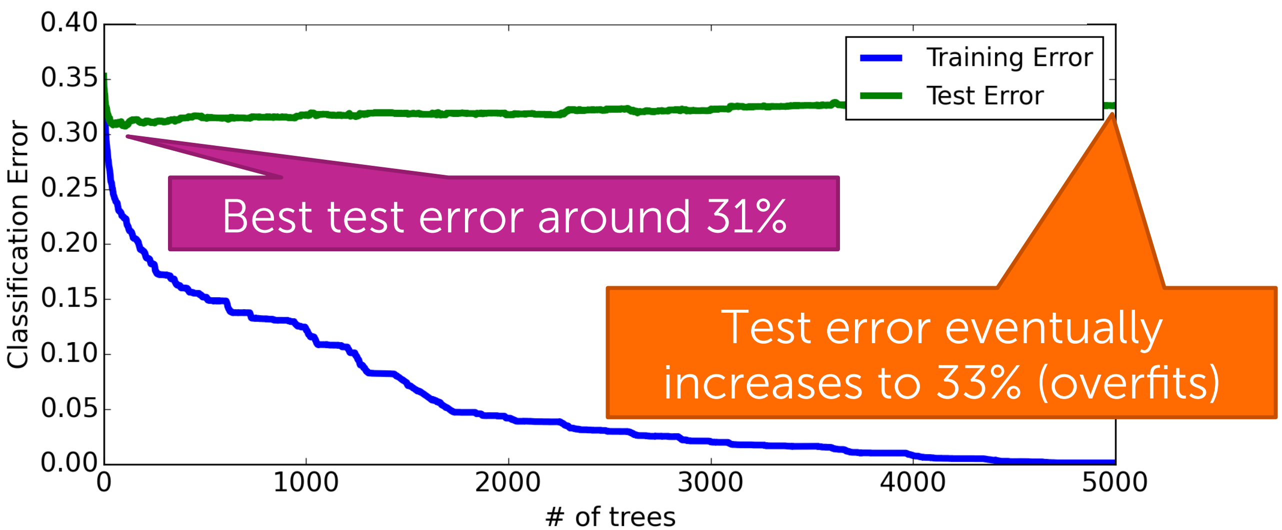
Boosting tends to be robust to overfitting



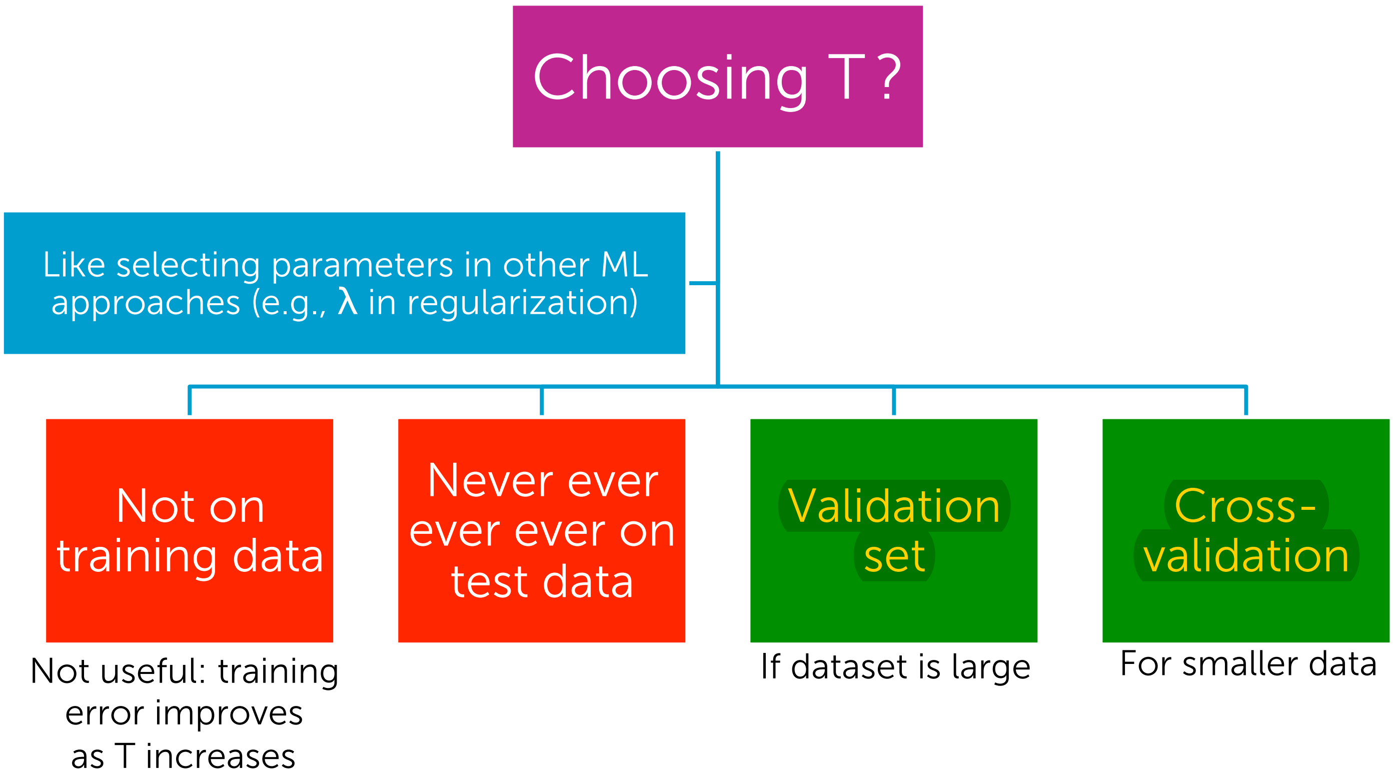
Test set performance about constant for many iterations
➔ Less sensitive to choice of T

any of these values for T would be fine

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



How do we decide when to stop boosting?



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

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 number 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 the 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
- Discuss convergence properties of AdaBoost & how to pick the maximum number of iterations T