BOOSTING

The boosting question

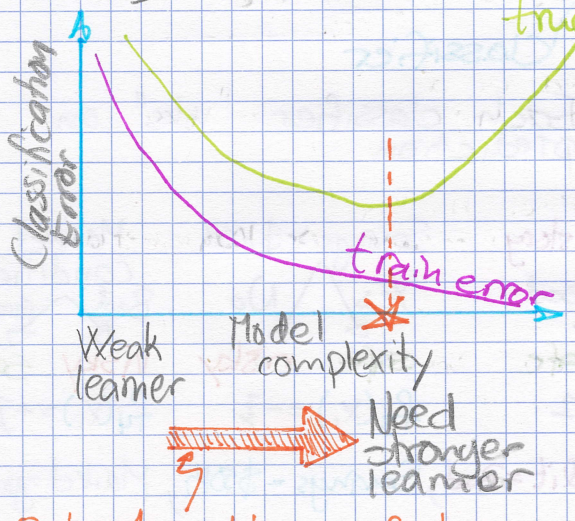
Simple (weak) classifiers are good!

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: ????

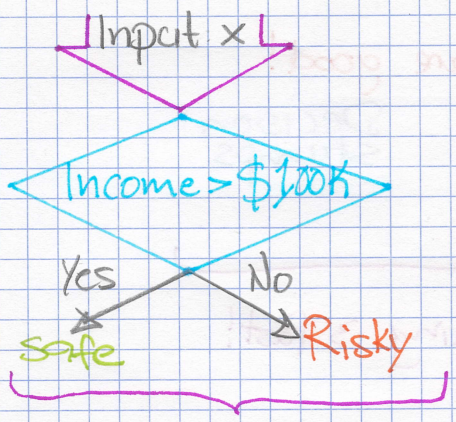
"Can a set of weak learners be combined to create a stronger learner?"

Yes! Schapire (1990)

Boosting

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

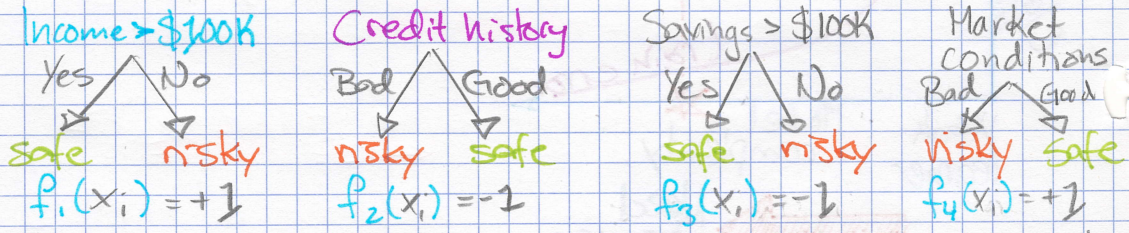
Ensemble classifier



Output: $\hat{y} = f(x)$ ← Classifier

- Either +1 or -1

Ensemble methods: Each classifier "votes" on prediction



x_i = (income = $120K, Credit = Bad, Savings = $50K, Market = good)

Combine?

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

F: Ensemble model

$w_1 f_1, w_2 f_2, w_3 f_3, w_4 f_4$: learn coefficients

Prediction with ensemble

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

w_1	2
w_2	1.5
w_3	1.5
w_4	0.5

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

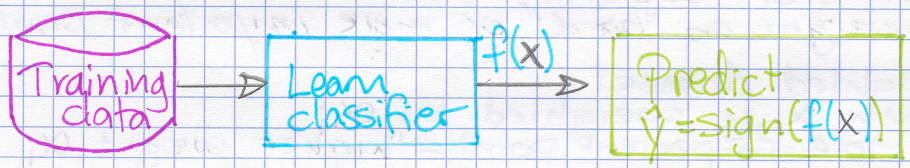
Ensemble classifier in general

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

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

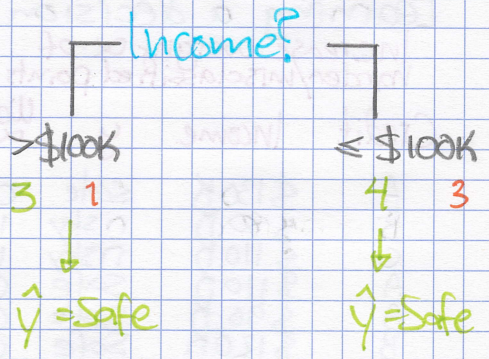
Boosting

Training a classifier

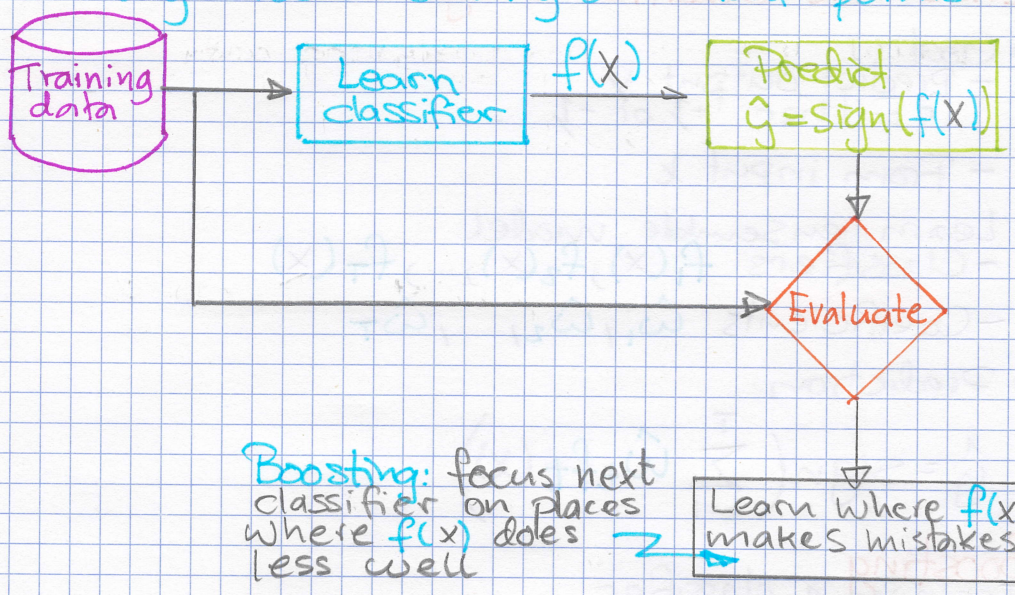


Learning decision stump

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 → Learn where $f(x)$ makes mistakes

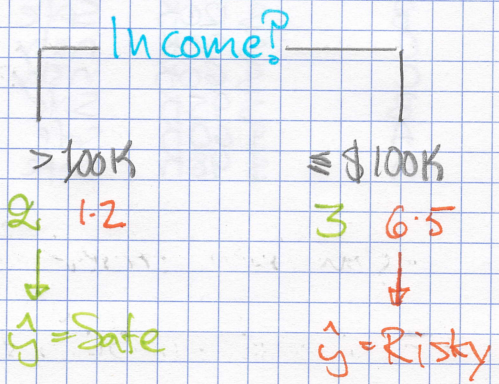
Learning on weighted data:
More weight on "hard" or more important points

- Weighted dataset:
  - Each 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



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

$$w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta \sum_{i=1}^{N} \alpha_i h_j(x_i)\left(\mathbb{1}[y_i = +1] - P(y = +1 | x_i, w^{(t)})\right)$$

Weigh each point by α_i

Boosting = Greedy learning ensembles from data

Training data

↓

Learn classifier

↓

Predict $\hat{y} = \text{sign}(f_1(x))$

↓

Higher weight for points where $f_1(x)$ is wrong → Weighted data

↓

Learn classifier & coefficient

↓

Predict $\hat{y} = \text{sign}(\hat{w}_1 f_1(x) + \hat{w}_2 f_2(x))$

AdaBoost overview

AdaBoost: learning ensemble
[Freund & Schapire 1999]

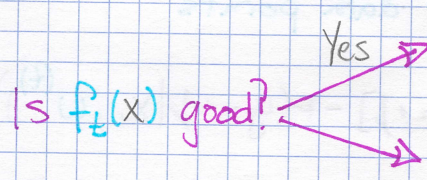
- Start same weight for all points: $\alpha_i = 1/N$
- For $t = 1, \ldots, T$
  - Learn $f_t(x)$ with data weights α_i
  - Compute coefficient $\hat{w}_t$ ← Problem 1: How much do I trust f_t?
  - Recompute weights α_i ← Problem 2: weigh mistakes more?
- Final model predicts by:

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

Weighted errors

Computing coefficient $\hat{w}_t$

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



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

Weighted classification error

"sushi was great" / "Food was OK" → Learned classifier → $\hat{y} = +, +$

Data point
(Sushi was great, +, $\alpha = 1.2$)
(Food was OK, −, $\alpha = 0.5$)

Correct!
Mistake!
+ −
Hide label

Weight of correct 1.2
Weight of mistakes 0.5

Weighted classification error

- Total weight of mistakes:

$$= \sum_{i=1}^{N} \alpha_i \, \mathbb{1}(\hat{y}_i \neq y_i)$$

- Total weight of all points:

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

- Weighted error measures fraction of weight of mistakes

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

- Best possible value is 0.0 → worst ½·0

Computing coefficient of each ensemble component

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

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

Is $f_t(x)$ good?	Weighted_error(f_t) on training data	$\frac{1-\text{weighted\_error}(f_t)}{\text{weighted\_error}(f_t)}$	$\hat{w}$
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
	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 $t = 1, \ldots, T$
  - Learn $f_t(x)$ with data weights α_i
  - Compute coefficient $\hat{w}_t$ — $\hat{w}_t = \frac{1}{2}\ln\left(\frac{1-\text{weighted\_error}(f_t)}{\text{weighted\_error}(f_t)}\right)$
  - Recompute weights α_i —
$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t}, & \text{if } f_t(x_i) = y_i \\ \alpha_i e^{\hat{w}_t}, & \text{if } f_t(x_i) \neq y_i \end{cases}$$
- Final model predicts by

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

Recompute weights α_i

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

Did f_t get x_i right?
- Yes → Decrease α_i
- No → Increase α_i

AdaBoost: Formula for updating weights α_i

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

Did f_t get x_i right?

	$f_t(x_i)=y_i$	$\hat{w}_t$	Multiply α_i by	Implication
Yes	Correct	2.3	$e^{-2.3}=0.1$	Decrease importance of x_i, y_i
Yes	Correct	0	$e^{-0}=1$	Keep importance the same
No	Mistake	2.3	$e^{2.3}=9.98$	Increase importance of x_i, y_i
No	Mistake	0	$e^{0}=1$	Keep importance the same

Normalizing weights

AdaBoost: Normalizing weights α_i

If x_i often mistake, weights α_i gets very large

If x_i often correct, weights α_i gets very small

↓

Can cause numerical instability after many iterations

↓

CONT'D

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 $t = 1, \ldots, T$
  - Learn $f_t(x)$ with data weights α_i
  - Compute coefficient $\hat{w}_t$
  - Recompute weights α_i
  - Normalize weights α_i
- Final model predicts by

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

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

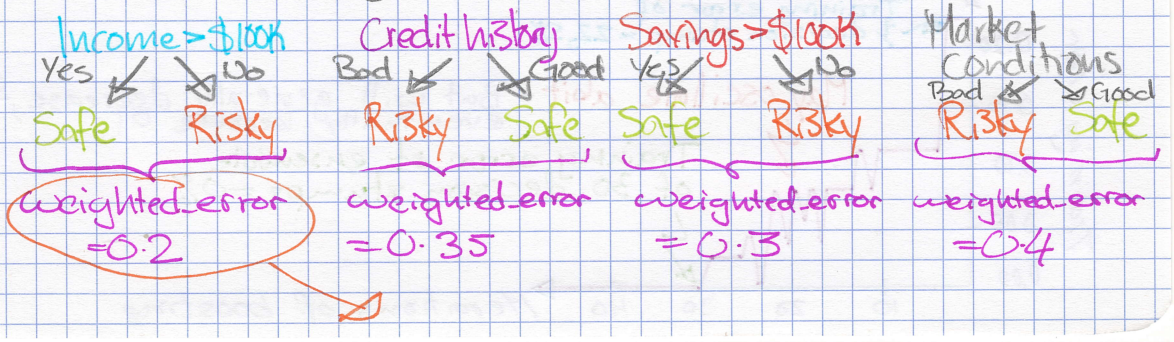
$$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t}, & \text{if } f_t(x_i) = y_i \\ \alpha_i e^{\hat{w}_t}, & \text{if } f_t(x_i) \neq y_i \end{cases}$$

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

Boosted decision stumps

Finding best next decision stump $f_t(x)$

Consider splitting on each feature:



f_t = Income > $100K?
Yes → Safe
No → Risky

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

Updating weights α_i

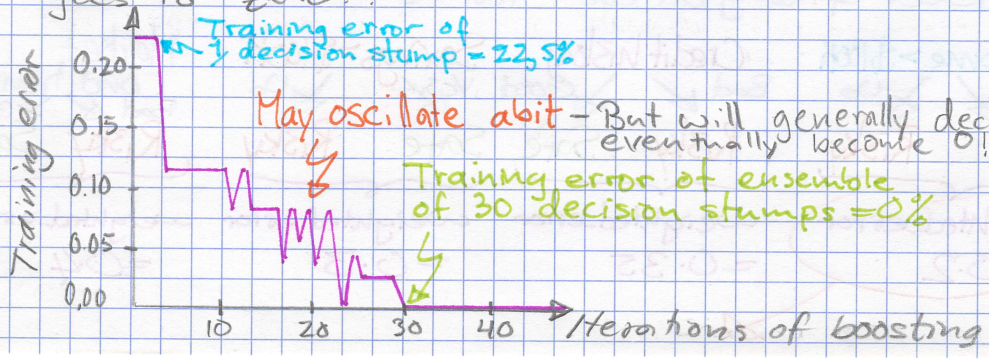
Credit	Income	y	$\hat{y}$	Previous weight α_i	New weight α_i
A	$130K	Safe	Safe	0.5	0.5/2 = 0.25
B	$80K	Risky	Risky	1.5	0.75
C	$110K	Risky	Safe	1.5	2 * 1.5 = 3
A	$110K	Safe	Safe	2	1
A	$90K	Safe	Risky	1	2
B	$120K	Safe	Safe	2.5	1.25
C	$30K	Risky	Risky	3	1.5
C	$60K	Risky	Risky	2	1
B	$95K	Safe	Risky	0.5	1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

Income > $100K?
Yes → Safe
No → Risky

$\alpha_i \leftarrow$
- $\alpha_i e^{-0.69} = \alpha_i/2$, $f_t(x_i) = y_i$
- $\alpha_i e^{0.69} = 2\alpha_i$, $f_t(x_i) \neq y_i$

The Boosting Theorem
Boosting convergence & overfitting

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



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

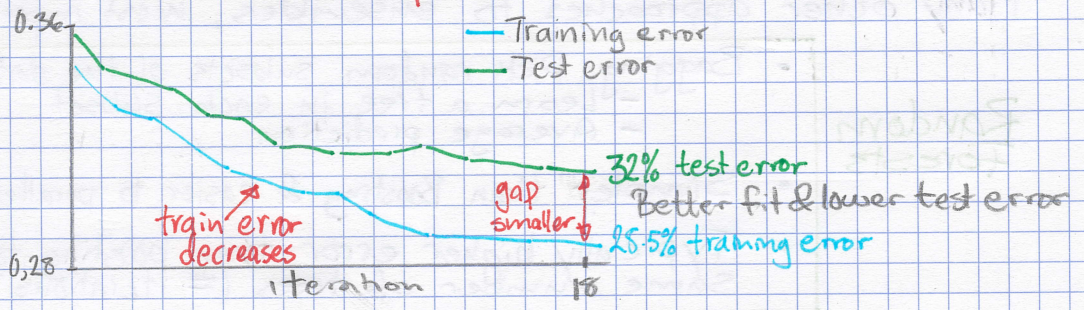
Extreme example: No classifier can separate a +1 on top of -1

⇓

Nonetheless, boosting often yields great training error

Overfitting in Boosting

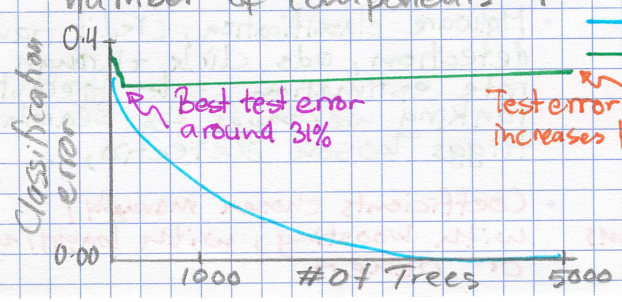
Boosted decision stumps on loan data



Boosting tends to be robust to overfitting

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

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



How do we decide when to stop boosting

Choosing T?

like selecting parameters in other ML approaches (e.g., λ in regularization)

- **Not on training data** — Not useful: training error improves as T increases
- **Never ever ever ever on test data**
- **Validation set** — If dataset is large
- **Cross-Validation** — For smaller data

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 ensembles, most important

Random forests

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

Impact of Boosting

Amongst most useful ML methods ever created

Extremely useful in computer vision

- Standard approach for face detection

Used by most winners of ML competitions

- 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