Lecture 14: Empirical Risk Minimization & Boosting

Winter 2018

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikuar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Learning as Loss Minimization

Empirical loss minimization

Learning = minimize *empirical loss* on the training set

$$\min_{h \in H} \frac{1}{m} \sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

Is there a problem here?

Empirical loss minimization

Learning = minimize *empirical loss* on the training set

$$\min_{h \in H} \frac{1}{m} \sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

Is there a problem here? Overfitting!

We need something that biases the learner towards simpler hypotheses

- Achieved using a regularizer, which penalizes complex hypotheses

$$\min_{h \in H} R(h) + \frac{1}{m} \sum_i [L(h(x_i), f(x_i))]$$

Regularized loss minimization

- Learning: $\min_{h \in H} R(h) + \frac{1}{m} \sum_i [L(h(x_i), f(x_i))]$

- With linear classifiers:

$$H: \{h(x): sgn(w^T x \geq 0)\}, w \in R^d$$

$$\min_{\mathbf{w}} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i L(y_i, \mathbf{x}_i, \mathbf{w})$$

- What is a loss function?
  - Loss functions should penalize mistakes
  - We are minimizing average loss over the training data
- What is the ideal loss function for classification?

The 0-1 loss

Penalize classification mistakes between true label y and prediction y'

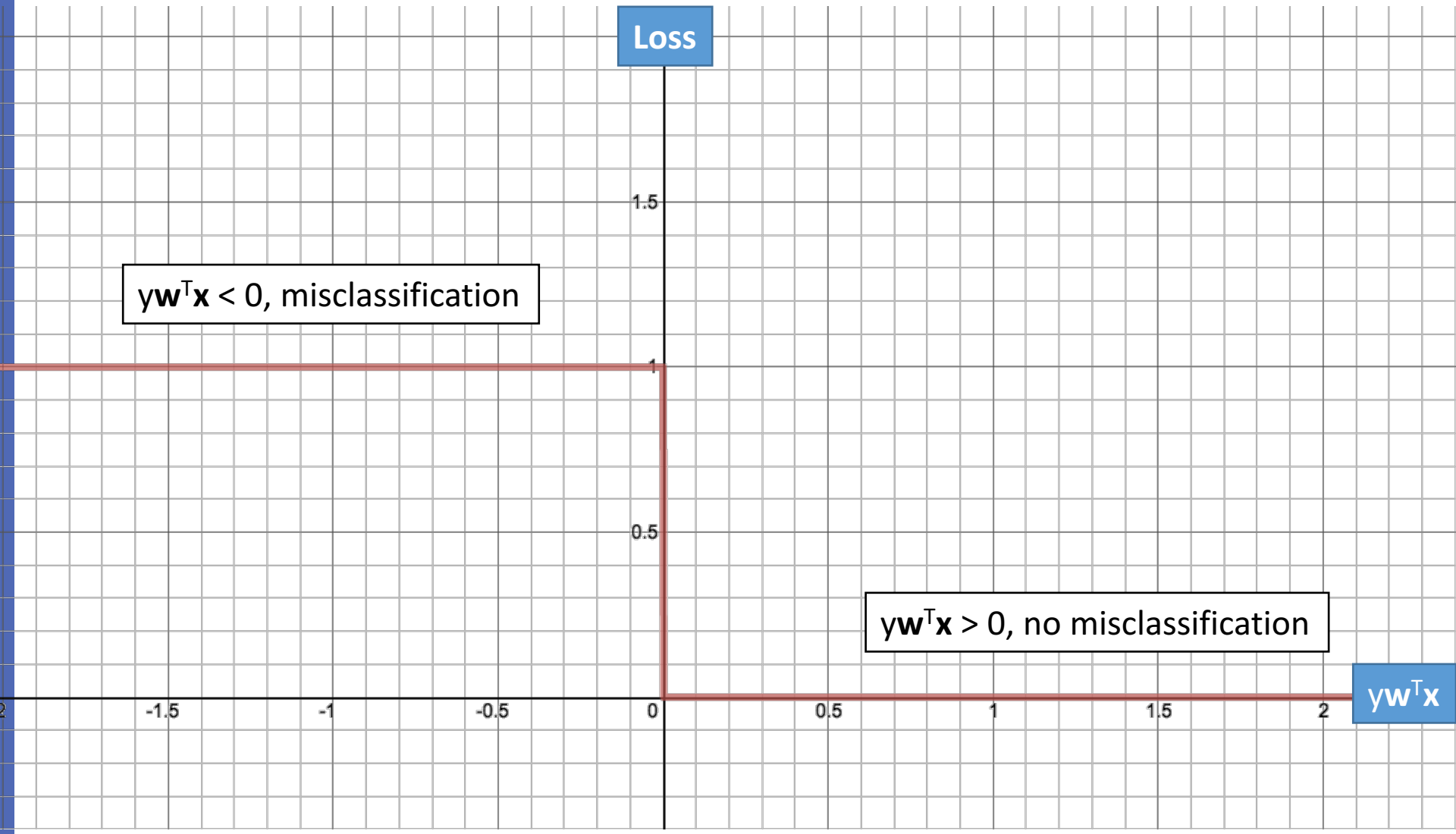
$$L_{0\text{-}1}(y, y') = \begin{cases} 1 & \text{if } y \neq y', \\ 0 & \text{if } y = y'. \end{cases}$$

- For linear classifiers, the prediction is sgn($\mathbf{w}^T\mathbf{x}$)
  - Mistake if $y\,\mathbf{w}^T\mathbf{x} \leq 0$

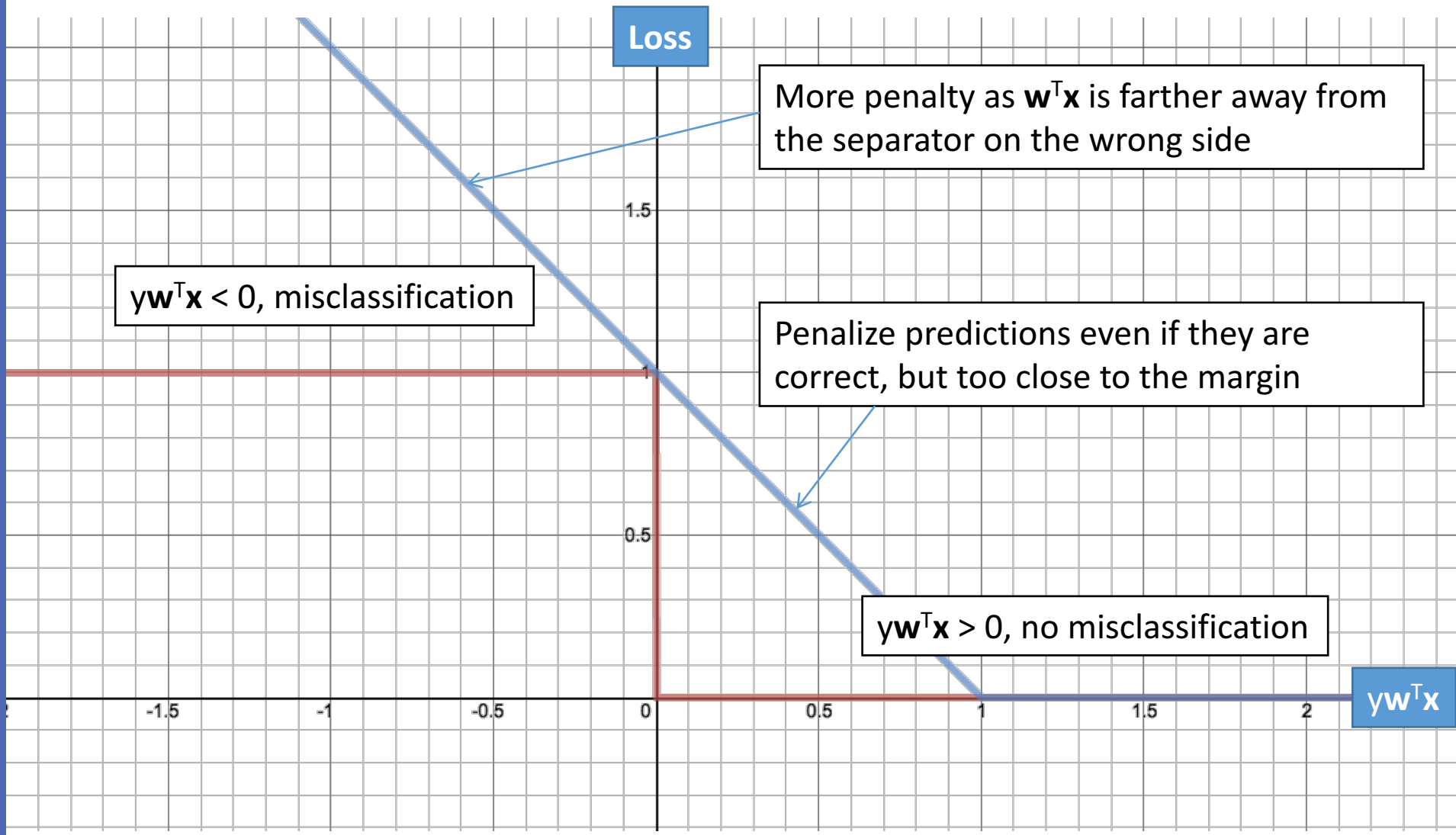
$$L_{0\text{-}1}(y, \mathbf{x}, \mathbf{w}) = \begin{cases} 1 & \text{if } y\ \mathbf{w}^T\mathbf{x} \leq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Minimizing 0-1 loss is intractable. Need surrogates

The 0-1 loss


Loss
1.5
1
0.5
$y\mathbf{w}^T\mathbf{x} < 0$, misclassification
$y\mathbf{w}^T\mathbf{x} > 0$, no misclassification
-1.5
-1
-0.5
0
0.5
1
1.5
2
$y\mathbf{w}^T\mathbf{x}$

Compare to the hinge loss



Support Vector Machines

- SVM = linear classifier combined with regularization

- Ideally, we would like to minimize 0-1 loss,
  - But we can't for computational reasons

- SVM minimizes hinge loss

$$L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$$

  - Variants exist

$$\min_{h \in H} \text{regularizer}(h) + C \frac{1}{m} \sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

The loss function zoo

Many loss functions exist

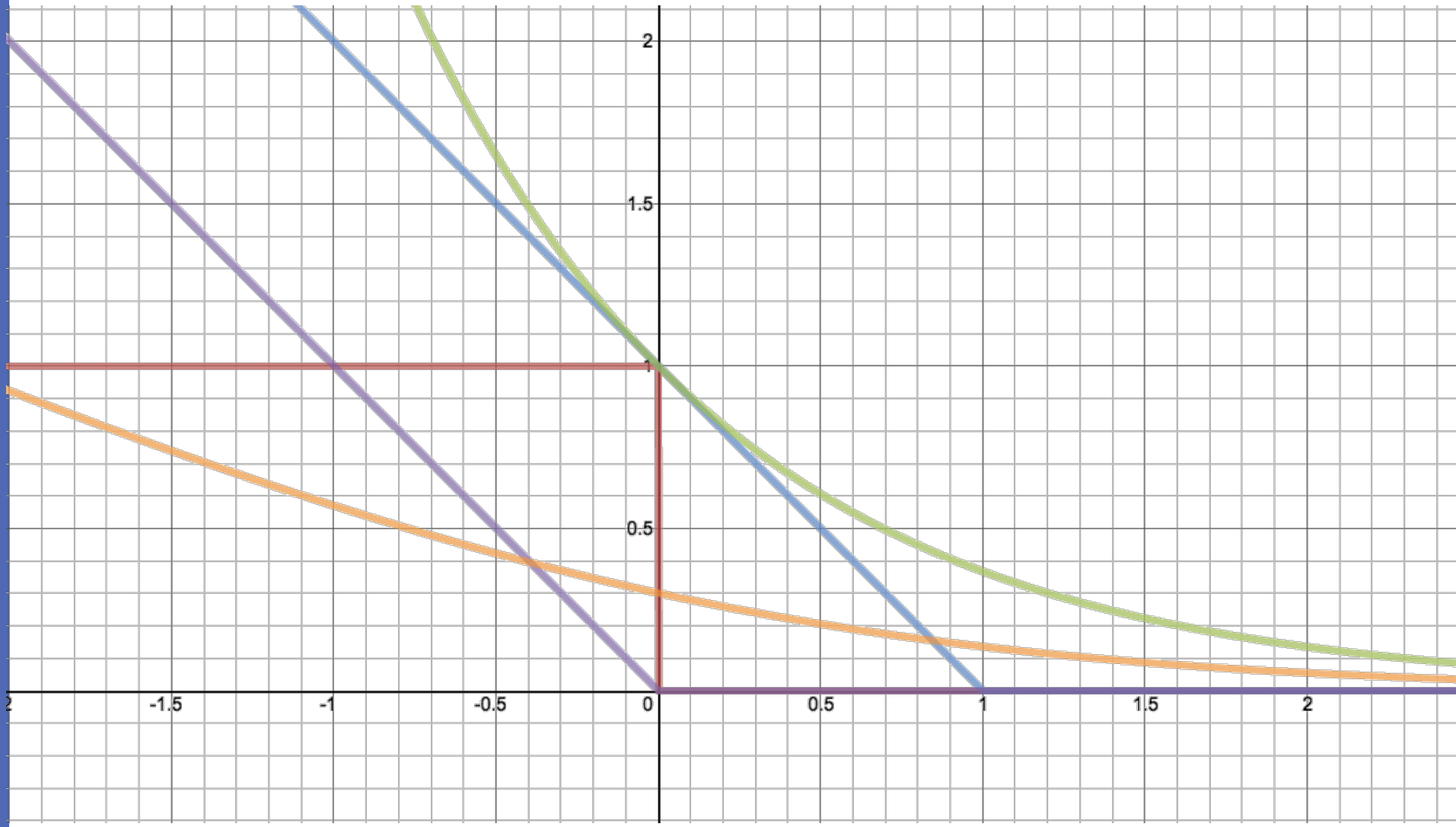
- Perceptron loss $L_{Perceptron}(y, \mathbf{x}, \mathbf{w}) = \max(0, -y\mathbf{w}^T\mathbf{x})$
- Hinge loss (SVM) $L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$
- Logistic loss (logistic regression)

$$L_{Logistic}(y, \mathbf{x}, \mathbf{w}) = \log(1 + e^{-y\mathbf{w}^T\mathbf{x}})$$

http://www.cs.toronto.edu/~mbrubake/teaching/C11/Handouts/AdaBoost.pdf

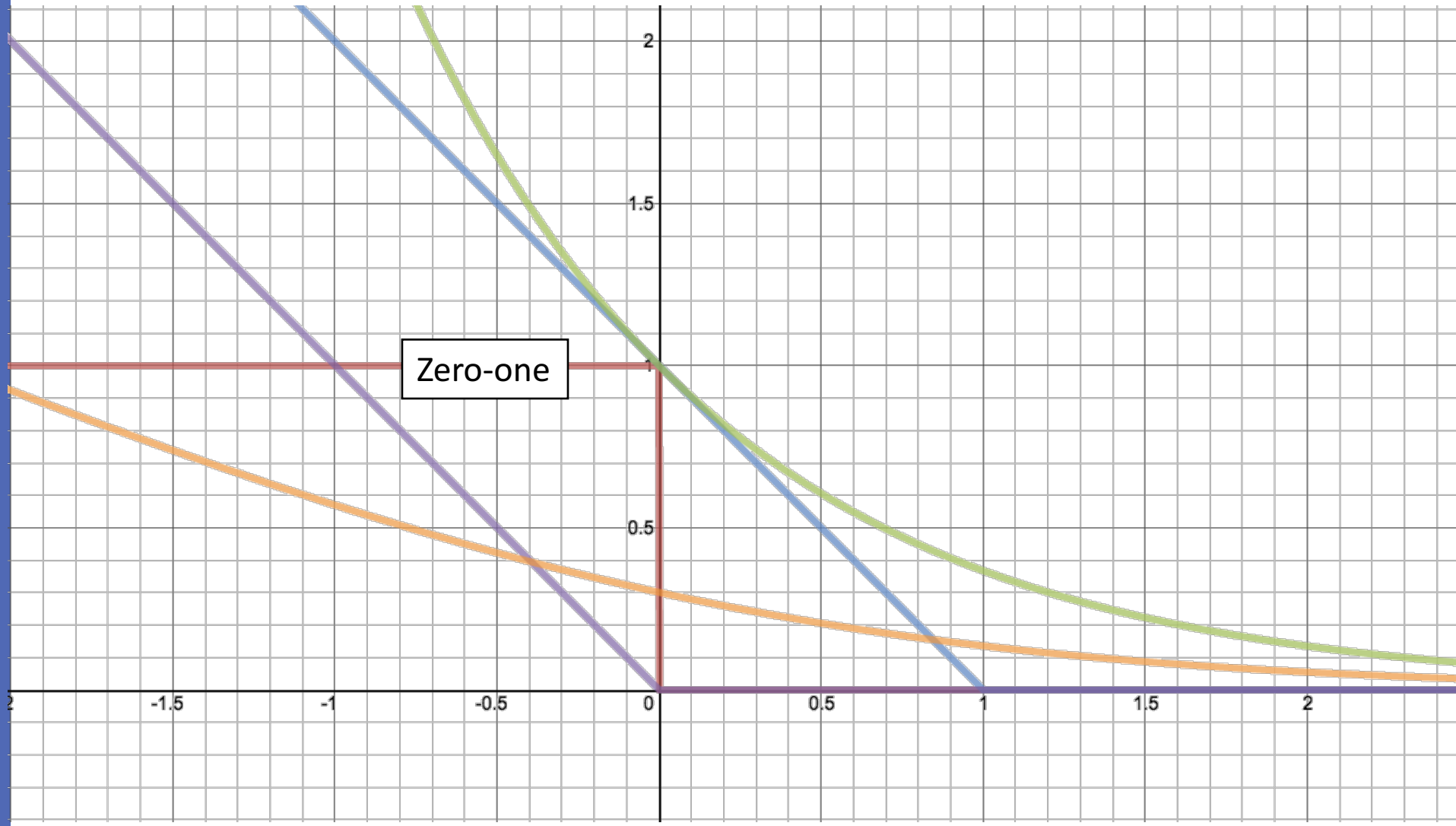
$$\min_{h \in H} \text{regularizer}(h) + C \frac{1}{m} \sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

The loss function zoo



$$\min_{h \in H} \text{regularizer}(h) + C \frac{1}{m} \sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

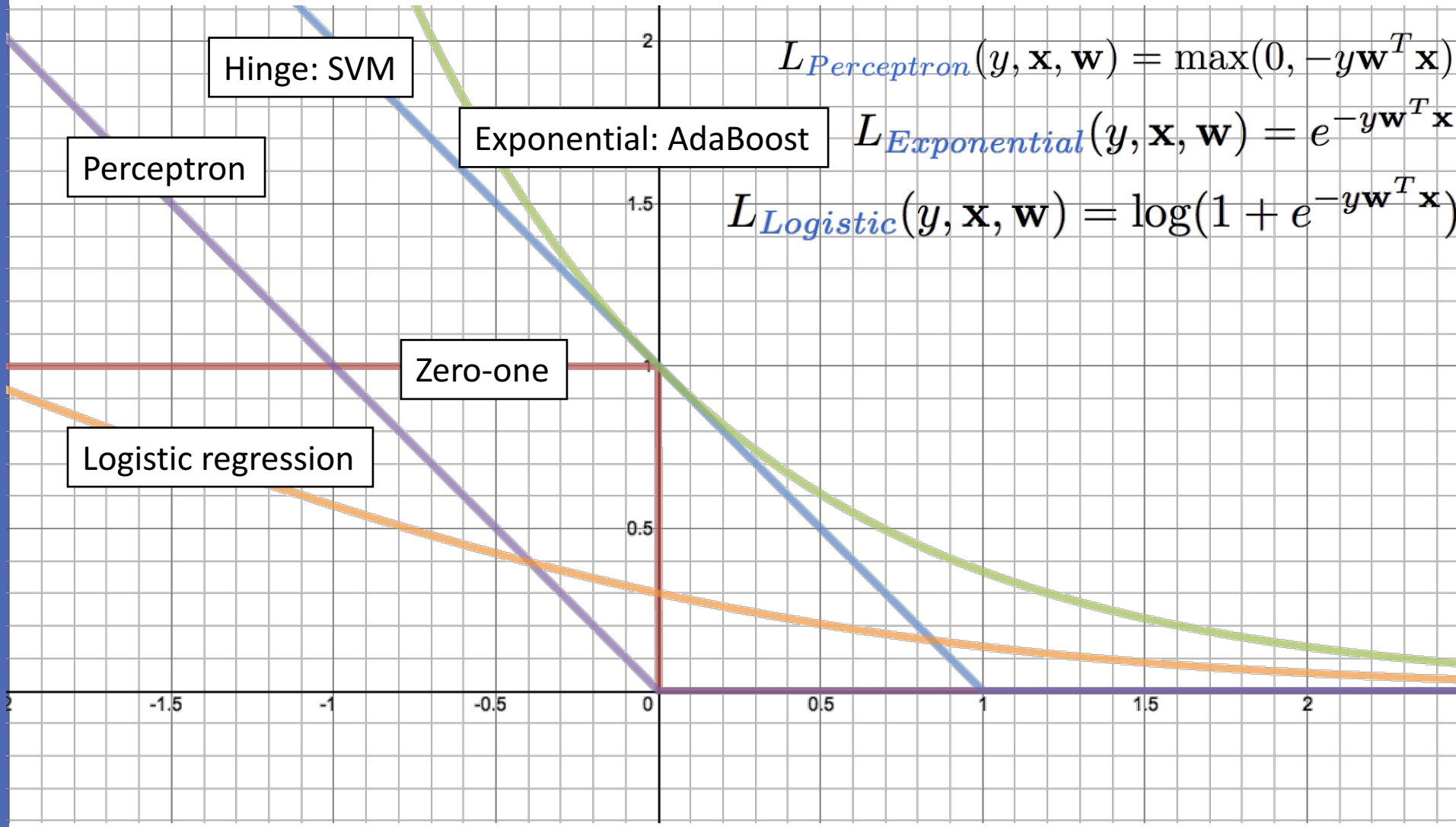
The loss function zoo



$$\min_{h \in H} \text{regularizer}(h) + C \frac{1}{m} \sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

The loss function zoo

$$L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$$



$$L_{Perceptron}(y, \mathbf{x}, \mathbf{w}) = \max(0, -y\mathbf{w}^T\mathbf{x})$$

$$L_{Exponential}(y, \mathbf{x}, \mathbf{w}) = e^{-y\mathbf{w}^T\mathbf{x}}$$

$$L_{Logistic}(y, \mathbf{x}, \mathbf{w}) = \log(1 + e^{-y\mathbf{w}^T\mathbf{x}})$$

Approximate 0-1 loss

- Usually, we cannot minimize 0-1 loss
  - It is a combinatorial optimization problem: NP-hard
- Idea: minimizing its upper-bound

Many choices of R(w)

- Minimizing the empirical loss with linear function

$$\min_{w \in R^d} \; R(w) \; + \; \frac{1}{\left|\widehat{D}\right|} \sum_{(x,y)\in \widehat{D}} [L(x, w, y)]$$

- Prefer simpler model: (how?)
  - Sparse:
    $R(w)$ = #non-zero elements in w (L0 regularizer)
    $R(w) = \sum_i |w_i|$ (L1 regularizer)
  - Gaussian prior (large margin w/ hinge loss):
    $R(w) = \sum_i w_i^2 = w^T w$ (L2 regularizer)

Learning via Loss Minimization: Summary

- Learning via Loss Minimization
  - Write down a loss function
  - Minimize empirical loss
- Regularize to avoid overfitting
  - Neural networks use other strategies such as dropout
- Widely applicable, different loss functions and regularizers

Boosting and Ensembles

Yet another way to introduce non-linearity in linear models

Two heads are better than one

- Is it true in ML?
  - Can we get a stronger learner by combining many weak learners?

Boosting and Ensembles

- Ensemble methods
- BoostingAdaBoost

Boosting and Bagging

- A general learning approach for constructing a *strong learner*, given a collection of (possibly infinite) weak learners

- Historically: An answer to a theoretical question in PAC learning

The Strength of Weak Learnability

ROBERT E. SCHAPIRE

1989-90

Why it may work?

- The predictors make different types of mistakes
- Combine them may make a stronger predictor

Practical story: the Netflix prize

- Goal: predict how a user will rate a movie
  - Based on other user's ratings of the movie.
  - Based on this user's ratings of other movies.
  - Application called collaborative filtering.
  - Netflix prize: 1M prize for the first team to do 10% better than the Netflix system.
- Winner: an ensemble of more than 800 rating systems.

Bagging

Short for ***B**ootstrap **agg**regating* [Breiman, 1994]

- Given a training set with m examples
- Repeat t = 1, 2, ⋯, m:
  - Draw m' (< m) samples with replacement from the training set
  - Train a classifier (any classifier you like) h_i
- Construct final classifier by taking votes from each h_i
- Any other ideas beyond simple voting?

Bagging

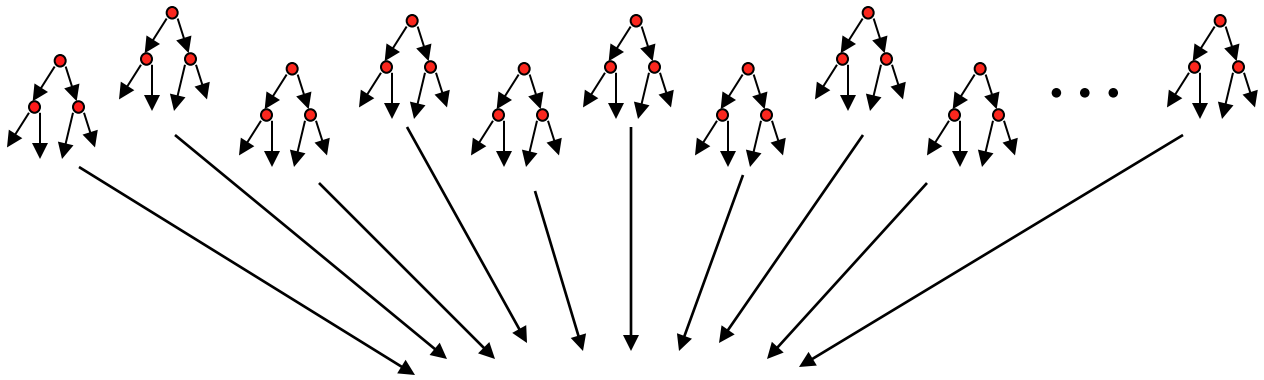
Short for ***B**ootstrap **agg**regat**ing***

- A method for generating multiple versions of a predictor and using these to get an aggregated predictor.
  - Averages over the versions when predicting a numerical outcome (regression)
  - Does a plurality vote when predicting a class (classification)

- Instability of the prediction method: If perturbing the training set can cause significant changes in the learned classifier *then* bagging can improve accuracy

Example: Bagged Decision Trees

- Draw T bootstrap samples of data
- Train trees on each sample ⇒ T trees
- Average prediction of trees on out-of-bag samples



Average prediction

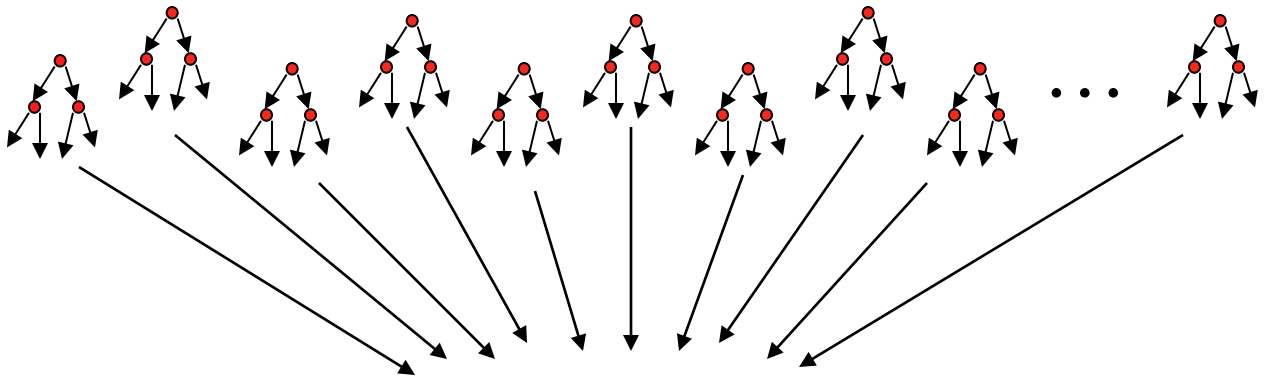
(0.23 + 0.19 + 0.34 + 0.22 + 0.26 + … + 0.31) / # Trees = 0.24

Vote prediction

Major(T, F, T, T, F, T, T, T) = True

Random Forests (Bagged Trees++)

- Draw T (possibly ***1000s***) bootstrap samples of data
- ***Draw sample of available attributes at each split***
- Train trees on each sample/attribute set ! T trees
- Average prediction of trees on out-of-bag samples



Average prediction

(0.23 + 0.19 + 0.34 + 0.22 + 0.26 + … + 0.31) / # Trees = 0.24

Vote prediction

Major(T, F, T, T, F, T, T, T) = True

Boosting and Ensembles

- Ensemble methods
- Boosting AdaBoost

One boosting approach

- Select a set of examples
- Derive a rough rule of thumb
- Example a second set of examples
- Derive a second rule of thumb
- Repeat T times…
- Combine rules of thumb into a single prediction rule

One boosting approach

- Select a small subset of examples
- Derive a rough rule of thumb
- Example a second set of examples
- Derive a second rule of thumb
- Repeat T times…
- Combine rules of thumb into a single prediction rule

Need to specify:
How to select these subsets?

Need to specify:
How to combine these rules of thumb?

rule of thumb – weak classifier

One boosting approach

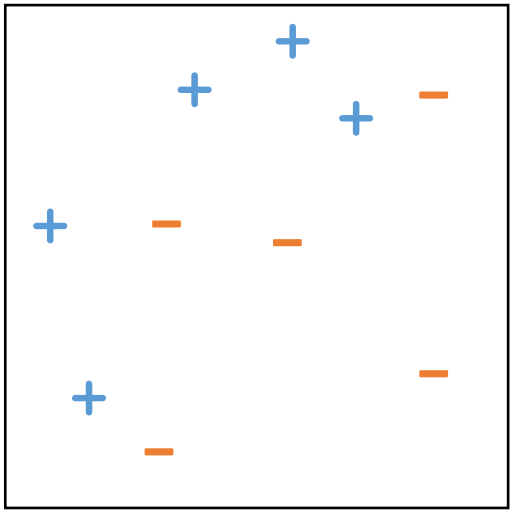
- Select a small subset of examples
- Derive a rough rule of thumb
- Example a second set of examples
- Derive a second rule of thumb
- Repeat T times…
- Combine rules of thumb into a single prediction rule

Boosting: A general method for converting rough rules of thumb into accurate prediction classifiers

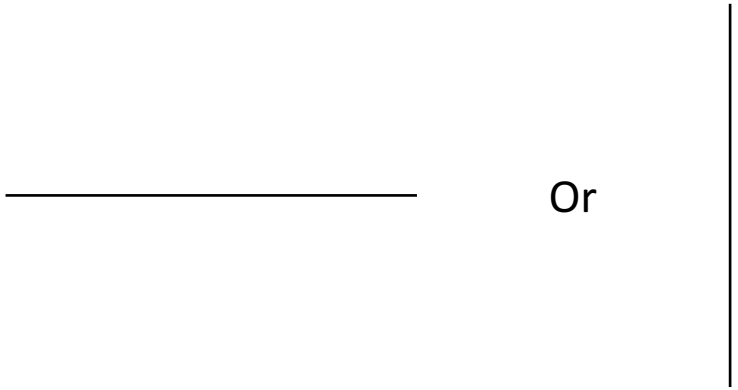
Boosting and Ensembles

- Ensemble methods
- AdaBoost
  - Intuition
  - The algorithm
  - Why does it work

A toy example

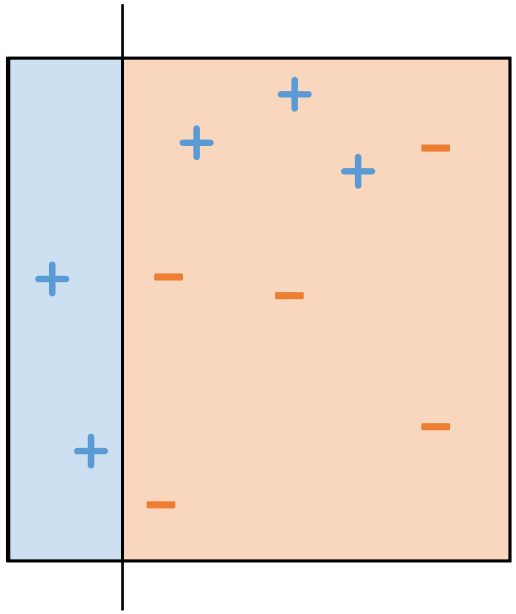


Our weak learner: An axis parallel line



Initially all examples are equally important

A toy example



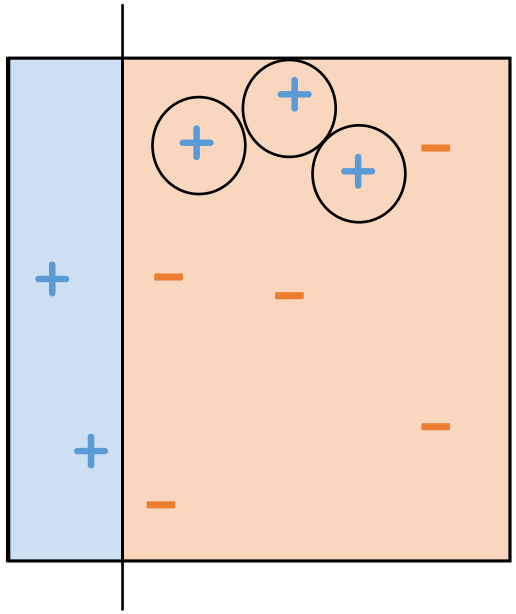
Our weak learner: An axis parallel line

Or

Initially all examples are equally important

h_1 = The best classifier on this data

A toy example



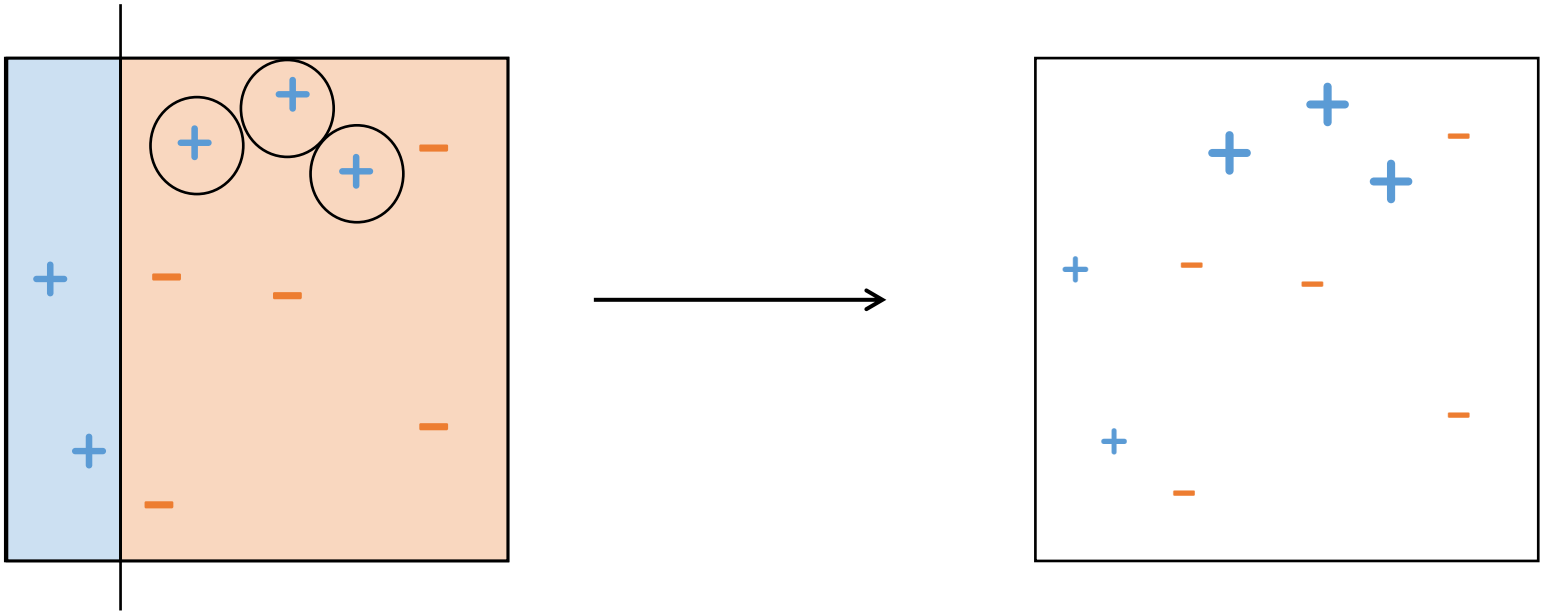
Our weak learner: An axis parallel line

Or

Initially all examples are equally important

h_1 = The best classifier on this data
Clearly there are mistakes. Error ϵ_1 = 0.3

A toy example

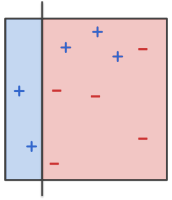


Initially all examples are equally important

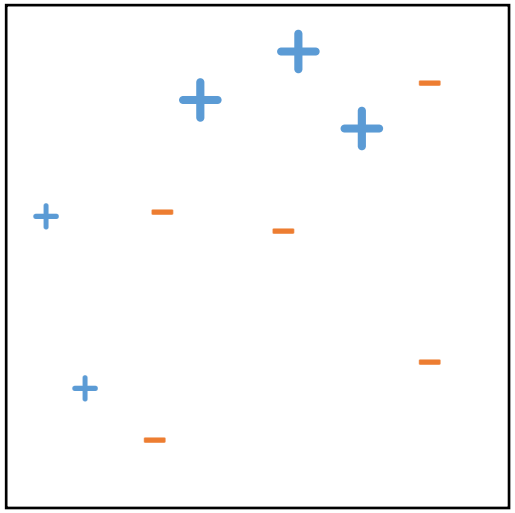
h_1 = The best classifier on this data
Clearly there are mistakes. Error ϵ_2 = 0.3

For the next round, increase the importance of the examples with mistakes and down-weight the examples that h_1 got correctly

A toy example

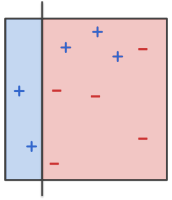


h_1

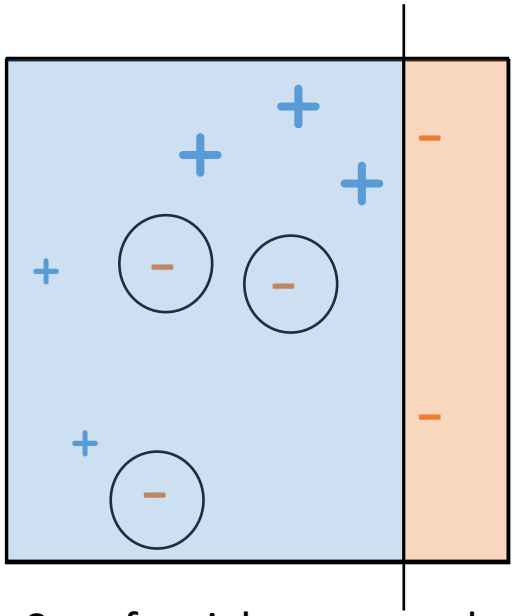


D_t = Set of weights at round t, one for each example. Think "How much should the weak learner care about this example in its choice of the classifier?"

A toy example



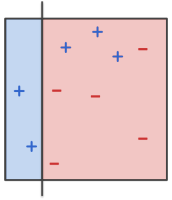
h_1



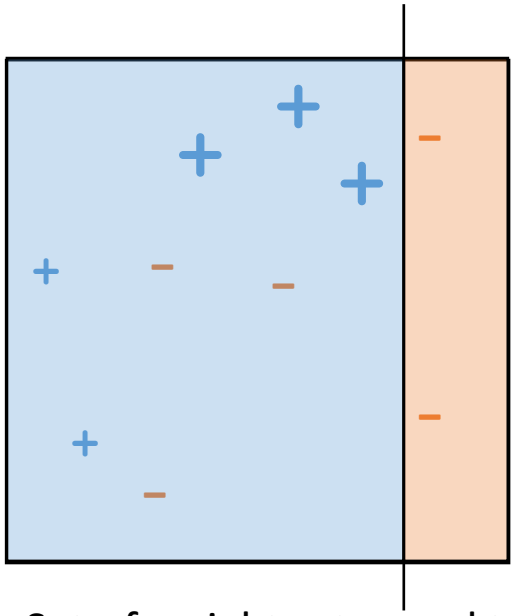
D_t = Set of weights at round t, one for each example. Think "How much should the weak learner care about this example in its choice of the classifier?"

h_2 = A classifier learned on this data. *Has an error* $\epsilon_2 = 0.21$

A toy example

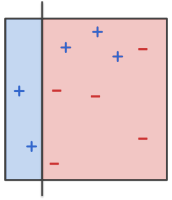


h_1

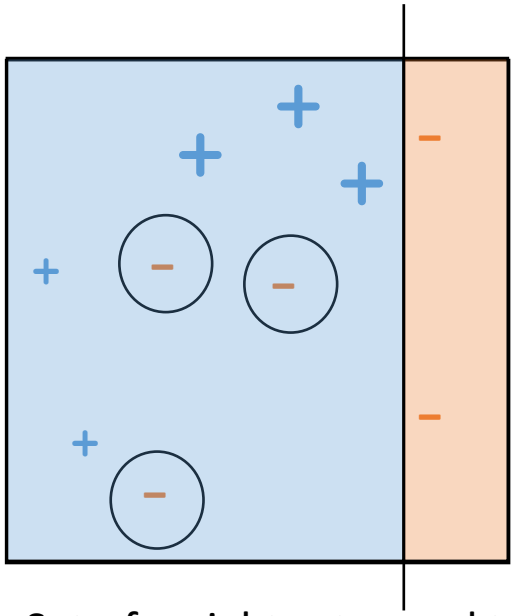


D_t = Set of weights at round t, one for each example. Think "How much should the weak learner care about this example in its choice of the classifier?"

A toy example



h_1

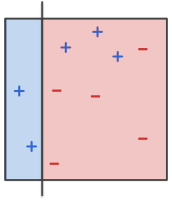


D_t = Set of weights at round t, one for each example. Think "How much should the weak learner care about this example in its choice of the classifier?"

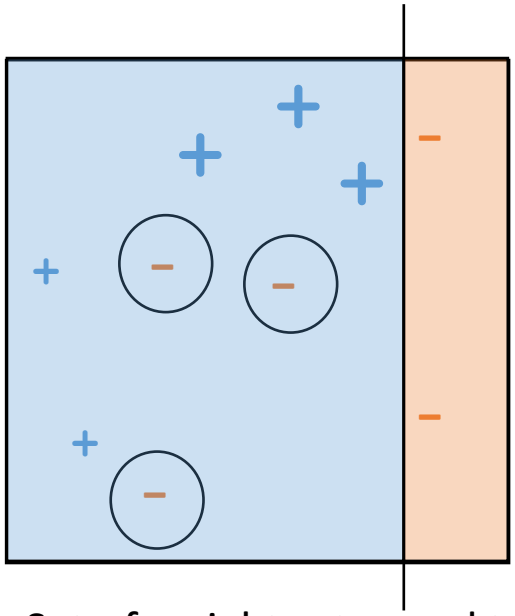
h_2 = A classifier learned on this data. *Has an error* $\epsilon_2 = 0.21$

Why not 0.3? Because while computing error, we will weight each example x_i by its $D_t(i)$

A toy example



h_1



$$\epsilon_t = \frac{1}{2} - \frac{1}{2}\left(\sum_{i=1}^{m} D_t(i)\ y_i h(x_i)\right)$$

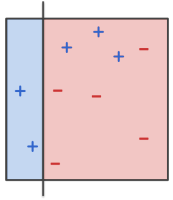
Why is this a reasonable definition?

D_t = Set of weights at round t, one for each example. Think "How much should the weak learner care about this example in its choice of the classifier?"

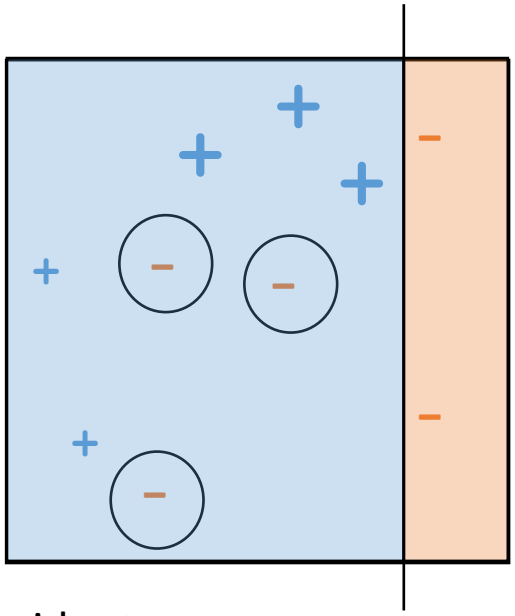
h_2 = A classifier learned on this data. *Has an error* $\epsilon_2 = 0.21$

Why not 0.3? Because while computing error, we will weight each example x_i by its $D_t(i)$

A toy example



h_1



$$\epsilon_t = \frac{1}{2} - \frac{1}{2}\left(\sum_{i=1}^{m} D_t(i)\ y_i h(x_i)\right)$$

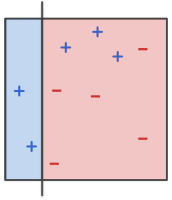
Why is this a reasonable definition?

Consider two cases

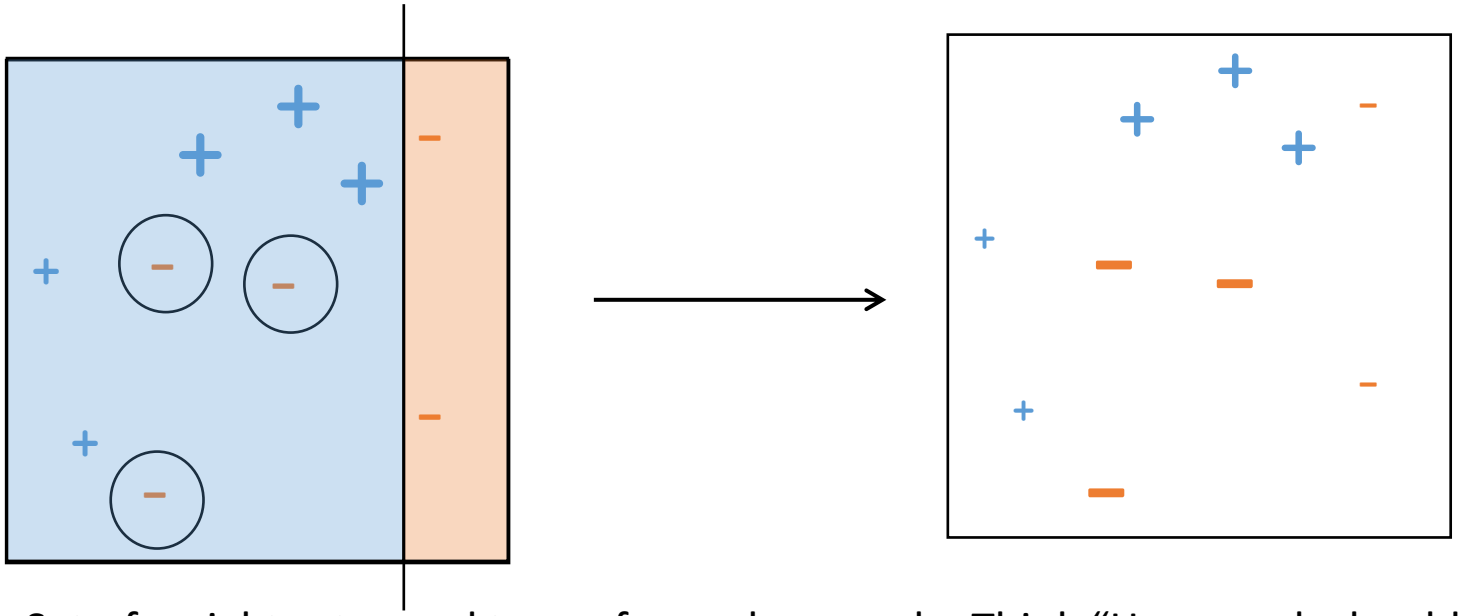
Case 1: When $y = +1$, $h(x) = -1$ OR $y = -1$, $h(x) = +1$

Case 2: When $y = h(x)$

A toy example



h_1

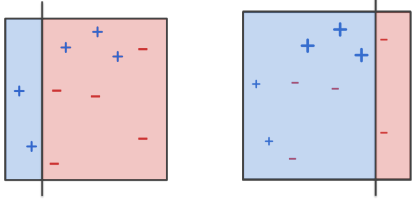


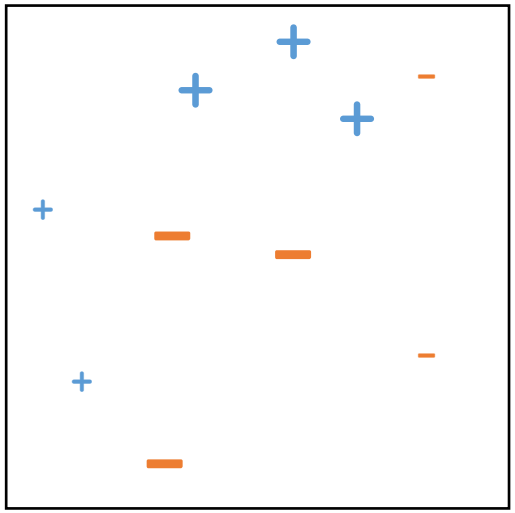
D_t = Set of weights at round t, one for each example. Think "How much should the weak learner care about this example in its choice of the classifier?"

h_2 = A classifier learned on this data. *Has an error* $\epsilon_2 = 0.21$

For the next round, increase the importance of the mistakes and down-weight the examples that h_2 got correctly

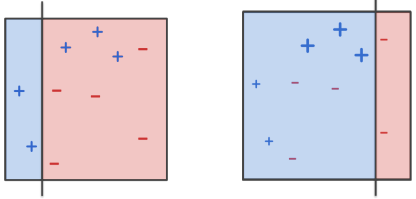
A toy example



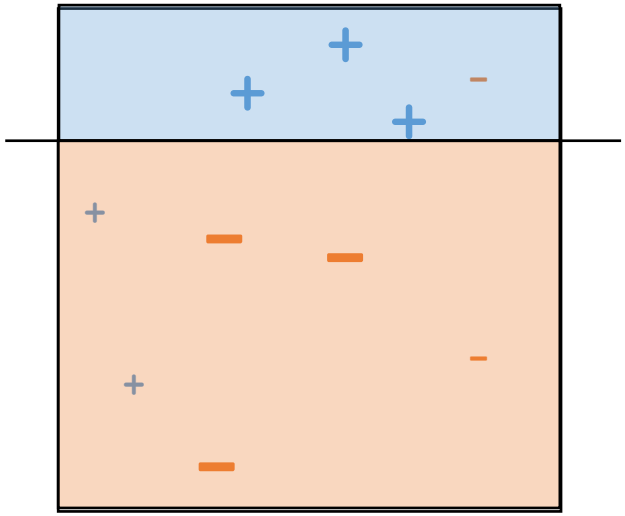


D_t = Set of weights at round t, one for each example. Think "How much should the weak learner care about this example in its choice of the classifier?"

A toy example



h_1 h_2

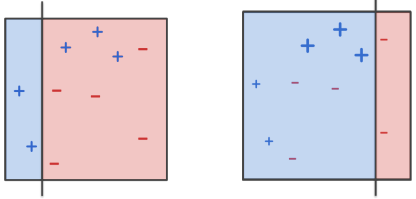


$$\epsilon_t = \frac{1}{2} - \frac{1}{2}\left(\sum_{i=1}^{m} D_t(i)\ y_i h(x_i)\right)$$

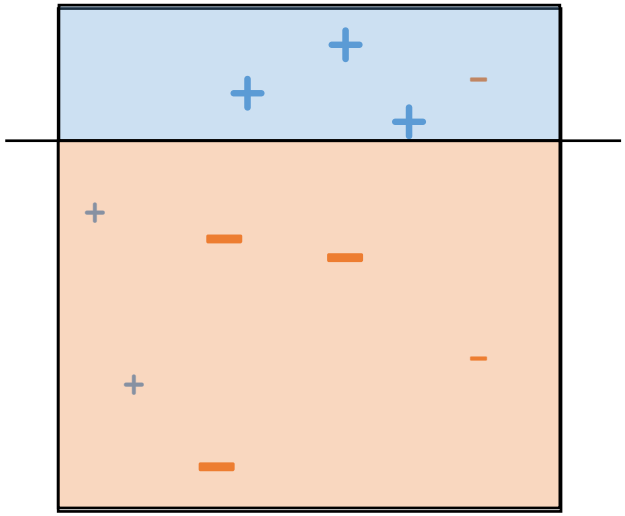
D_t = Set of weights at round t, one for each example. Think "How much should the weak learner care about this example in its choice of the classifier?"

h_3 = A classifier learned on this data. *Has an error* $\epsilon_3 = 0.14$

A toy example



h_1 h_2



$$\epsilon_t = \frac{1}{2} - \frac{1}{2}\left(\sum_{i=1}^{m} D_t(i)\ y_i h(x_i)\right)$$

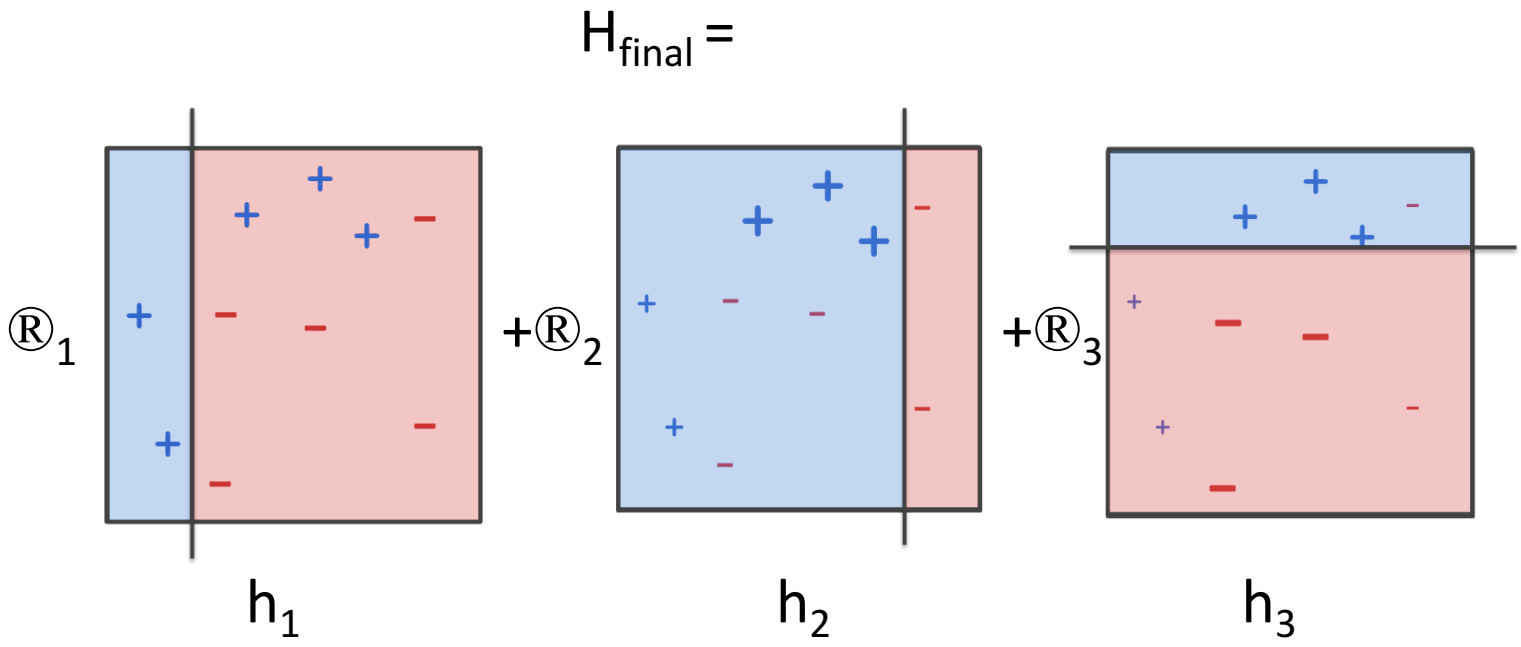
D_t = Set of weights at round t, one for each example. Think "How much should the weak learner care about this example in its choice of the classifier?"

h_3 = A classifier learned on this data. *Has an error* $\epsilon_3 = 0.14$

Why not 0.3? Because while computing error, we will weight each example x_i by its $D_t(i)$

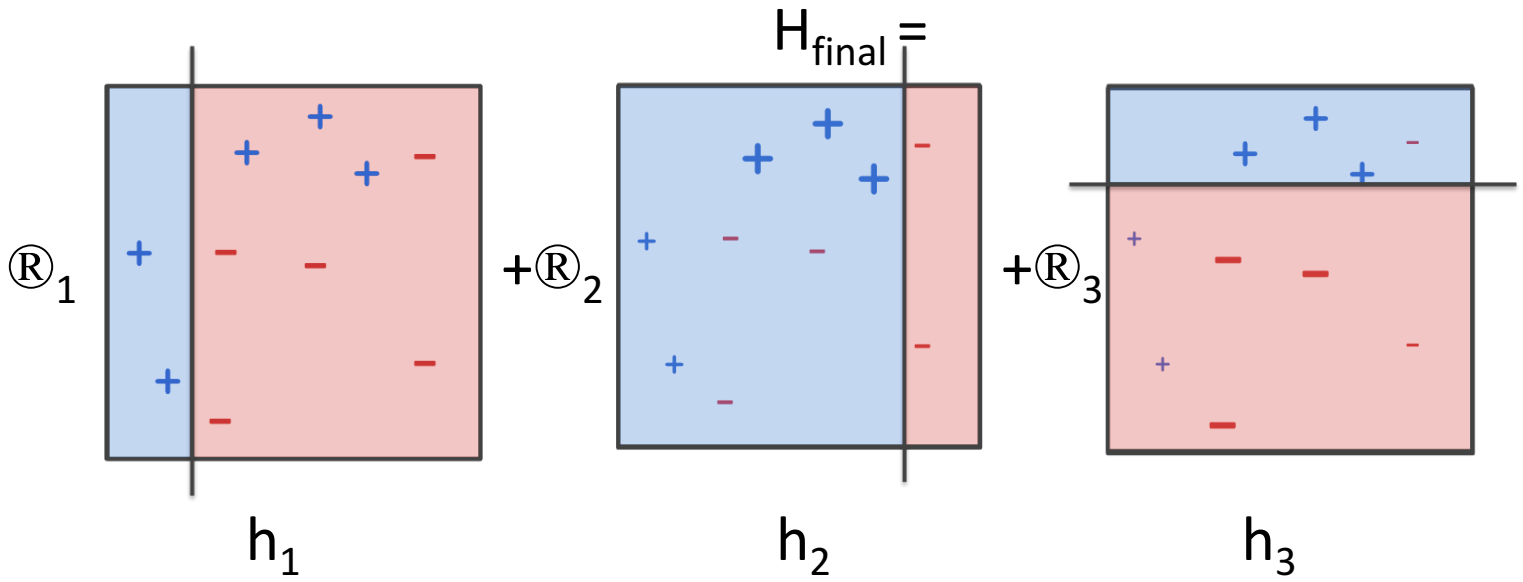
A toy example

The final hypothesis is a combination of all the h_i's we have seen so far



A toy example

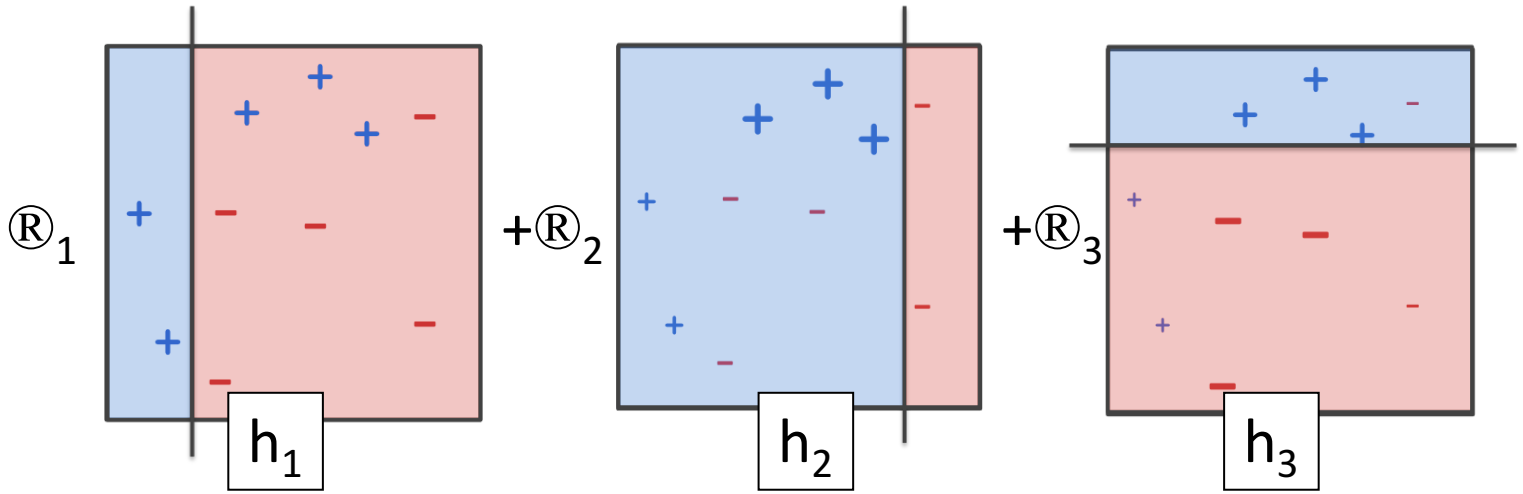
The final hypothesis is a combination of all the h_i's we have seen so far

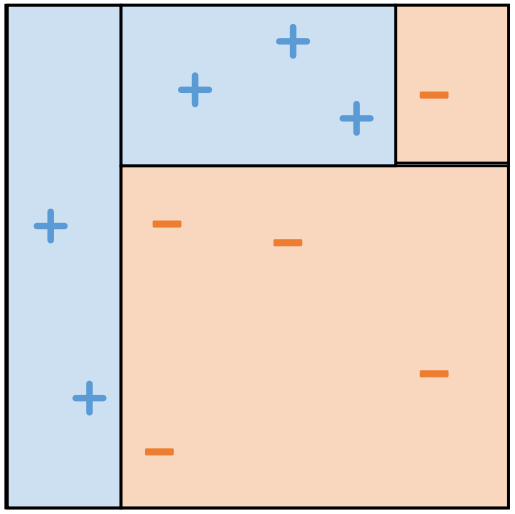


Think of the ® values as the vote for each weak classifier and the boosting algorithm has to somehow specify them

A toy example: final classifier:

$H_{final} =$





An outline of Boosting

How to?

Given a training set $(x_1, y_1), \cdots, (x_m, y_m)$

- Instances $x_i \in X$ labeled with $y_i \in \{-1, +1\}$

- For t = 1, 2, ⋯, T:
  - Construct a distribution D_t on {1, 2, ⋯, m}
  - Find a weak hypothesis (rule of thumb) h_t such that it has a small *weighted* error ϵ_t on D_t

How to?

How to?

- Construct a final output H_{final}

How to?

AdaBoost: Constructing D_t

We have m examples

D_t is a set of weights over the examples at round t

$$D_t(1), D_t(2), \cdots D_t(m)$$

At every round, the weak learner looks for hypotheses h_t that emphasizes examples that have a higher D_t

AdaBoost: Constructing D_t

Initially (t = 1), use the uniform distribution over all examples

$$D_1(i) = \frac{1}{m}$$

AdaBoost: Constructing D_t

Initially (t = 1), use the uniform distribution over all examples

$$D_1(i) = \frac{1}{m}$$

After t rounds

- What we have
  - D_t and the hypothesis h_t that was learned
  - The error ϵ_t of that hypothesis on the training data

- (Intuition) What we want from the $(t+1)^{th}$ round
  - Find a hypothesis so that examples that were incorrect in the previous round are correctly predicted by the new one
  - That is, increase the importance of misclassified examples and decrease the importance of correctly predicted ones

AdaBoost: Constructing D_t

Initially (t = 1), use the uniform distribution over all examples

$$D_1(i) = \frac{1}{m}$$

After t rounds, we have some D_t and a hypothesis h_t that the weak learner produced (choose some $\alpha_t > 0$)

Create D_{t+1} as follows:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

$$= \frac{D_t(i)}{Z_t} \cdot \exp\left(-\alpha_t \cdot y_i h_t(x_i)\right)$$

Demote correctly predicted examples

Promote incorrectly predicted examples

AdaBoost: Constructing D_t

After t rounds, we have some D_t and a hypothesis h_t that the weak learner produced

Create D_{t+1} as follows:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

$$= \frac{D_t(i)}{Z_t} \cdot \exp\left(-\alpha_t \cdot y_i h_t(x_i)\right)$$

Z_t: A normalization constant. Ensures that the weights D_{t+1} add up to 1

AdaBoost: Constructing D_t

After t rounds, we have some D_t and a hypothesis h_t that the weak learner produced

Create D_{t+1} as follows:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

$$= \frac{D_t(i)}{Z_t} \cdot \exp\left(-\alpha_t \cdot y_i h_t(x_i)\right)$$

Our magical choice of α_t

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

Show that $\epsilon_t > 0.5 \Rightarrow \alpha_t > 0$

An outline of Boosting

How to?

Given a training set $(x_1, y_1), \cdots, (x_m, y_m)$

- Instances $x_i \in X$ labeled with $y_i \in \{-1, +1\}$

- For t = 1, 2, ⋯, T:
  - Construct a distribution D_t on {1, 2, ⋯, m}
  - Find a weak hypothesis (rule of thumb) h_t such that it has a small *weighted* error ϵ_t on D_t

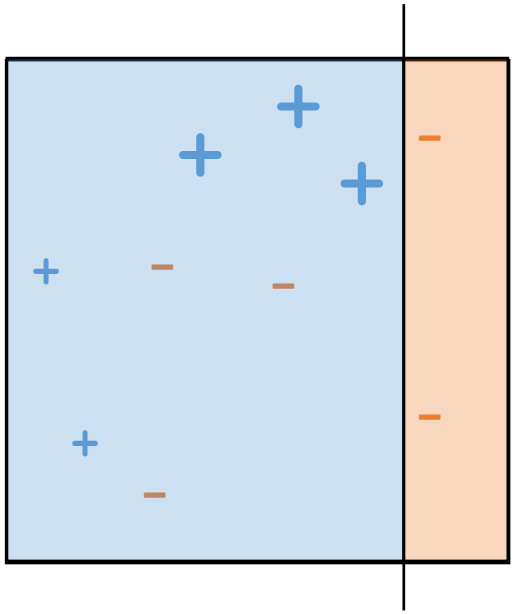
✓

If the classifier is simple, this is often easy

- Construct a final output H_{final}

How to?

Example (Decision Stumps



We can simply enumerate all possible ways of putting vertical and horizontal lines to separate the data points into two classes and nd the one with the smallest weighted classfication error!

This efficiency of finding weak classifier is an attractive quality of boosting

An outline of Boosting

How to?

Given a training set $(x_1, y_1), \cdots, (x_m, y_m)$

- Instances $x_i \in X$ labeled with $y_i \in \{-1, +1\}$

- For t = 1, 2, ⋯, T:
  - Construct a distribution D_t on {1, 2, ⋯, m}
  - Find a weak hypothesis (rule of thumb) h_t such that it has a small *weighted* error ϵ_t on D_t

✓

✓

- Construct a final output H_{final}

How to?

The final hypothesis

- After T rounds, we have
  - T weak classifiers h_1, h_2, $\cdots$ h_T
  - T values of $®_t$

- Recall that each weak classifier is takes an example $\mathbf{x}$ and produces a -1 or a +1
- Define the final hypothesis $\mathrm{H}_{\mathrm{final}}$ as

$$H_{final}(x) = \mathrm{sgn}\left(\sum_t \alpha_t h_t(x)\right)$$

AdaBoost: The full algorithm

Given a training set $(x_1, y_1), \cdots, (x_m, y_m)$

Instances $x_i \in X$ labeled with $y_i \in \{-1, +1\}$

T: a parameter to the learner

1. Initialize $D_1(i) = 1/m$ for all $i = 1, 2, \cdots, m$
2. For $t = 1, 2, \cdots T$:
   1. Find a classifier h_t whose *weighted classification error* is better than chance
   2. Compute its vote

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

   3. Update the values of the weights for the training examples

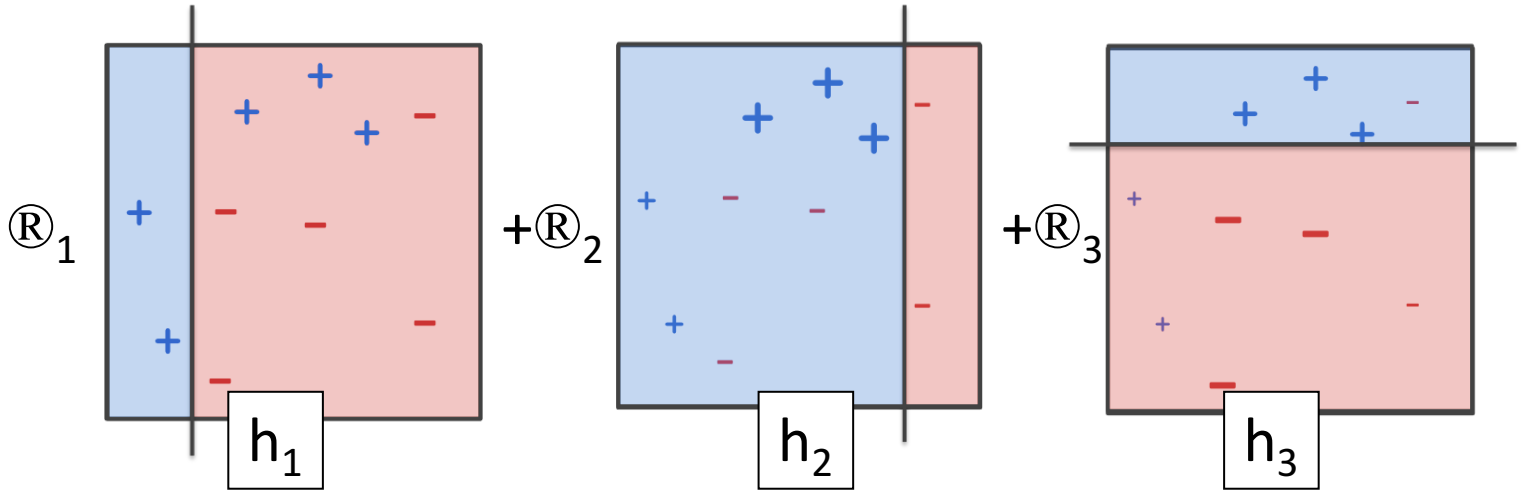
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \exp\left(-\alpha_t \cdot y_i h_t(x_i)\right)$$

3. Return the final hypothesis

$$H_{final}(x) = \operatorname{sgn}\left(\sum_t \alpha_t h_t(x) \right)$$

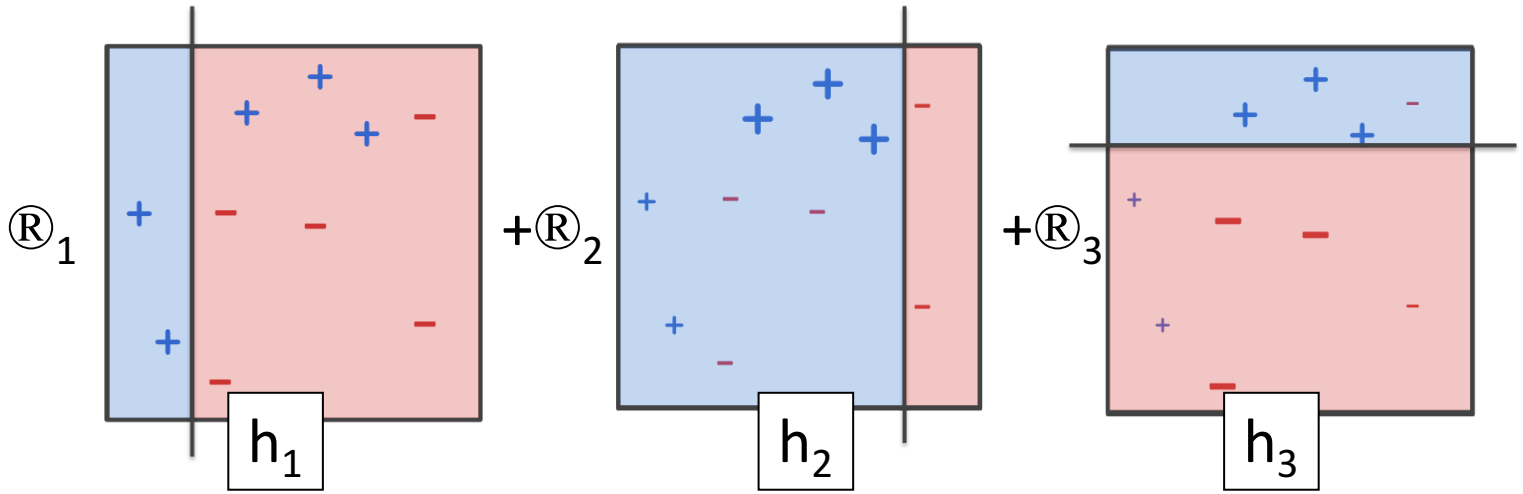
Back to the toy example

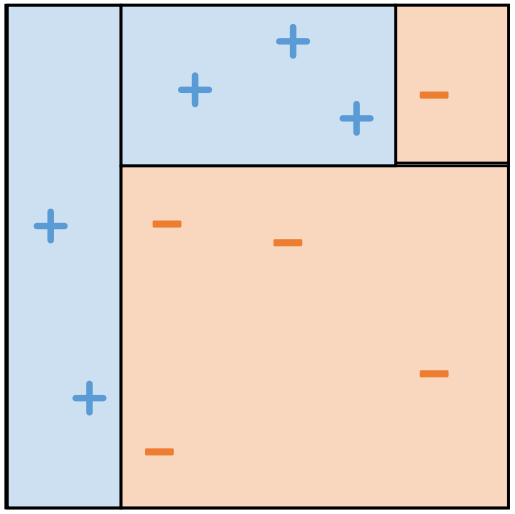
$H_{final} =$



Back to the toy example

$H_{final} =$





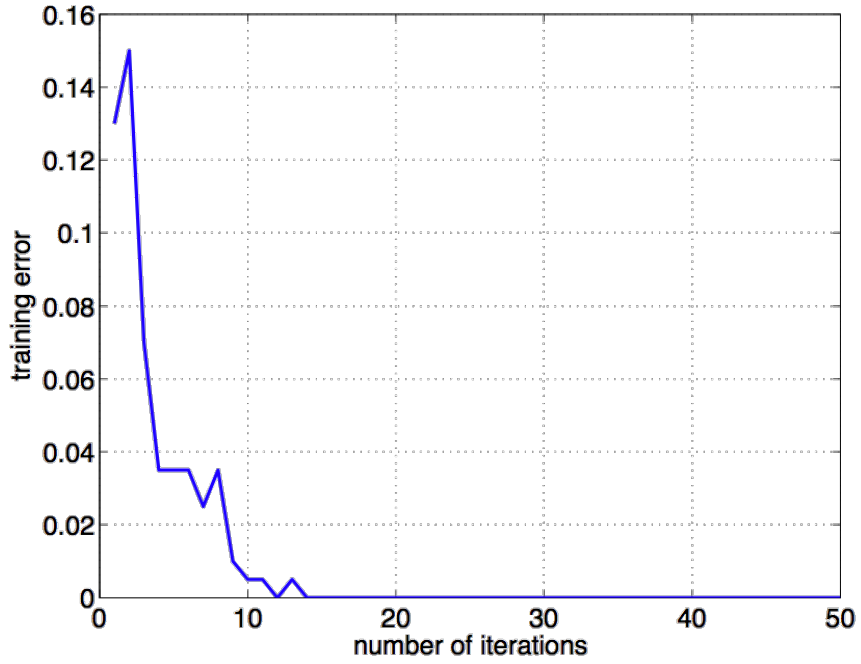
Advanced topic: (not covered in the exam) Boosting: Theoretical Motivation

- Strong PAC algorithm
  - For any distribution over examples,
  - For any $\delta > 0$, $\epsilon > 0$,
  - Given a polynomially many random examples
  - Finds a hypothesis with error ϵ with probability $1 - \delta$

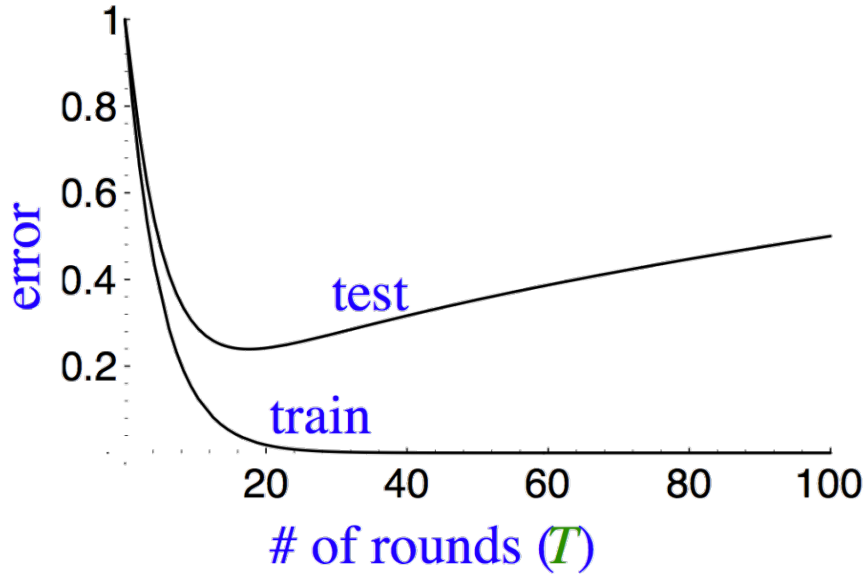
- Weak PAC algorithm
  - Same, but only for $\epsilon \sim 0.5$
- Question [Kearns and Valiant '88]:
  - Does weak learnability imply strong learnability?

Adaboost: Training error

The training error of the combined classifier decreases exponentially fast if the errors of the weak classifiers (the ϵ_t) are strictly better than chance



What about the test error?



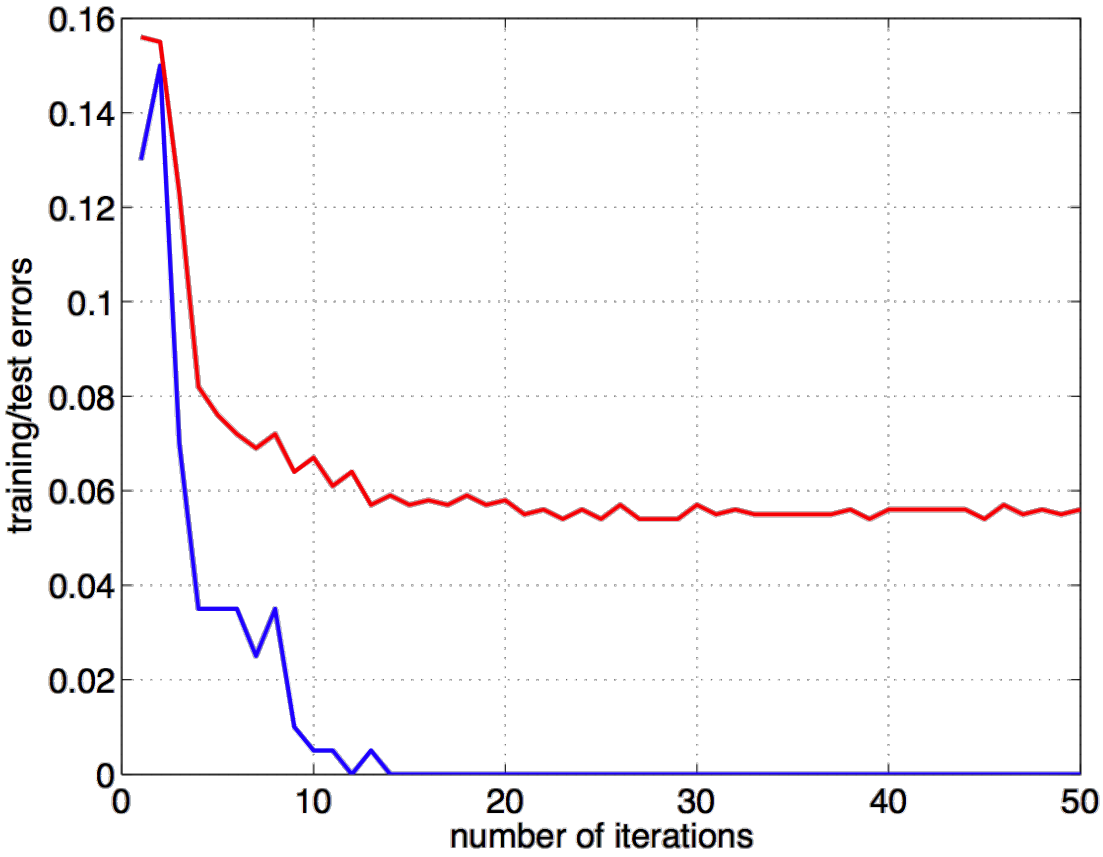
What the theory tells us:

Training error will keep decreasing or reach zero (the AdaBoost theorem)

Test error will increase after the H_{final} becomes too "complex"

- Overfitting

In practice


0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
training/test errors
0
10
20
30
40
50
number of iterations

Boosting

- Initialization:
  - Weigh all training samples equally
- Iteration Step:
  - Train model on weighted train set
  - Compute weighted error of model on train set
  - Increase weights on training cases model gets wrong
- Typically requires 100's to 1000's of iterations
- Return final model:
  - Carefully weighted prediction of each model

What you have learned

- Bagging: A simple but effective approach to combine multiple classifiers
- Boosting: Algorithm and key intuition

- More practically used algorithms:
  - Bagging+ Decision tree ⇒ Random forest
  - Gradient boosting
    https://en.wikipedia.org/wiki/Grdient_boosting