Lecture 13: Boosting & Multi-class methods

Winter 2018

Kai-Wei Chang

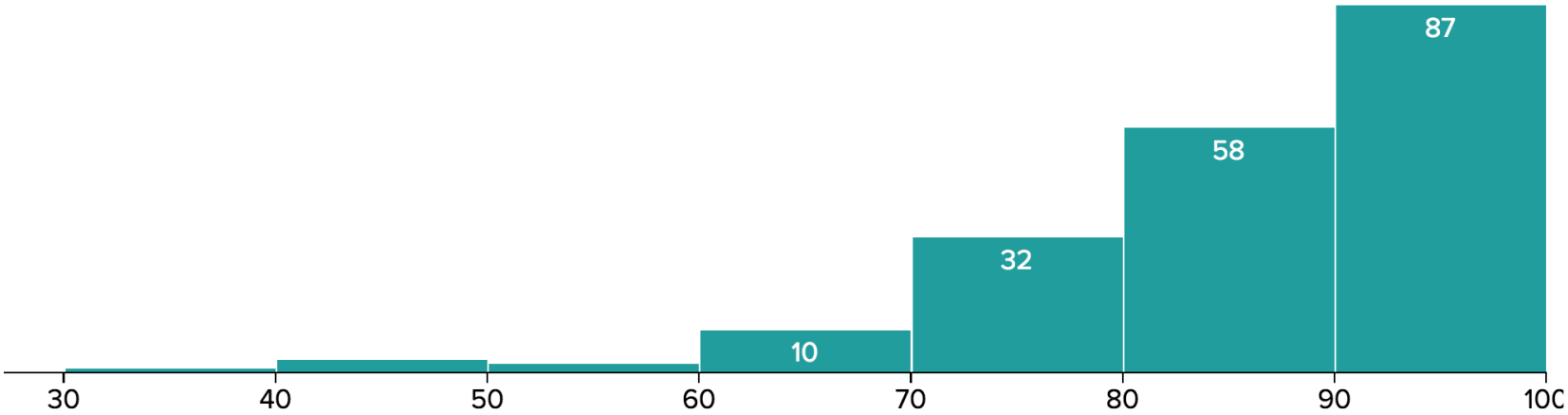
CS @ UCLA

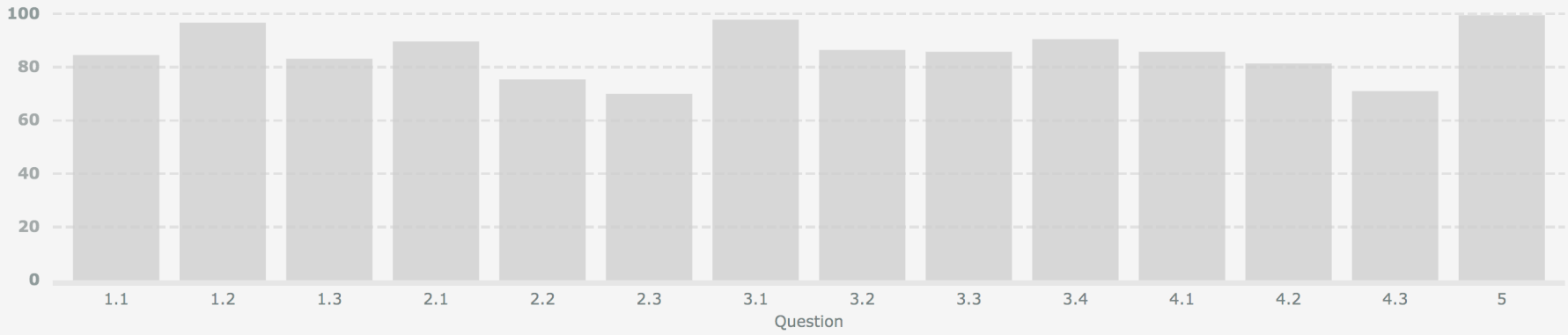
kw+cm146@kwchang.net

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.

Midterm Exam

MEDIAN	MAXIMUM	MEAN	STD DEV
88.0%	100.0% (11)	85.0%	13.13%





Feedback

1. (1) I am keeping up with the course material

- (0) Strongly Disagree:	2 (2.82 %)
- (1) Disagree:	7 (9.86 %)
- (2) Neutral:	16 (22.54 %)
- (3) Agree:	29 (40.85 %)
- (4) Strongly Agree:	17 (23.94 %)

Average: 2.73

2. (2) The level of difficulty of the material is

- (0) Too Easy:	1 (1.41 %)
- (1) Easy:	4 (5.63 %)
- (2) Okay:	28 (39.44 %)
- (3) Difficult:	33 (46.48 %)
- (4) Too Difficult:	5 (7.04 %)

Average: 2.52

3. (3) The lectures help me understand the course concept

- (0) Not at all:	3 (4.23 %)
- (1) Sometimes:	14 (19.72 %)
- (2) Usually:	17 (23.94 %)
- (3) Mostly:	28 (39.44 %)
- (4) Always:	9 (12.68 %)

Average: 2.37

4. (4) The assignments help me understand the course concepts

- (0) Not at all:	3 (4.23 %)
- (1) Sometimes:	15 (21.13 %)
- (2) Usually:	14 (19.72 %)
- (3) Mostly:	24 (33.80 %)
- (4) Always:	15 (21.13 %)

Average: 2.46

5. (5) The pace of the course is

- (0) Too Slow:	0
- (1) Slow:	10 (14.08 %)
- (2) Just Right:	35 (49.30 %)
- (3) Fast:	23 (32.39 %)
- (4) Too Fast:	3 (4.23 %)

Average: 2.27

6. I feel the time spent on homework problems is well-spent on learning

- (0) Strongly Disagree:	2 (2.82 %)
- (1) Disagree:	13 (18.31 %)
- (2) Neutral:	21 (29.58 %)
- (3) Agree:	27 (38.03 %)
- (4) Strongly Agree:	8 (11.27 %)

Average: 2.37

7. (7) The exam is

- (0) Too Easy:	0
- (1) Easy:	5 (7.04 %)
- (2) Just the Right Difficulty:	49 (69.01 %)
- (3) Difficult:	15 (21.13 %)
- (4) Too Difficult:	2 (2.82 %)

Average: 2.20

Kernel Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}^{m}_{i=1}$, $\boldsymbol{\phi(.)}$

1. Initialize $\boldsymbol{\alpha} \leftarrow \mathbf{0} \in \mathbb{R}^{\boldsymbol{m}}$
2. For $(\boldsymbol{x_j}, y_j)$ in $\mathcal{D}$: $(\mathrm{j} = 1 \ldots m)$
3. if $y_j\left(\sum_{1..m} \alpha_i \, y_i K(x_i, x_j)\right) \leq \mathbf{0}$
4. $\alpha_j \leftarrow \alpha_j + 1$
5. Return $\boldsymbol{\alpha}$
6.

$$w = \sum_{1..m} \alpha_i \, y_i \phi(x_i)$$

Let $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}\left(\sum_{1..m} \alpha_i \, y_i K(x_i, x^{Test})\right)$

Example

- Kernel Perceptron for XOR data in 2D space (1: true, -1: false
- Kernel: $\mathrm{K}(\mathrm{x}, z) = (x^T z)^2$
- What is the kernel matrix?
- What are α learned from Perceptron
- What is the implicit separator

	x_1	x_2	y
I1	-1	1	1
I2	1	-1	1
I3	-1	-1	-1
I4	1	1	-1

	I1	I2	I3	I4
I1	4	4	0	0
I2	4	4	0	0
I3	0	0	4	4
I4	0	0	4	4

Kernel Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}_{i=1}^{m}$, $\boldsymbol{\phi(.)}$

1. Initialize $\boldsymbol{\alpha} \leftarrow \mathbf{0} \in \mathbb{R}^{\boldsymbol{m}}$
2. For $(\boldsymbol{x_j}, y_j)$ in $\mathcal{D}$: $(\mathrm{j} = 1 \ldots m)$
3. if $y_j\left(\sum_{1..m} \alpha_i \, y_i K(x_i, x_j)\right) \leq \mathbf{0}$
4. $\alpha_j \leftarrow \alpha_j + 1$
5. Return $\boldsymbol{\alpha}$
6.

	x_1	x_2	y
I1	-1	1	1
I2	1	-1	1
I3	-1	-1	-1
I4	1	1	-1

	I1	I2	I3	I4
I1	4	4	0	0
I2	4	4	0	0
I3	0	0	4	4
I4	0	0	4	4

Let $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

$w = \sum_{1..m} \alpha_i \, y_i \phi(x_i)$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}\left(\sum_{1..m} \alpha_i \, y_i K(x_i, x^{Test})\right)$

Kernel Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}_{i=1}^{m}$, $\boldsymbol{\phi(.)}$

1. Initialize $\boldsymbol{\alpha} \leftarrow \mathbf{0} \in \mathbb{R}^{\boldsymbol{m}}$
2. For $(\boldsymbol{x_j}, y_j)$ in $\mathcal{D}$: (j= $1 \ldots m$)
3. if $y_j\left(\sum_{1..m} \alpha_i \, y_i K(x_i, x_j)\right) \leq \mathbf{0}$
4. $\alpha_j \leftarrow \alpha_j + 1$
5. Return $\boldsymbol{\alpha}$
6.

	x_1	x_2	y
I1	-1	1	1
I2	1	-1	1
I3	-1	-1	-1
I4	1	1	-1

Let $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

	I1	I2	I3	I4
I1	4	4	0	0
I2	4	4	0	0
I3	0	0	4	4
I4	0	0	4	4

$w = \sum_{1..m} \alpha_i \, y_i \phi(x_i)$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}\left(\sum_{1..m} \alpha_i \, y_i K(x_i, x^{Test})\right)$

Exercise

- Kernel Perceptron for XOR data in 2D space (1: true, -1: false
- Kernel: $\mathrm{K}(\mathrm{x}, z) = (1 + x^T z)^2$
- What is the kernel matrix?
- What are α learned from Perceptron
- What is the implicit separator

	x_1	x_2	y
I1	-1	1	1
I2	1	-1	1
I3	-1	-1	-1
I4	1	1	-1

	I1	I2	I3	I4
I1				
I2				
I3				
I4				

Recap: Dual SVM problem

- w may be infinite variables
- We can derive the Lagrangian dual problem

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \le \alpha_i \le C, i = 1, \ldots, l \\ & \mathbf{y}^T\boldsymbol{\alpha} = 0, \end{aligned}$$

where $Q_{ij} = y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ and $\mathbf{e} = [1, \ldots, 1]^T$

$$Q_{ij} = y_i y_j K(x_i, x_j)$$

	I1	I2	I3	I4
I1	4	4	0	0
I2	4	4	0	0
I3	0	0	4	4
I4	0	0	4	4

https://www.csie.ntu.edu.tw/~cjlin/talks/MLSS.pdf

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?

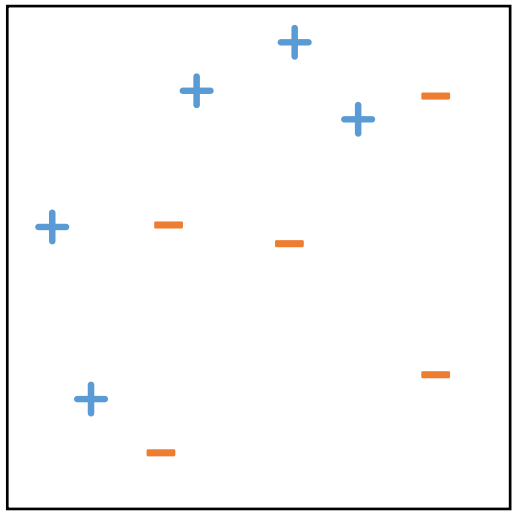
Advanced topic: (not covered in the exam) Boosting: Theoretical Motivation

- 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 $\epsilon \leq \frac{1}{2} - \gamma \quad \gamma > 0$ is a small value
- Question [Kearns and Valiant '88]:
  - Does weak learnability imply strong learnability?

A toy example

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

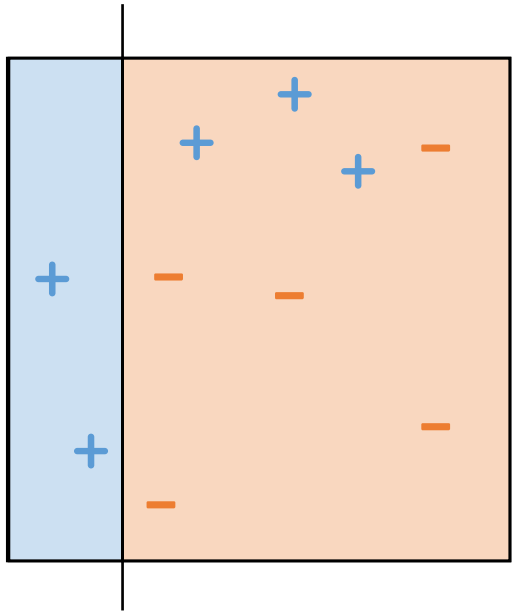


Our weak learner: An axis parallel line

Or

Initially all examples are equally important

A toy example



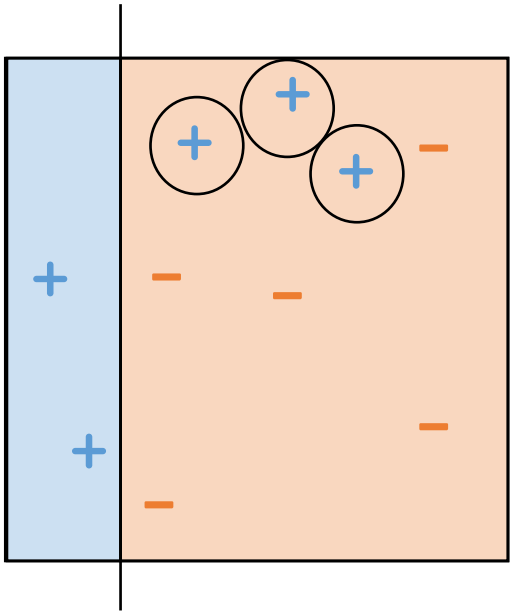
Our weak learner: An axis parallel line

Or

Initially all examples are equally important

h_1 = The best classifier on this data

A toy example



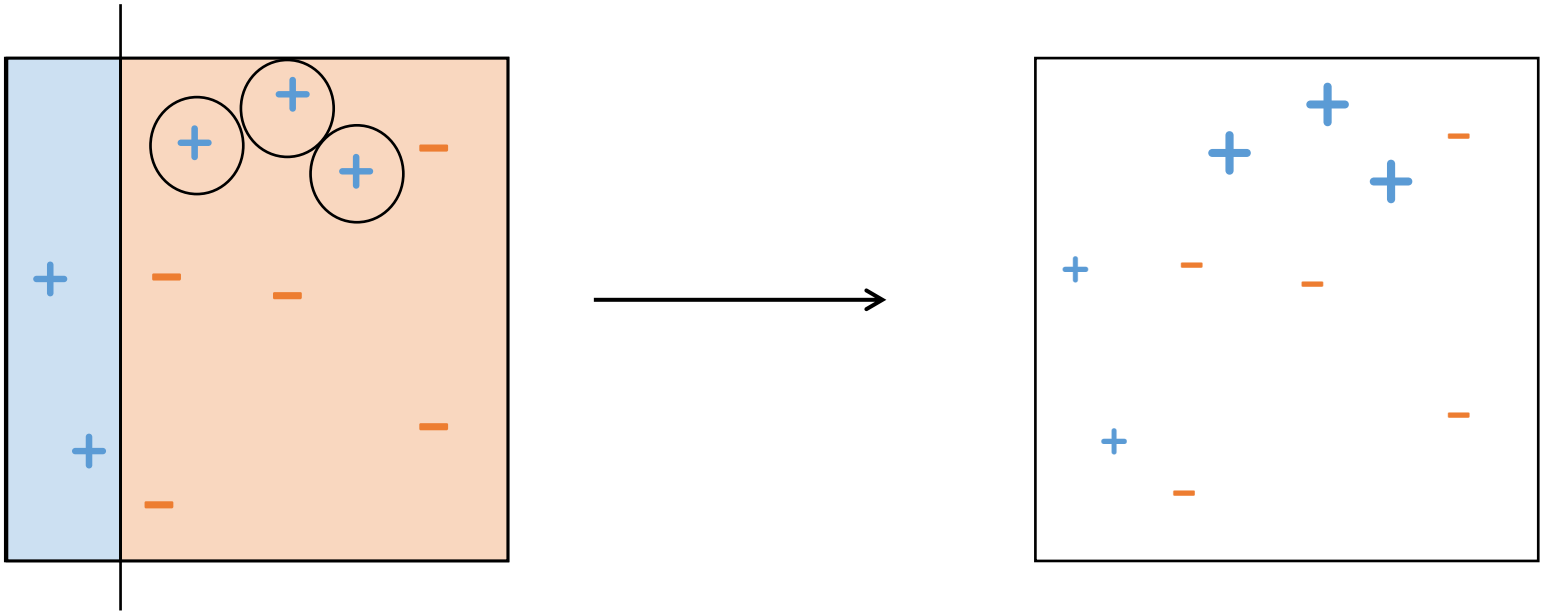
Our weak learner: An axis parallel line

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

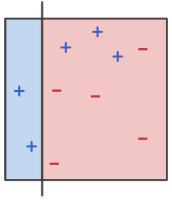
h_1 = The best classifier on this data
Clearly there are mistakes. Error ϵ_2 = 0.3

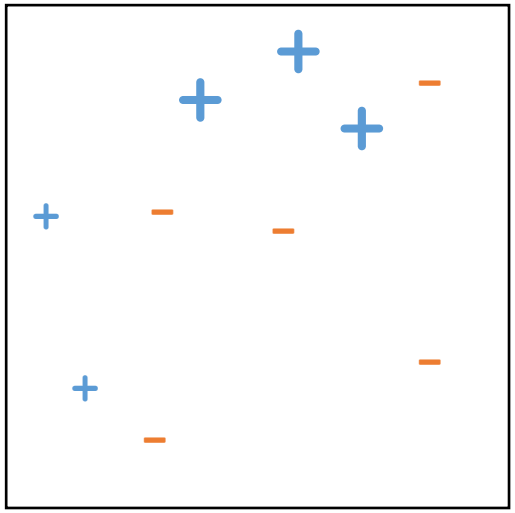
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

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

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

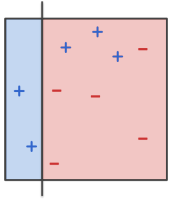
A toy example



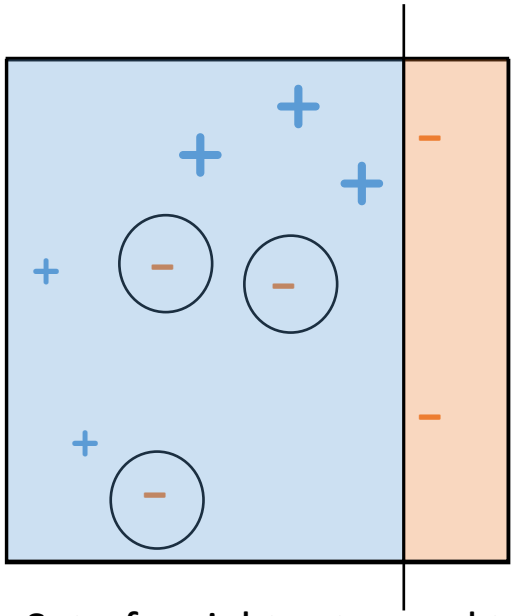


D_t = Set of weights at round t, one for each example. Think "How much should the weak learner care about this example in its choice of the classifier?"

A toy example



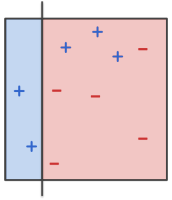
h_1



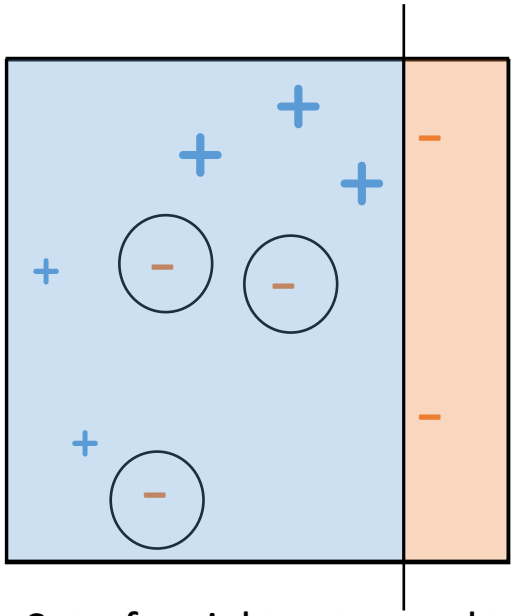
D_t = Set of weights at round t, one for each example. Think "How much should the weak learner care about this example in its choice of the classifier?"

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

A toy example



h_1



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

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

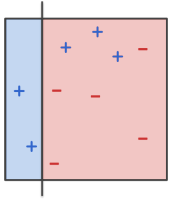
$$= \frac{D_t(i)}{Z_t} \cdot \exp\left(-\alpha_t \cdot y_i h_t(x_i)\right)$$

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?"

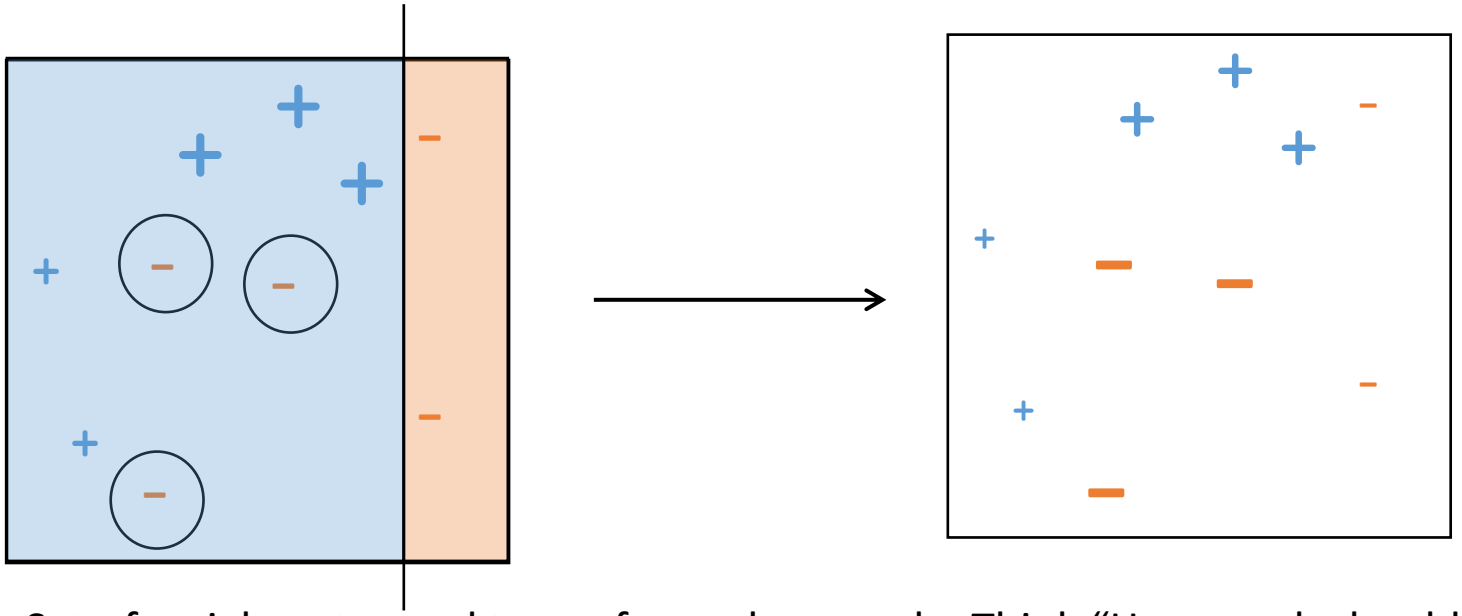
h_2 = A classifier learned on this data. *Has an error* $\epsilon_2 = 0.21$

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

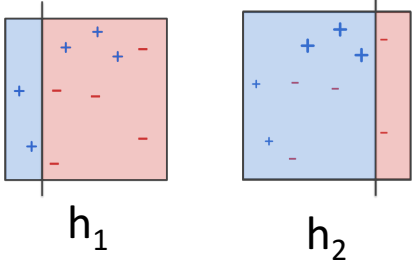


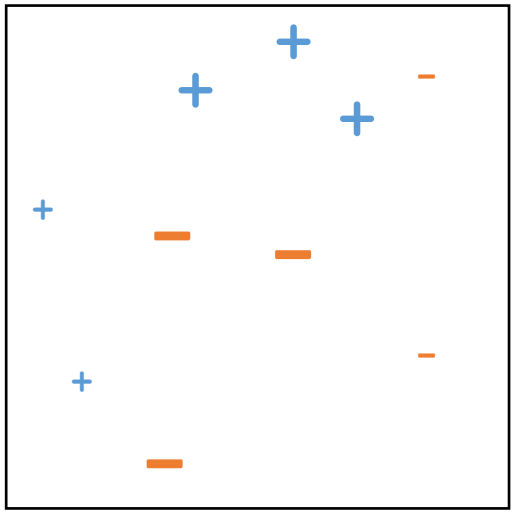
D_t = Set of weights at round t, one for each example. Think "How much should the weak learner care about this example in its choice of the classifier?"

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

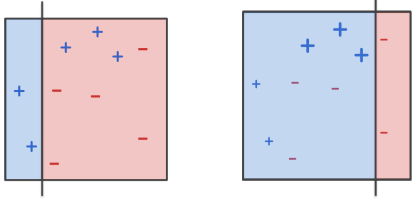
A toy example

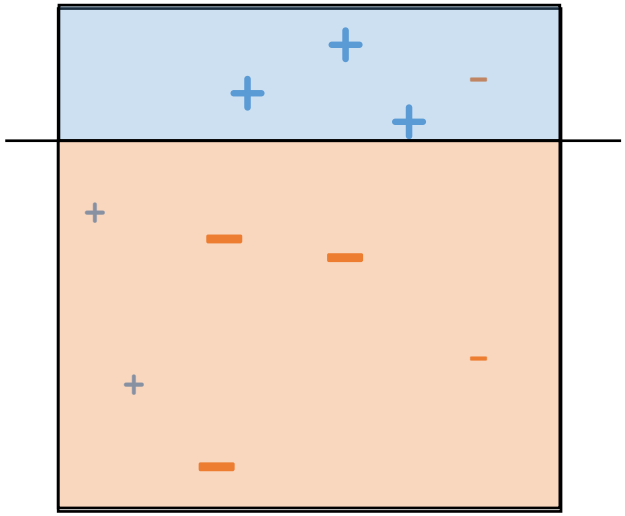




D_t = Set of weights at round t, one for each example. Think "How much should the weak learner care about this example in its choice of the classifier?"

A toy example





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 $D_t(i)$

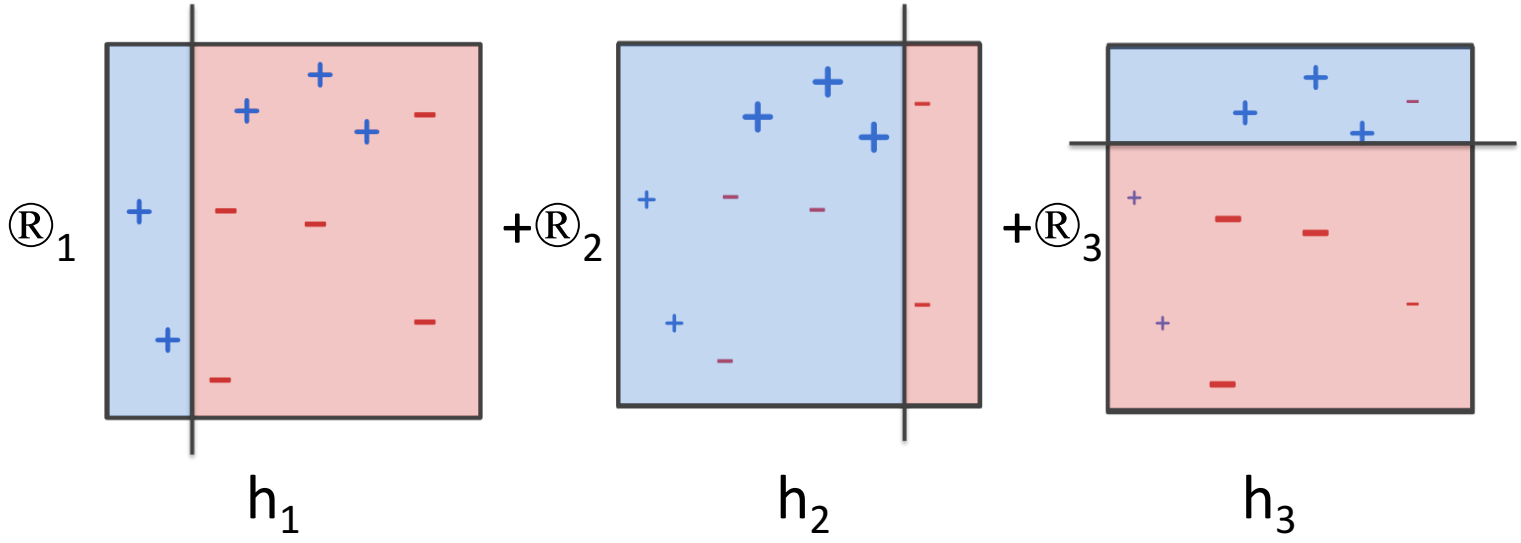
A toy example

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

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

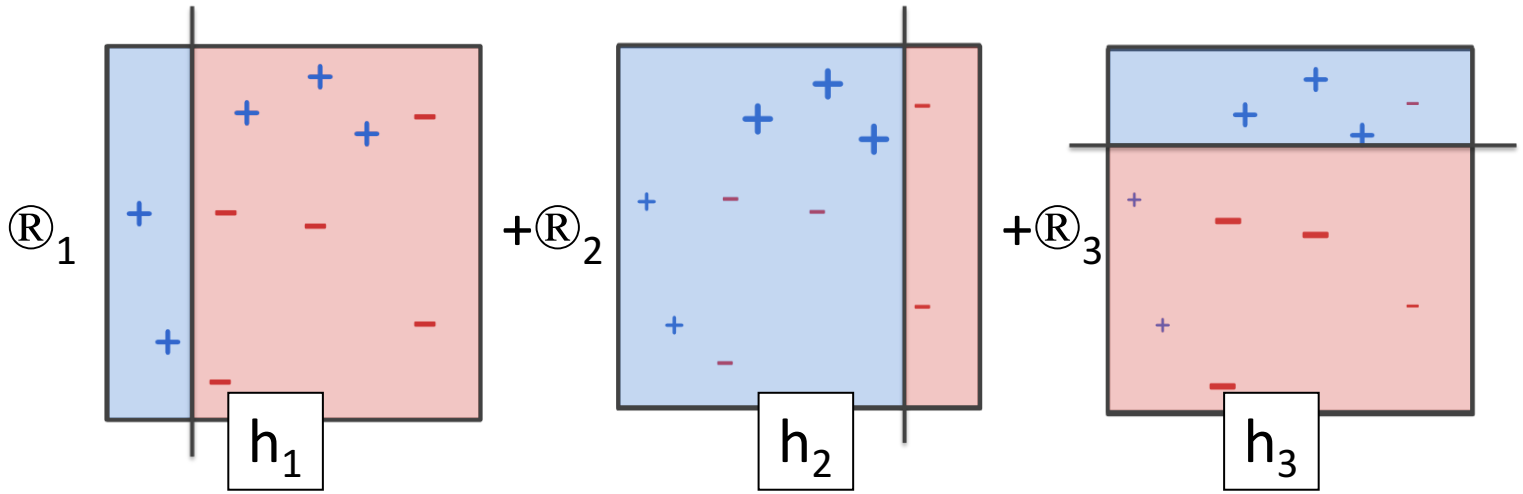
$$H_{final}(x) = \text{sgn}\left(\sum_t \alpha_t h_t(x)\right)$$

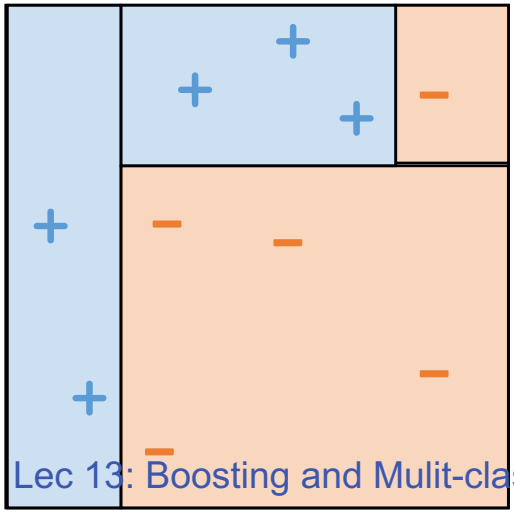
H_{final} =



A toy example: final classifier:

$H_{final} =$





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

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

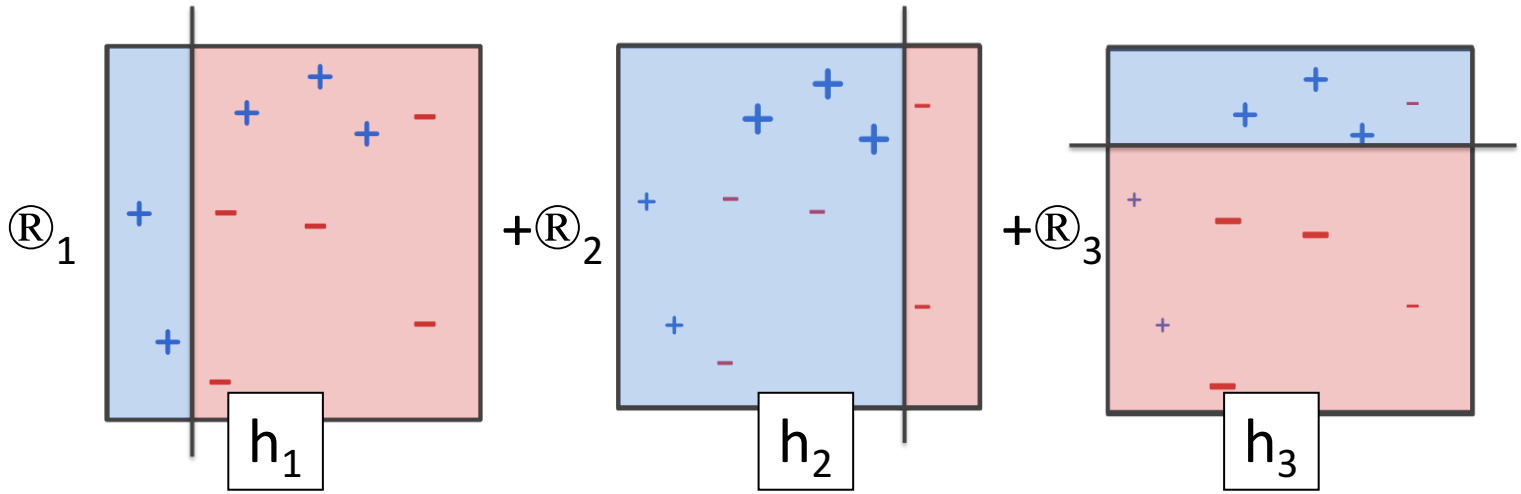
   3. Update the values of the weights for the training examples

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

3. Return the final hypothesis

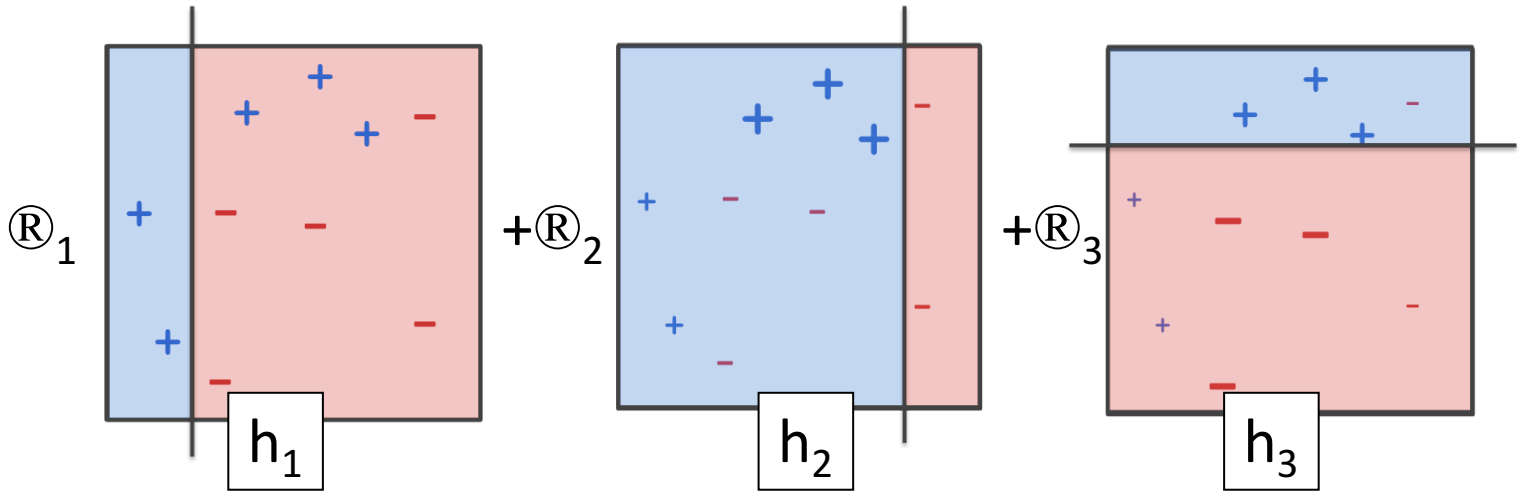
Back to the toy example

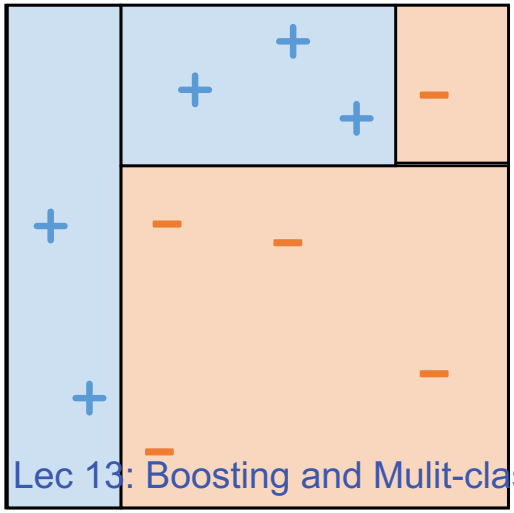
$H_{final} =$



Back to the toy example

$H_{final} =$





Analyzing the training error

Theorem:

- Run AdaBoost for T rounds
- Let ϵ_t = ½ - γ_t
- Let 0 < $\gamma \leq \gamma_t$ for all t
- Then,

$$\text{Training error}(\mathrm{H}_{\text{final}}) \leq e^{-2\gamma^2 T}$$

Analyzing the training error

Theorem:

- Run AdaBoost for T rounds
- Let $\epsilon_t = ½ - \gamma_t$
- Let $0 < \gamma \leq \gamma_t$ for all t
- Then,

We have a weak learner

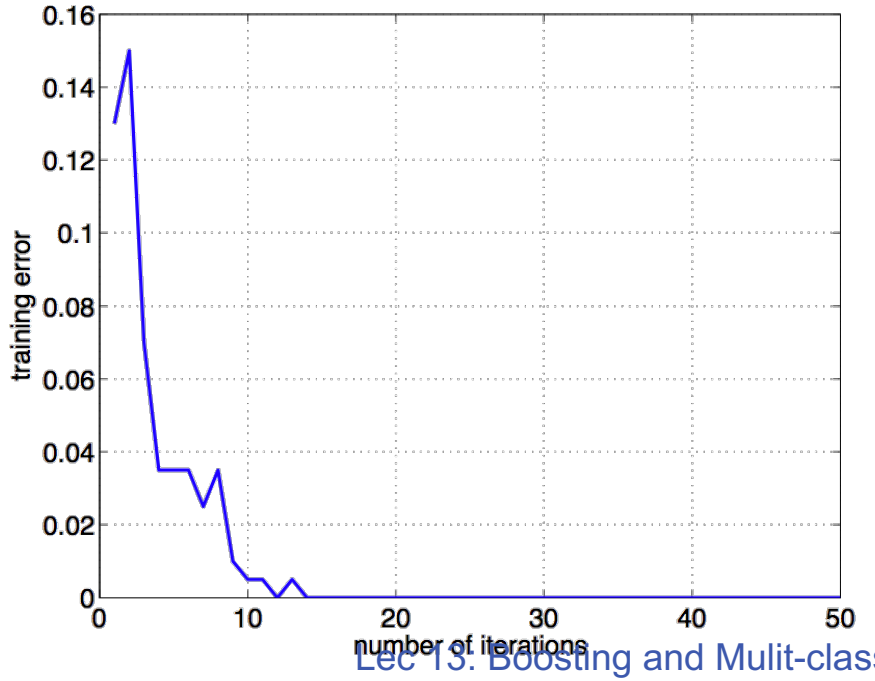
As T increases, the training error drops exponentially

$$\text{Training error}(H_{\text{final}}) \leq e^{-2\gamma^2 T}$$

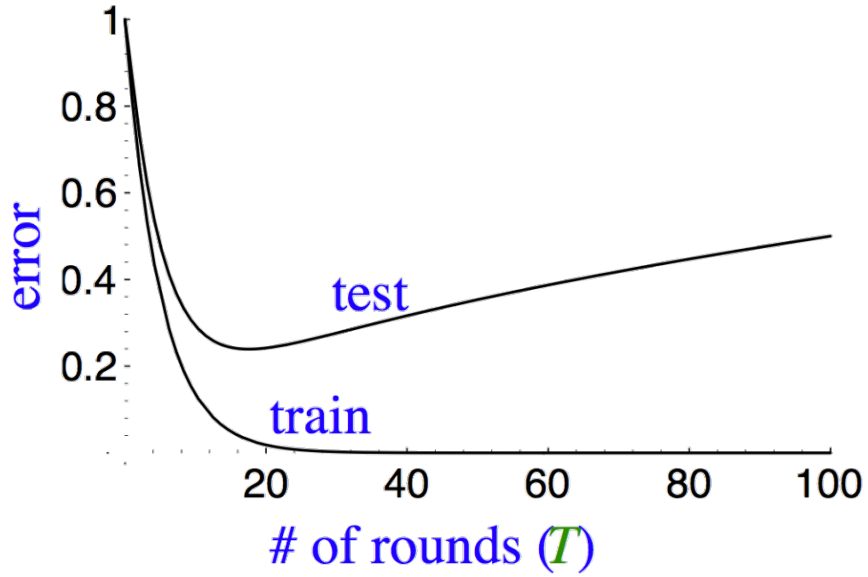
See the note at: https://svivek.com/teaching/machine-learning/fall2017/readings/boosting.pdf

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?



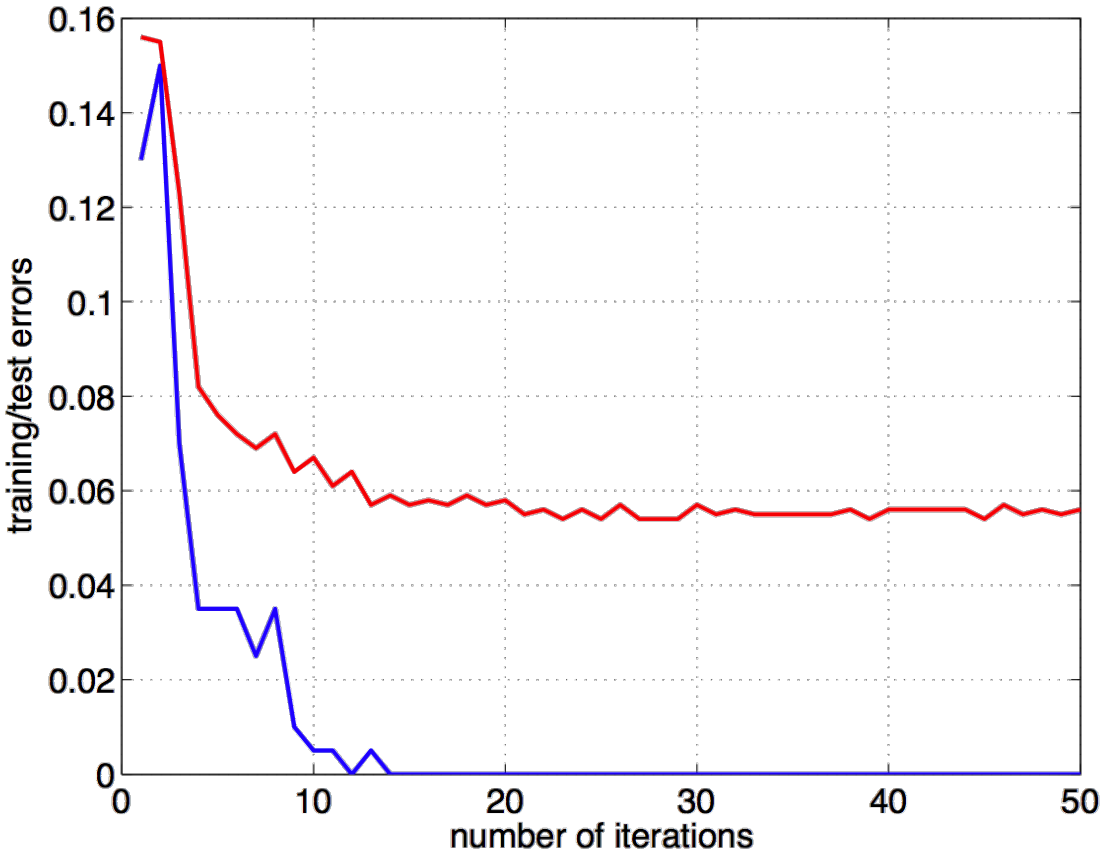
What the theory tells us:

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

Multi-class classification

What we have seen so far

- Binary linear classification models
  - Perceptron, SVMs, Logistic regression
- Prediction is simple:
  - Given an example x, prediction is $sgn(w^T \mathrm{x})$
  - Note that all these linear classifier have the same inference rule
  - In logistic regression, we can further estimate the probability

$$P(y = 1|\mathbf{x}, \mathbf{w}) = \frac{e^{\mathbf{w}^T\mathbf{x}}}{1 + e^{\mathbf{w}^T\mathbf{x}}} = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

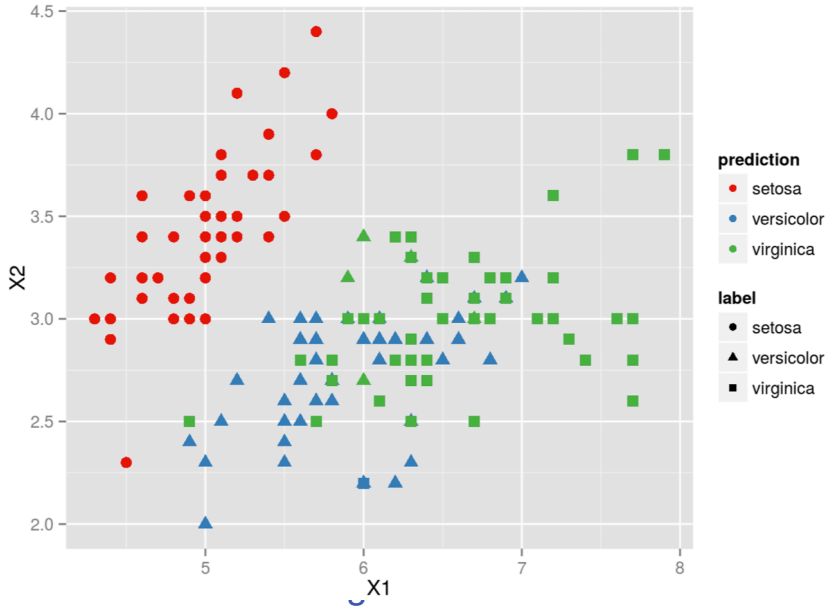
- KNN/Decision tree can easily handle multiclass

This Lecture

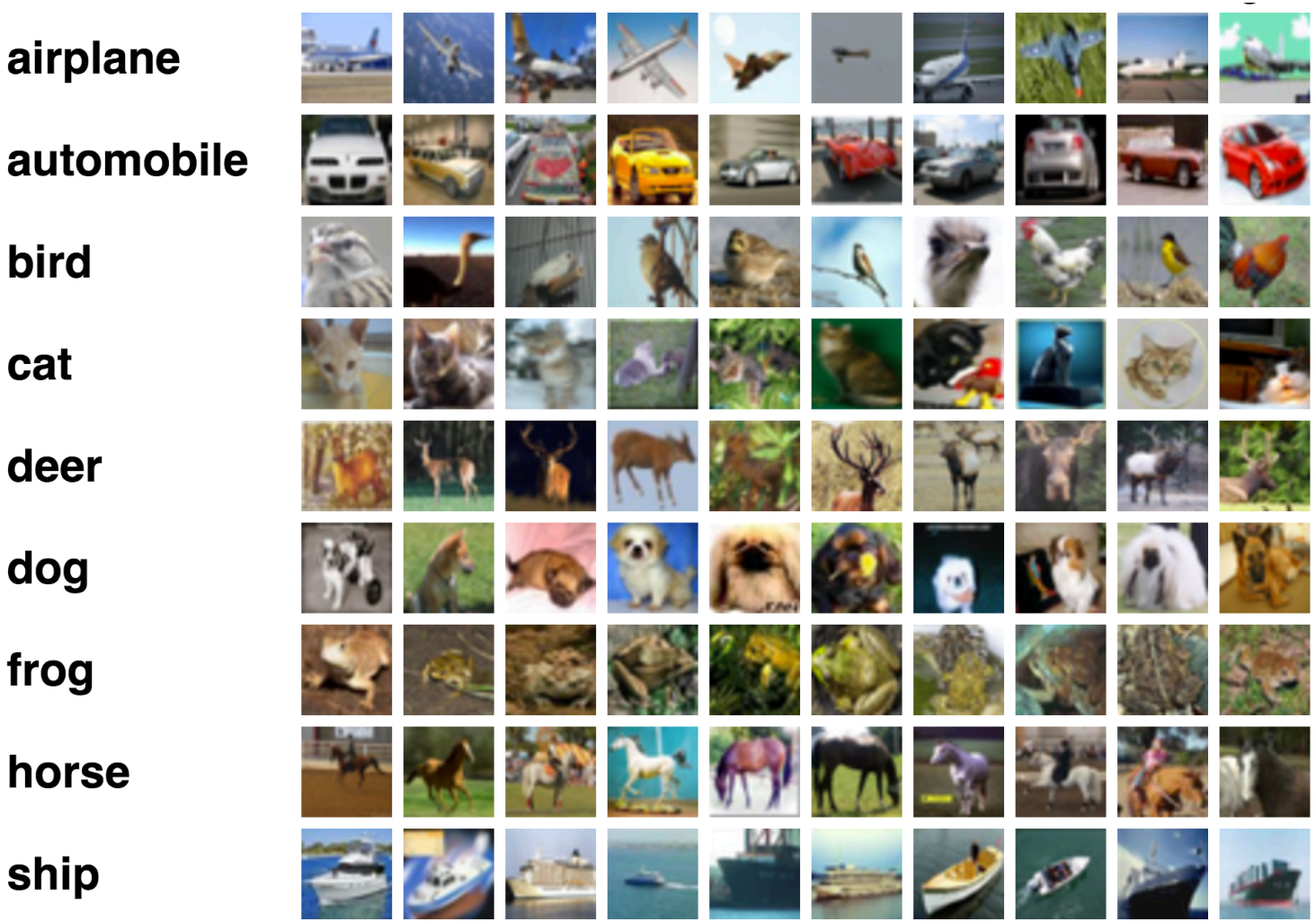
- Multiclass classification overview
- Reducing multiclass to binary
  - One-against-all & One-vs-one
- Training a single classifier
  - Multiclass Perceptron

What is multiclass

- Output $\in \{1,2,3, \ldots K\}$
  - In some cases, output space can be very large (i.e., K is very large)
- Each input belongs to exactly one class (c.f. in multilabel, input belongs to many classes)



Example applications



Two key ideas to solve multiclass

- Reducing multiclass to binary
  - Decompose the multiclass prediction into multiple binary decisions
  - Make final decision based on multiple binary classifiers
- Training a single classifier
  - Minimize the empirical risk
  - Consider all classes simultaneously

Reduction v.s. single classifier

- Reduction
  - Future-proof: binary classification improved so does muti-class
  - Easy to implement
- Single classifier
  - Global optimization: directly minimize the empirical loss; easier for joint prediction
  - Easy to add constraints and domain knowledge

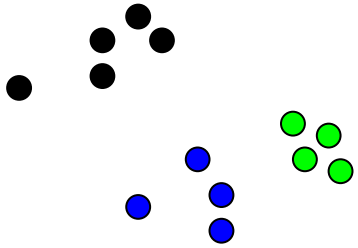
This Lecture

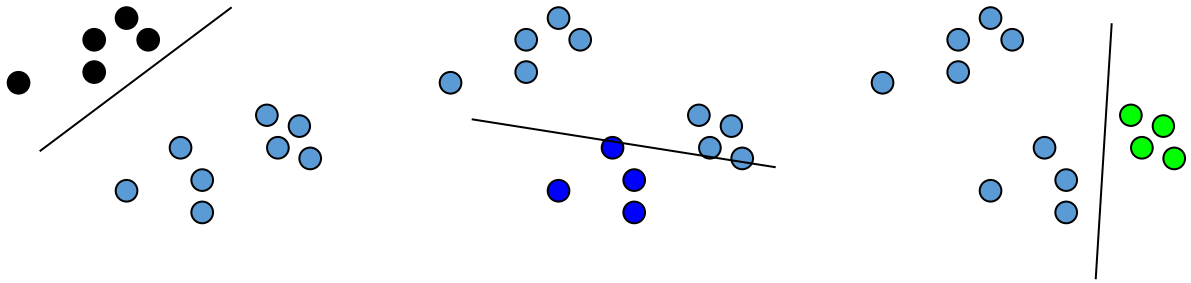
- Multiclass classification overview
- Reducing multiclass to binary
  - One-against-all & One-vs-one
- Training a single classifier
  - Multiclass Perceptron

One against all strategy

One against All learning

- Multiclass classifier
  - Function $f : R^n \rightarrow \{1,2,3,\ldots,k\}$
- Decompose into binary problems



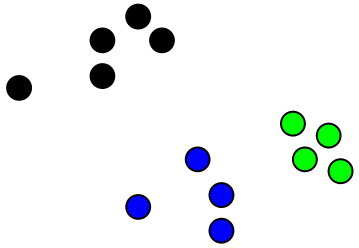


One-again-All learning algorithm

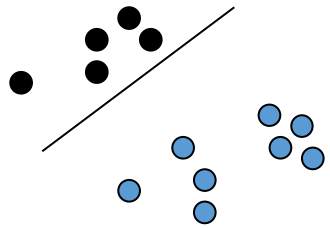
- Learning: Given a dataset $D = \{(x_i, y_i)\}$
  $x_i \in R^n, y_i \in \{1,2,3, \ldots K\}$
- Decompose into K binary classification tasks
  - Learn K models: $w_1, w_2, w_3, \ldots w_K$
  - For class k, construct a binary classification task as:
    - Positive examples: Elements of D with label k
    - Negative examples: All other elements of D
  - The binary classification can be solved by any algorithm we have seen

One against All learning

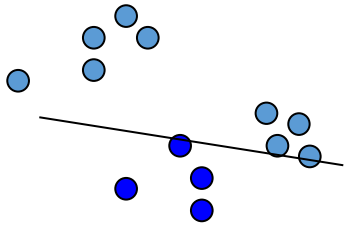
- Multiclass classifier
  - Function $f : R^n \rightarrow \{1,2,3,\ldots,k\}$
- Decompose into binary problems



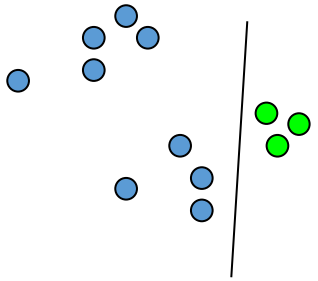
Ideal case: only the correct label will have a positive score



$w_{black}^T x > 0$



$w_{blue}^T x > 0$



$w_{green}^T x > 0$

One-again-All Inference

- Learning: Given a dataset $D = \{(x_i, y_i)\}$
  $x_i \in R^n, y_i \in \{1,2,3, \dots K\}$
- Decompose into K binary classification tasks
  - Learn K models: $w_1, w_2, w_3, \dots w_K$
- Inference: "Winner takes all"
  - $\hat{y} = \text{argmax}_{y \in \{1,2,\dots K\}} \, w_y^T \, x$

For example: $\text{y} = \text{argmax}(w_{black}^T \, x \, , \;\; w_{blue}^T \, x, \;\; w_{green}^T \;\; x)$

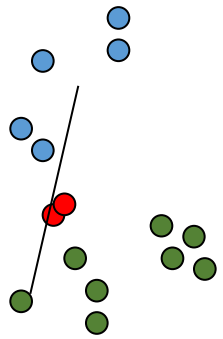
  - An instance of the general form

$$\hat{y} = \text{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \, f(\boldsymbol{y}; \boldsymbol{w}, \boldsymbol{x})$$

$$w = \{w_1, w_2, \dots w_K\}, \; f(\boldsymbol{y}; \boldsymbol{w}, \boldsymbol{x}) = \boldsymbol{w}_y^T \, x$$

One-again-All analysis

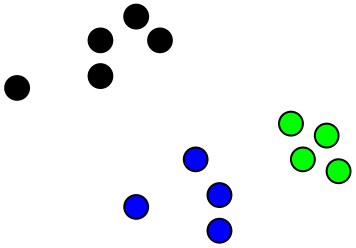
- Not always possible to learn
  - Assumption: each class individually separable from all the others
- No theoretical justification
  - Need to make sure the range of all classifiers is the same – we are comparing scores produced by K classifiers trained independently.
- Easy to implement; work well in practice

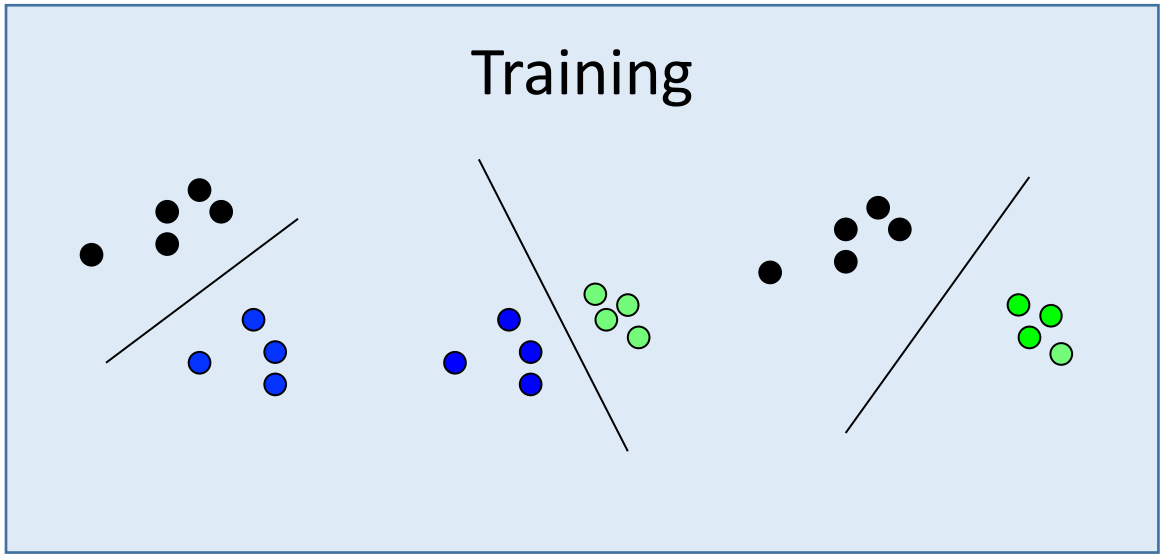


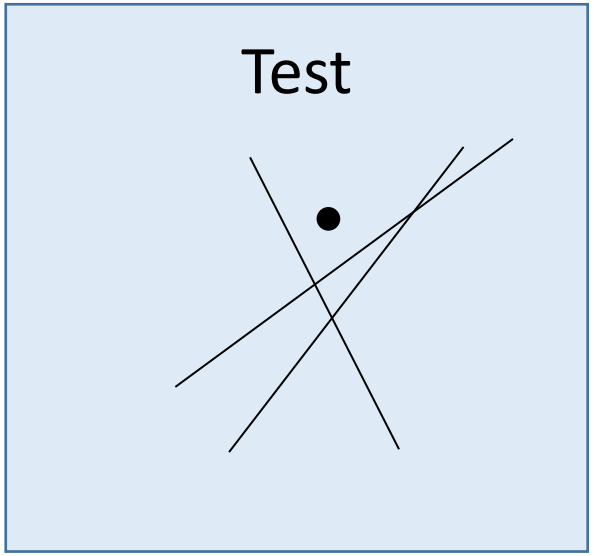
One v.s. One (All against All) strategy

One v.s. One learning

- Multiclass classifier
  - Function $f : R^n \rightarrow \{1,2,3,\ldots,k\}$
- Decompose into binary problems







One-v.s-One learning algorithm

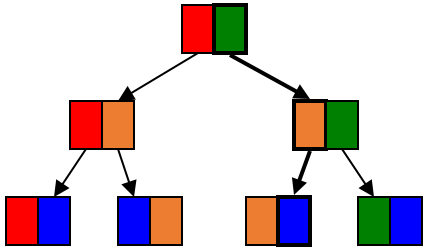
- Learning: Given a dataset $D = \{(x_i, y_i)\}$
  $x_i \in R^n, y_i \in \{1,2,3, \ldots K\}$
- Decompose into C(K,2) binary classification tasks
  - Learn C(K,2) models: $w_1, w_2, w_3, \ldots w_{K*(K-1)/2}$
  - For each class pair (i,j), construct a binary classification task as:
    - Positive examples: Elements of D with label i
    - Negative examples Elements of D with label j
    - The binary classification can be solved by any algorithm we have seen

One-v.s-One Inference algorithm

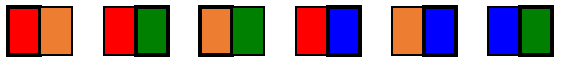
- Decision Options:
  - More complex; each label gets k-1 votes
  - Output of binary classifier may not cohere.
  - Majority: classify example x to take label i if i wins on x more often than j (j=1,…k)
  - A tournament: start with n/2 pairs; continue with winners

Classifying with One-vs-one

Tournament



Majority Vote



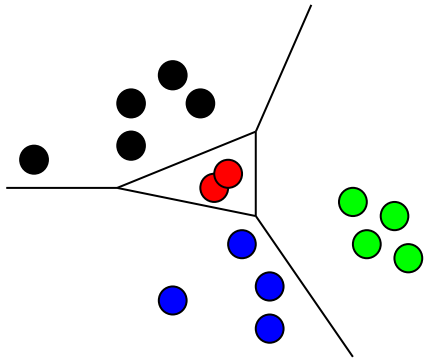
1 red, 2 yellow, 2 green

➔ ?

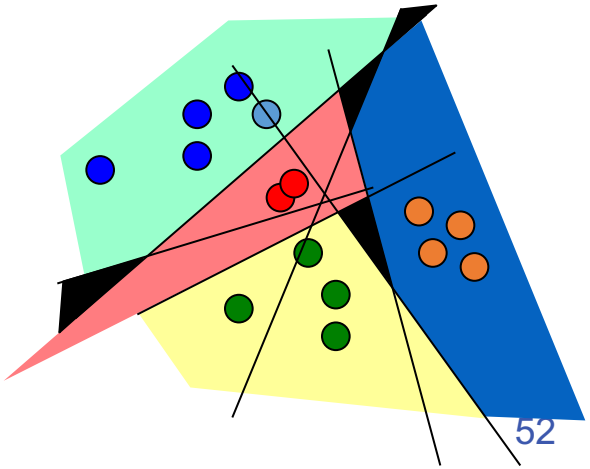
All are post-learning and *might* cause weird stuff

One-v.s.-one Assumption

- Every pair of classes is separable



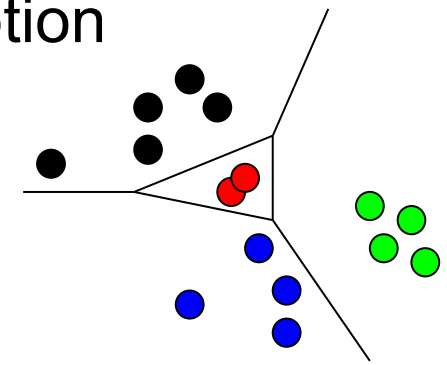
It is possible to separate all k classes with the $O(k^2)$ classifiers



Decision Regions

Comparisons

- One against all
  - O(K) weight vectors to train and store
  - Training set of the binary classifiers may unbalanced
  - Less expressive; make a strong assumption



- One v.s. One (All v.s. All)
  - O(K^2) weight vectors to train and store
  - Size of training set for a pair of labels could be small
    ⇒ overfitting of the binary classifiers
  - Need large space to store model

Problems with Decompositions

- Learning optimizes over *local* metrics
  - Does not guarantee good *global* performance
  - We don't care about the performance of the *local* classifiers
- Poor decomposition $\Rightarrow$ poor performance
  - Difficult local problems
  - Irrelevant local problems
- Efficiency: e.g., All vs. All vs. One vs. All
- Not clear how to generalize multi-class to problems with a very large # of output

Still an ongoing research direction

Key questions:

- How to deal with large number of classes
- How to select "right samples" to train binary classifiers

- Error-correcting tournaments
  [Beygelzimer, Langford, Ravikumar 09]
- Logarithmic Time One-Against-Some
  [Daume, Karampatziakis, Langford, Mineiro 16]
- Label embedding trees for large multi-class tasks.
  [Bengio, Weston, Grangier 10]
- …

Decomposition methods: Summary

- General Ideas:
  - Decompose the multiclass problem into many binary problems
  - Prediction depends on the decomposition
    - Constructs the multiclass label from the output of the binary classifiers
- Learning optimizes local correctness
  - Each binary classifier don't need to be globally correct and isn't aware of the prediction procedure

This Lecture

- Multiclass classification overview
- Reducing multiclass to binary
  - One-against-all & One-vs-one
- Training a single classifier
  - Multiclass Perceptron

Revisit One-again-All learning algorithm

- Learning: Given a dataset $D = \{(x_i, y_i)\}$
  $x_i \in R^n, y_i \in \{1,2,3, \ldots K\}$
- Decompose into K binary classification tasks
  - Learn K models: $w_1, w_2, w_3, \ldots w_K$
  - w_k: separate class k from others
- Prediction

$$\hat{y} = \text{argmax}_{y \in \{1,2,\ldots K\}} \; w_y^T \; x$$

Observation

- At training time, we require $w_i^T x$ to be positive for examples of class i.
- Really, all we need is for $w_i^T x$ to be more than all others ⇒ this is a weaker requirement

For examples with label i, we need

$$w_i^T x > w_j^T x \qquad \text{for all } j$$

Perceptron-style algorithm

For examples with label i, we need

$$w_i^T x > w_j^T x \quad \text{for all } j$$

- For each training example (x, y)
  - If for some y', $w_y^T x \leq w_{y'}^T x$ mistake!
    - $w_y \leftarrow w_y + \eta x$ update to promote y
    - $w_{y'} \leftarrow w_{y'} - \eta x$ update to demote y'

Why add ηx to w_y promote label y:

Before update $s(y) = < w_y^{old}, x >$

After update

$$\begin{aligned} s(y) &= < w_y^{new}, x > \\ &= < w_y^{old} + \eta x, x > \\ &= < w_y^{old}, x > + \eta < x, x > \end{aligned}$$

Note! $< x, x > = x^T x > 0$

A Perceptron-style Algorithm

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$

For epoch $1 \ldots T$:

For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:

For $y' \neq y$

if $w_y^T x < w_{y'}^T x$ make a mistake

$w_y \leftarrow w_y + \eta x$ promote y

$w_{y'} \leftarrow w_{y'} - \eta x$ demote y'

Return $\boldsymbol{w}$

How to analyze this algorithm and simplify the update rules?

Prediction: $\text{argmax}_y \; w_y^T x$

What you need to know

- Reducing multiclass to binary
  - One-against-all & One-vs-one
- Training a single classifier
  - Multiclass Perceptron