Advanced Machine Learning

Content-based Rec. Systems
& AdaBoost

Outline

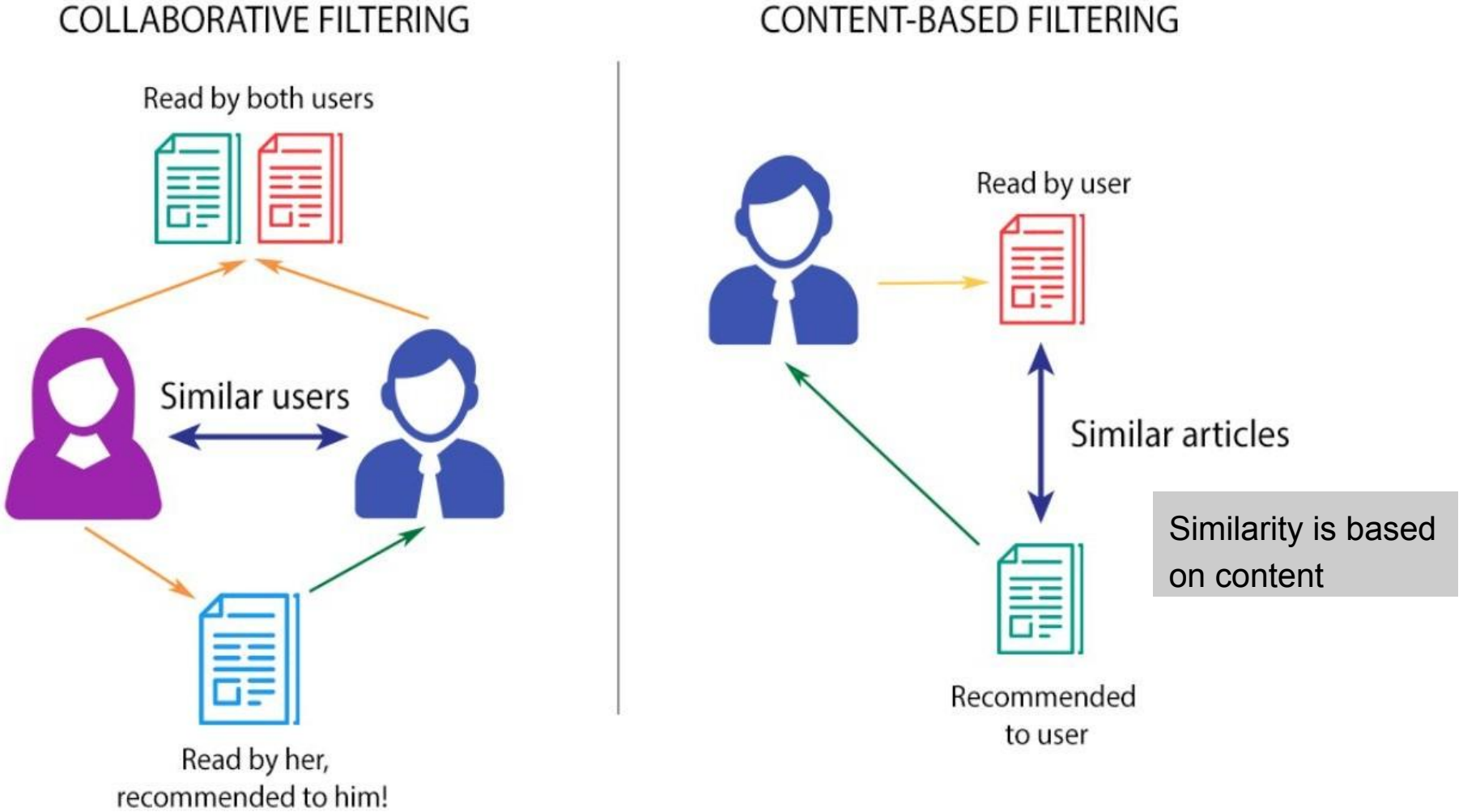
- Go over quiz
- Content-based Rec. Systems
- Review of decision trees
- Boosting question
- Ensemble models
- Weighted decision stumps
- AdaBoost

Content-based recommendations

Classification of Recom. Systems

- **Content-based systems:**
  - analysis of the attributes of the items (movies, books) to generate predictions
  - build user profiles based on the **content features** of the items rated by the user
  - match user profile attributes with against the attributes of the content object

- **Collaborative filtering:**
  - recommend items based on similarity measures between users and/or items
  - items recommended to a user are those preferred by similar users



https://www.themarketingtechnologist.co/building-a-recommendation-engine-for-geek-setting-up-the-prerequisites-13/

What type of system?

You run a website that recommends blog post to users. You don’t collect user ids or cookies.

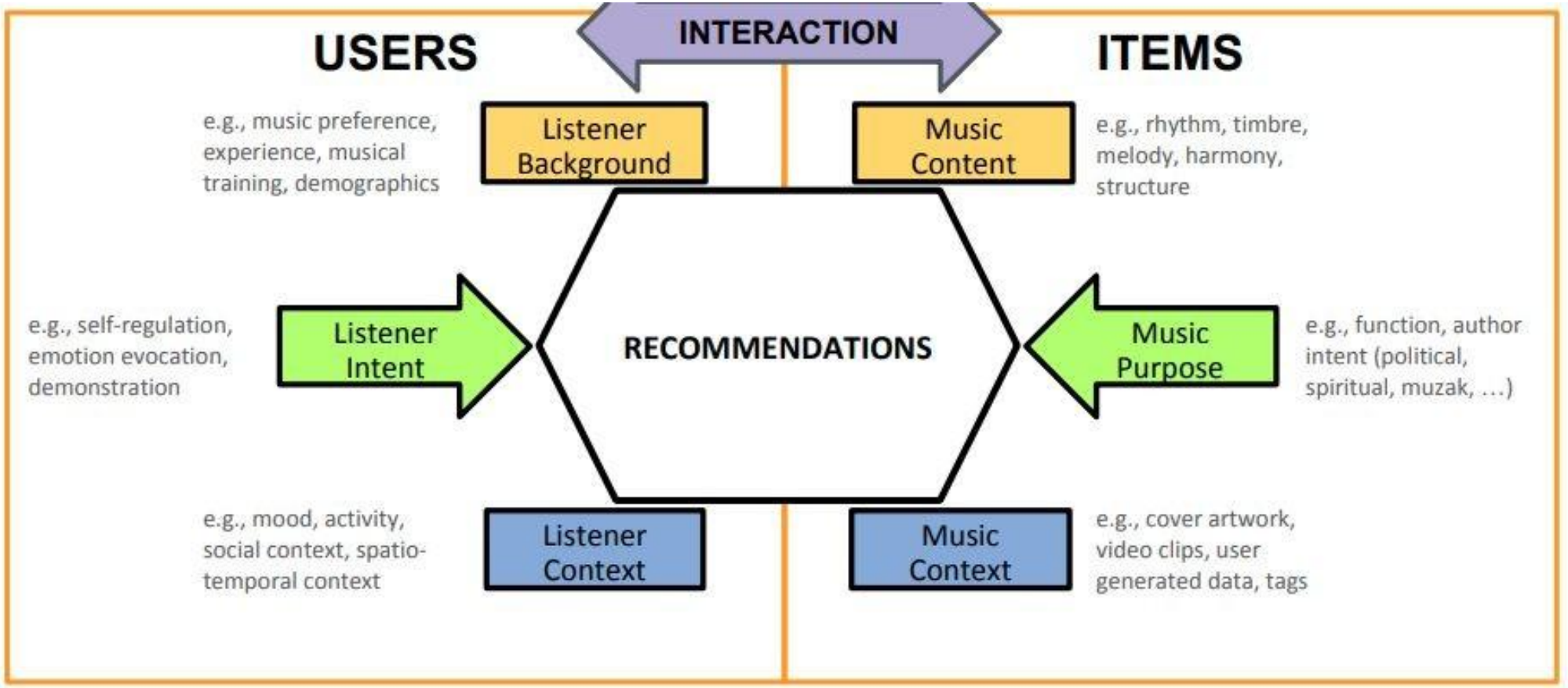
You want to recommend a next article to read. What type of system would you recommend with this type of data? Why?

Ingredients for Content-based Rec.

- Item representation
  - Based on content
- User representation
  - Based on items “liked” by user

- Similarity metric / Machine learning model
- Sometimes ratings

Example: music recommendation



Item profile for music recommendation

- Audio Content Analysis
  - Features can be extracted from any audio file
    - Beat, timbre, tonal features
  - Learn high level descriptors from low-level features via machine learning (e.g. deep learning)
    - Learning genre (house, bass, funk, DnB, jazz, classical, metal)
    - Learning semantic categories: not_danceable, gender_male, mood_not_happy
  - Many libraries will do some of this already

Item profile for music recommendation

- Audio Content Analysis
- Text Content Analysis
  - Text processing of user generated context and lyrics

Item profile for movie recommendation

- The set of actors of the movie.
- The director.
- The year in which the movie was made.
  - Some viewers prefer old movies,others watch only the latest releases.
- The genre or general type of movie.
  - Some viewers like only comedies, others dramas or romances.

Item profile for movie recommendation

- The set of actors of the movie.
- The director.
- The year in which the movie was made.
  - Some viewers prefer old movies, others watch only the latest releases.
- The genre or general type of movie.
  - Some viewers like only comedies, others dramas or romances.

Can you think about other features? Can we make features from reviews? How do we get a user profile?

Item profile for movie recommendation

- Source
  - Author, publisher
- Location
  - Movies (only interesting for a region), pictures can be tagged to location
  - Represented with latitude, longitude or country/state/city
- Image features
- Audio features
- Application specific

User profile for movie recommendations

- **Content-based profile:** summary of the profile of the items these user liked / purchased

Example **Netflix recommendations**

Item profile:

x_1 = has Julia Roberts,

x_2 = directed by Lars von Trier,

x_3 = is horror,

x_4 = is a comedy

User profile:

x_1 = probability that the user likes movies with Julia Roberts,

x_2 = probability that the user likes movies directed by Lars von Trier,

x_3 = probability that the user that likes horror movies,

x_4 = probability that the user likes comedies

User profile for movie recommendations

- Demographics
- Declared Interests
- User current location (from IP address)
- Usage-based features
  - Last time visit, frequency of visits (weekly, monthly), frequency per device
- Search history
- Item set
  - Set of items for which the user showed interest (e.g. clicked, shared, liked)

Recommending Items to Users based on content: *kNN*

- Compute profile vectors for users and items
- Find a similarity measure and compute similarity between users and items
- Recommend items with *high similarity* to the user prof.
  - K-nearest neighbors

Recommending Items to Users based on content: *kNN*

- Scaling up
  - When you have too many users and items it is not feasible to compute similarity between all of them
  - Locality-sensitive-hashing techniques can be used to place item profiles in buckets
  - Given a user, easy to find buckets with high similarity to the user

Recommending Items to Users:
training a model

- Compute profile vectors for users and items
- Train a model using the feature vector to predict observed ratings
  - Actual ratings, clicks, likes or a combination
  - Regression to predict numerical ratings
  - Classification to predict prob of click
  - This approach could include features that are not content-based

Advantages of Content-based Recommendation

- You don't need data on other users
- You don't have **cold-start** problem for a new item
  - Able to recommend new and unpopular items
- You can provide explanations of recommended items by listing content features that caused an item to be recommended

Challenges with Content-based Recommendation

- Constructing the feature vector could be a difficult task (need domain knowledge)
- New genres are hard to recognize
- Some kind of items are not amenable to easy feature extraction (movies, music)
- Hard to exploit quality of judgements of other users

Helpful References

- https://www.sciencedirect.com/science/article/pii/S1110866515000341
- https://research.googleblog.com/2018/01/the-google-brain-team-looking-back-on_12.html
- https://nlp.stanford.edu/manning/talks/Simons-Institute-Manning-2017.pdf
- http://www.cp.jku.at/tutorials/mrs_recsys_2017/slides.pd f

Boosting

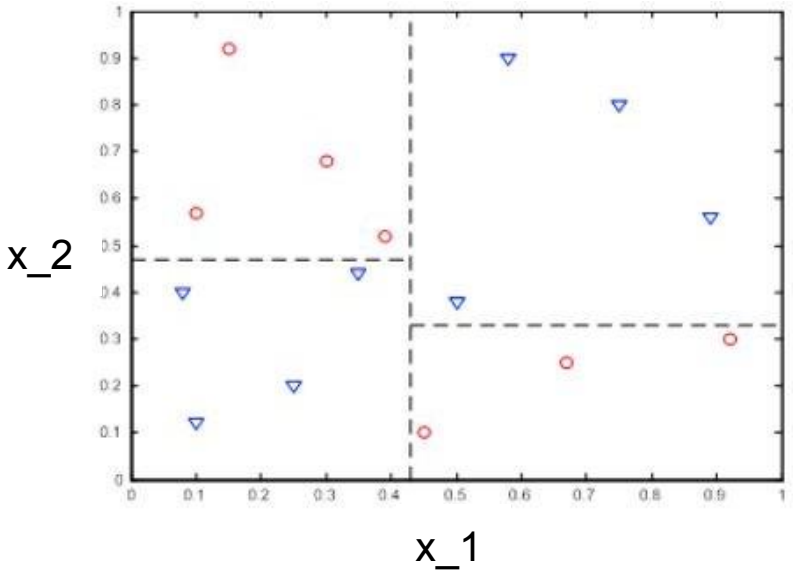
Required Reading: *Ch. 18 of*
Probabilistic Machine Learning
by Kevin Patrick Murphy

Emphasis on Sect. 18.5 - Boosting

Review of Decision Trees

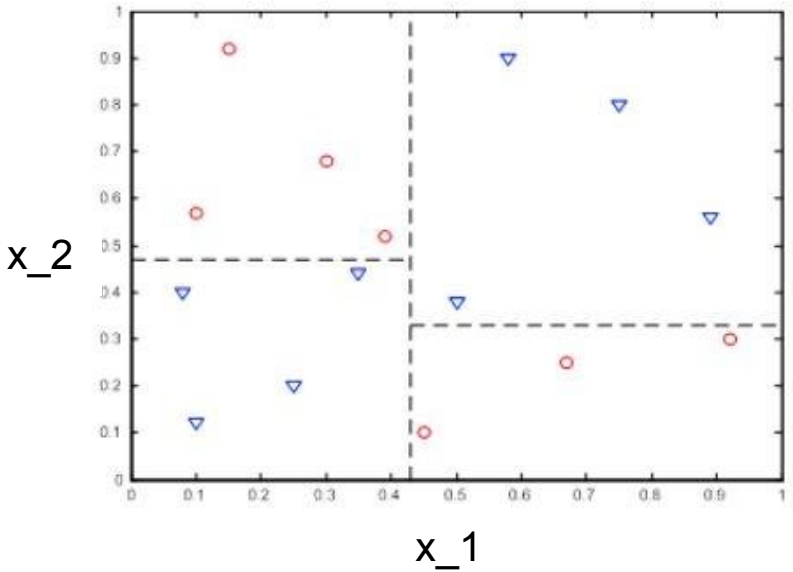
Review of Decision Trees

- Popular machine learning algorithm
- Building block for random forests and gradient boosting
- Interpretable
- “Bushy trees” tend to
  - Overfit
  - Have high variance (Sensitive to small changes in the training data)
  - Low bias (Fit well the training data)



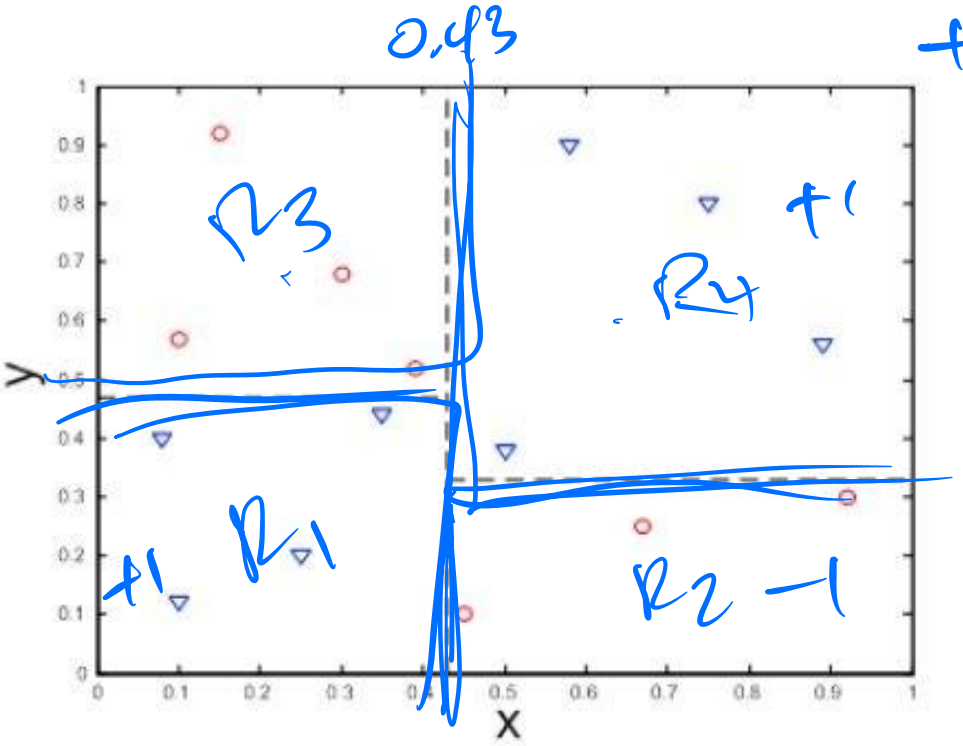
Review of Decision Trees

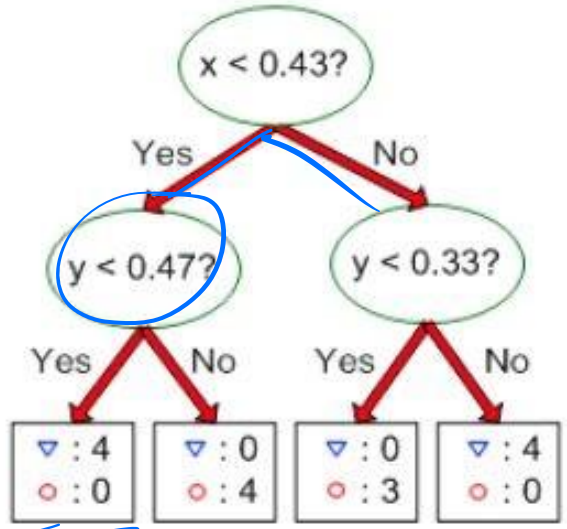
- Bushy trees tend to
  - Overfit
  - Have high variance (Sensitive to small changes in the training data)
  - Have low bias
- Shallow trees tend to
  - Underfit
  - Have high bias



Decision tree problem

$$\pi(x) = (1)\mathbb{I}(x_i \in R_1) + (-1)\mathbb{I}(x_i \in R_2) + (-1)\mathbb{I}(x_i \in R_3) + (1)\mathbb{I}(x_i \in R_4)$$



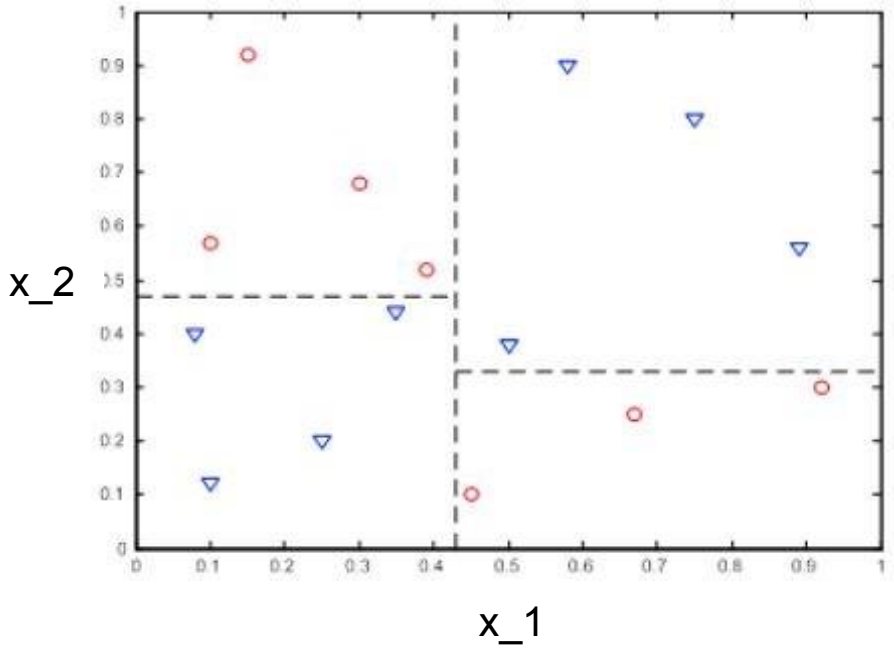


Problem: Find the best split that separates the circles from triangles. Splits must be parallel to axis.

Decision tree problem

Problem: Find the best split that separates the circles from triangles. Splits must be parallel to axis.

This is a really hard problem!!



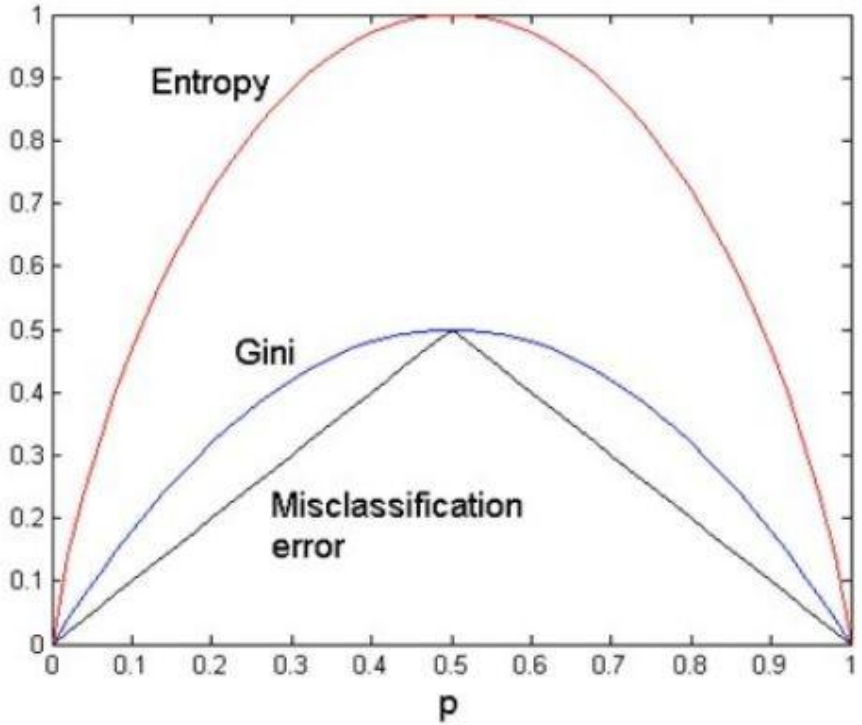
Impurity

How to formalize the decision tree problem?

Find the splits that minimize impurity.

Measures of impurity

For a two class problem:



http://slideplayer.com/slide/7696713/#

Decision tree pseudocode

1. Place the best attribute (`x_j` and value `a`) of the dataset at the root of the tree

2. Split the training set into subsets:

   `{x_j <= a}` and `{x_j > a}`

3. Repeat step 1 and step 2 on each subset until you find leaf nodes or a stopping condition is met

Binary decision stumps

Let's $x = (x_1, ..., x_D)$ where each feature x_j is a continuous variable.

$$T(x; \theta) = \begin{cases} 1, & \text{if } x_j > \alpha \\ -1, & \text{otherwise} \end{cases}$$

Or

$$T(x; \theta) = \begin{cases} 1, & \text{if } x_j \leq \alpha \\ -1, & \text{otherwise} \end{cases}$$

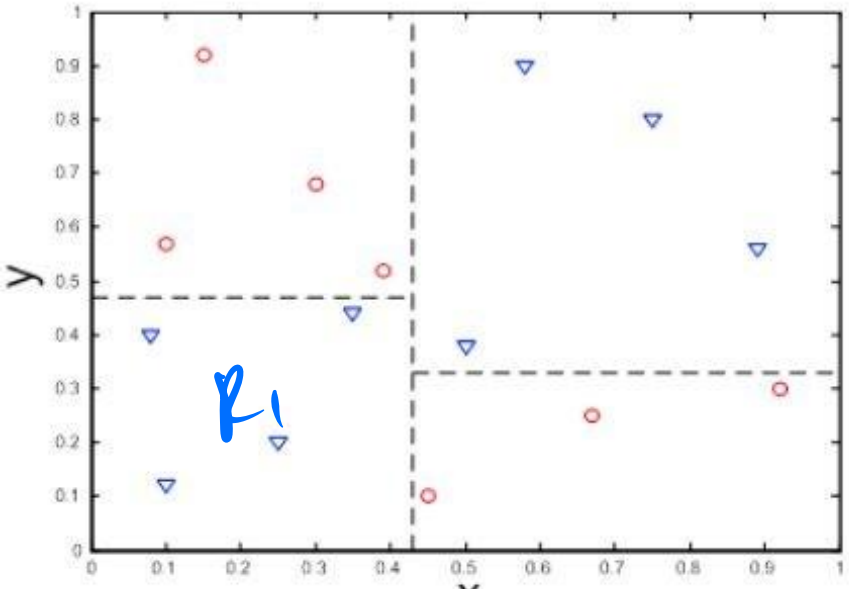
Tree notation

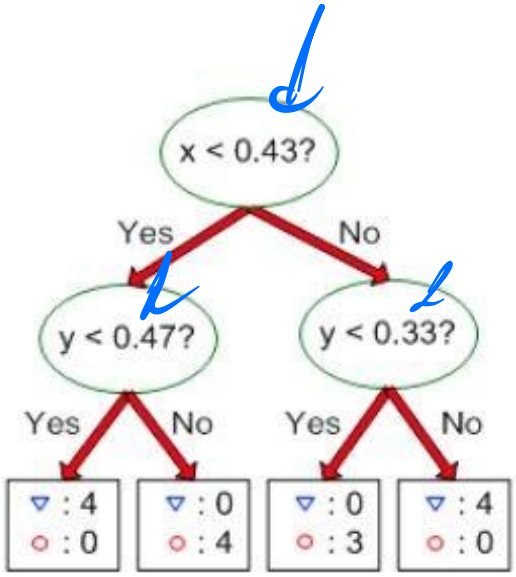
Trees partition the space into *disjoint* regions $R_j, j \dots J$. For x in particular region R_j the value of $T(x)$ constant. A tree can be formally expressed as:

$$T(x;\theta) = \sum_{j=1}^{J} \beta_j 1_{[x \in R_j]}$$

where θ is the set of parameters for the trees $\theta = \{\{R_j\}_1^J; \{\beta_j\}_1^J\}$.

Tree notation: Exercise





$$T(x;\theta) = \sum_{j=1}^{J} \beta_j 1_{[x \in R_j]}$$

$R_1 = \{x < 0.43\} \cap \{y < 0.47\}$

What are the parameters for this tree?

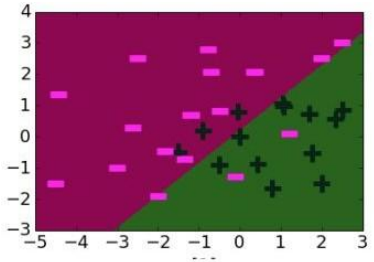
The Boosting Question

Binary classification with hard predictions

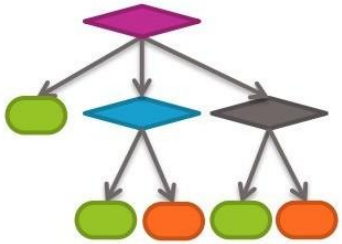
Given training observations, $(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})$

Let $\hat{y}^{(i)}$ be either 1 or -1.

Weak classifiers



Logistic regression w. simple features

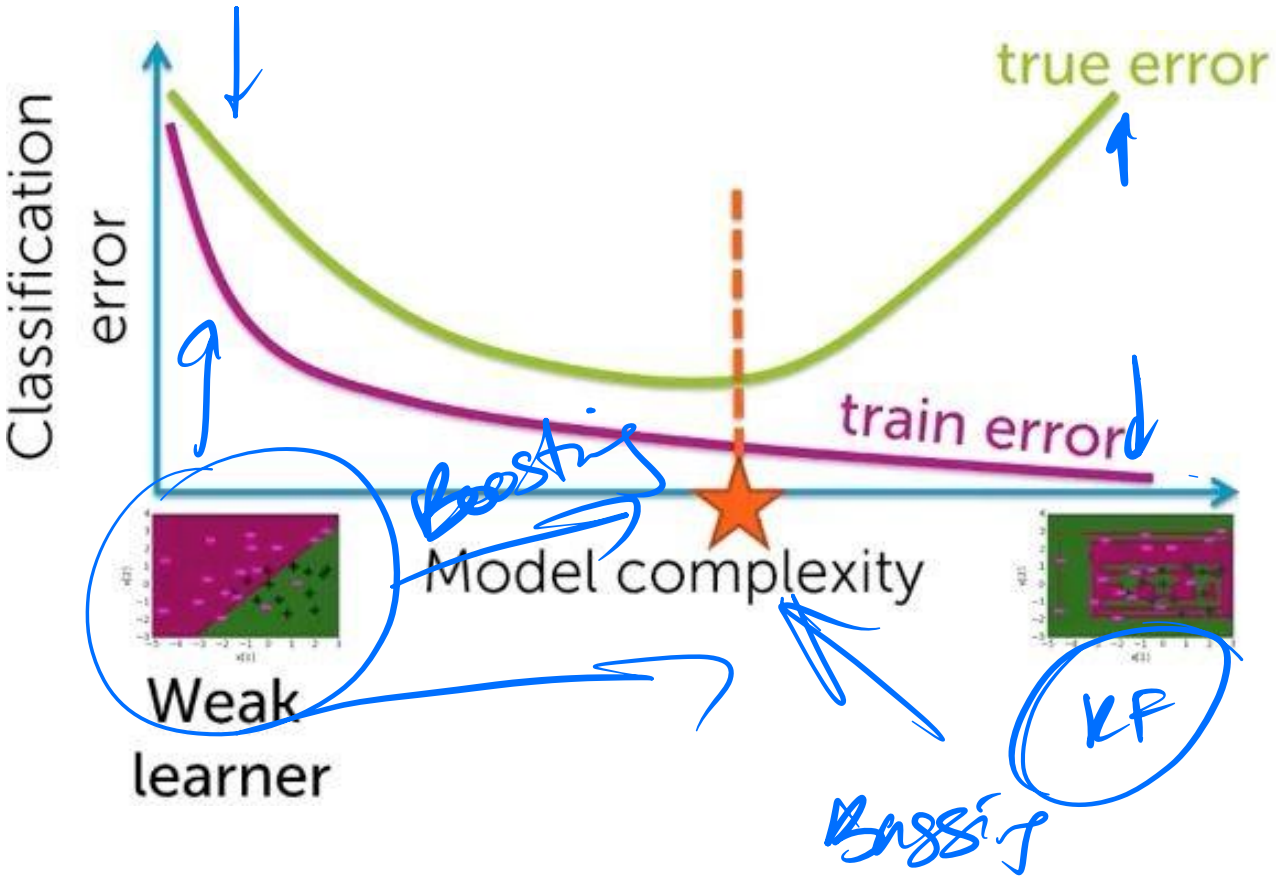


Shallow decision trees

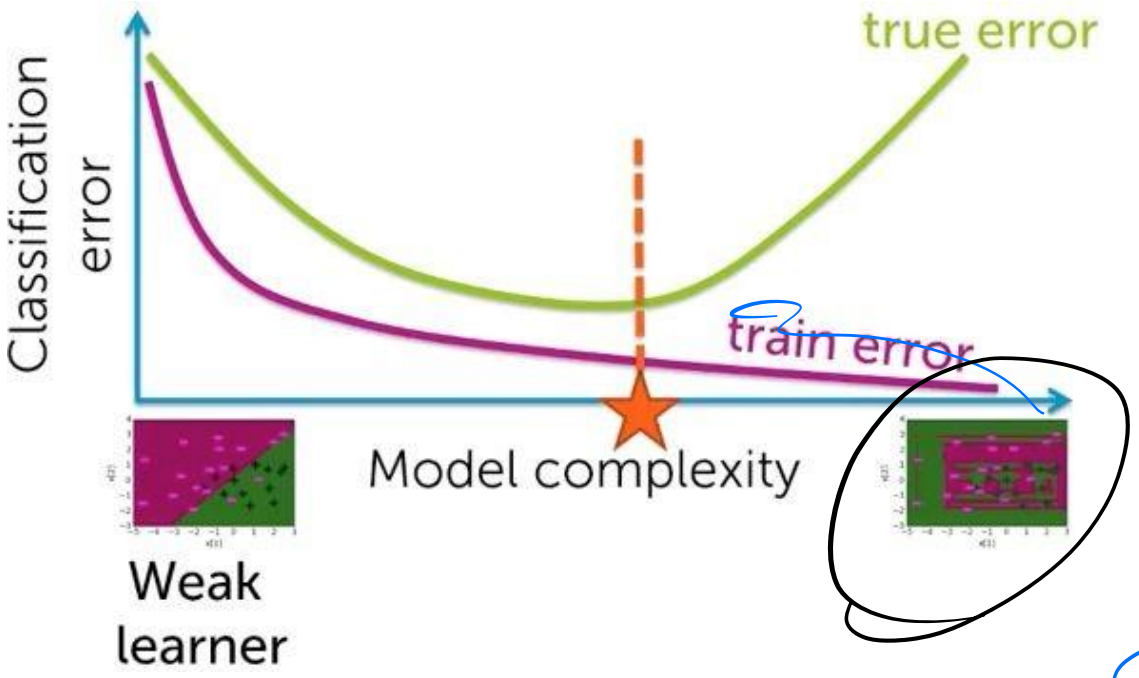
Decision stumps

- Fast to fit
- Does not overfit (low variance)
- Tends to underfit: does not fit our data well (high bias)

Finding the right classifier: Learning curve



How to get a strong classifier?



- Add polynomial features or more features
- Ensemble
  - Random Forest
- Is there another way??

The Boosting Question

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

Weak learner: a learner with > 50% accuracy

1 − class err.

Ensemble: each classifier “votes” on prediction

$G_1(x) = +1$ $G_2(x) = -1$ $G_3(x) = -1$ $G_4(x) = +1$

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

$$G(x) = sign(\alpha_1 G_1(x) + \alpha_2 G_2(x) + \alpha_3 G_3(x) + \alpha_4 G_4(x))$$

Ensemble classifier

- Base learner that outputs 1 or -1
- Ensemble model
  - M classifiers
  - M coefficients
- Prediction

$$\hat{y} = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$$

Prediction with ensemble:

What is the prediction in this example?

$G_1(x) = +1$ $G_2(x) = -1$ $G_3(x) = -1$ $G_4(x) = +1$

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

$$G(x) = sign(\alpha_1 G_1(x) + \alpha_2 G_2(x) + \alpha_3 G_3(x) + \alpha_4 G_4(x))$$

where $\alpha_1 = 2, \alpha_2 = 1.5, \alpha_3 = 1.5, \alpha_4 = 0.5$

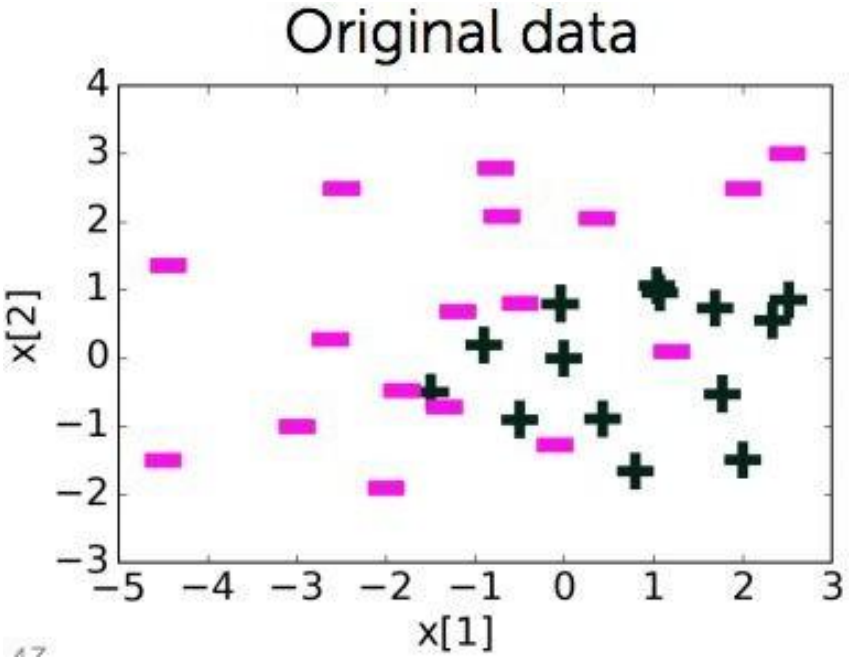
$G(x) = sign(2 - 1.5 - 1.5 + 0.5) = sign(-0.5) = -1$

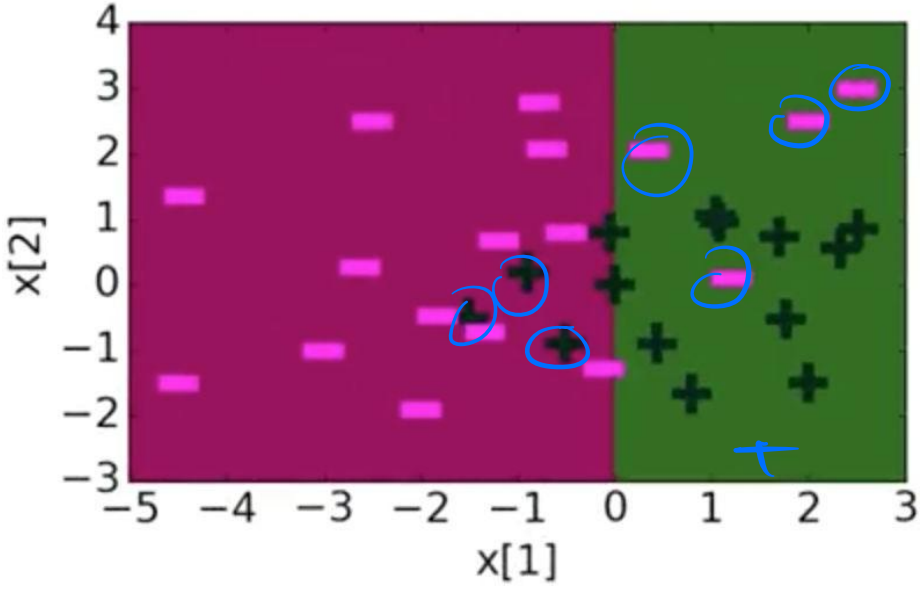
Preview of AdaBoost

Algorithm 1 Adaboost

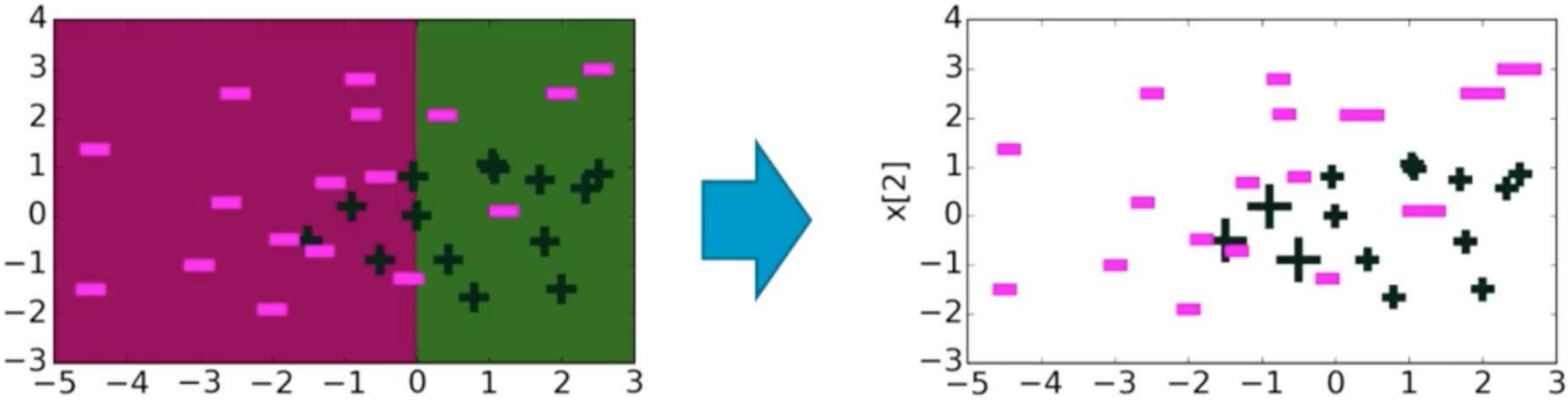
procedure ADABOOST
    Maintain a weight of each point in a training set $\{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$
    Initialize weights $w_i = \frac{1}{N}, i \in [1 \cdots N]$
    for iter $= 1$ to M **do**
        Apply base learner to weighted examples.
        Increase weights of misclassified examples.
    end for
    Combine models by weighted voting.
end procedure

AdaBoost: first decision stump

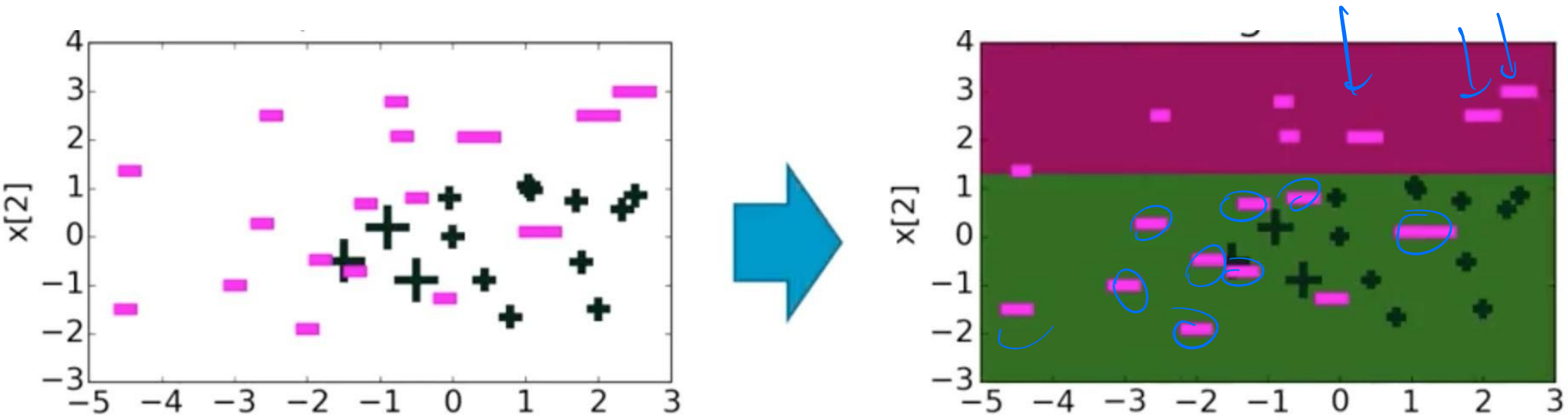




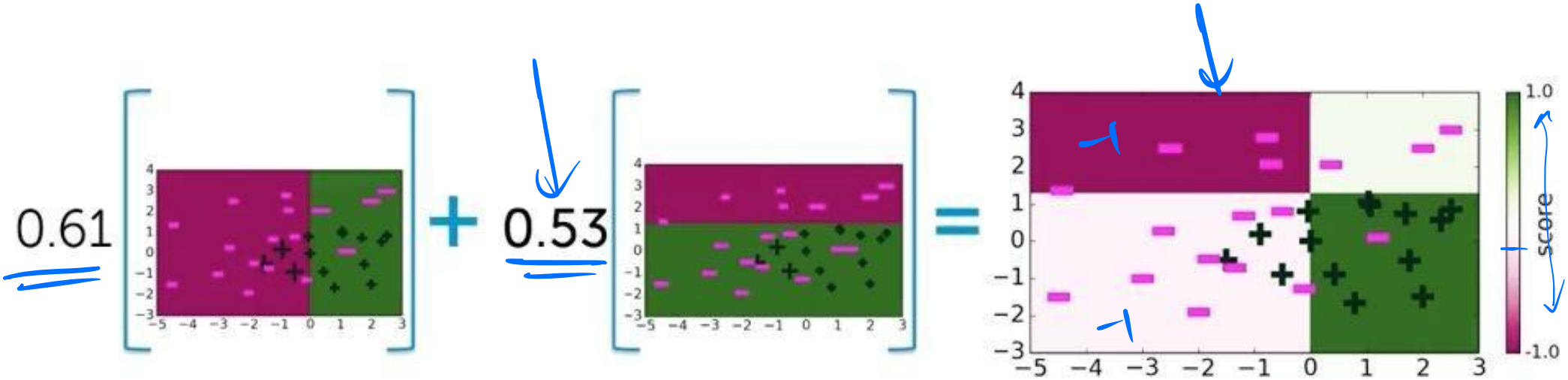
AdaBoost: learned weights



AdaBoost: Second decision stump



AdaBoost Ensemble



Learning a decision stump

$$\text{Classification error} = \frac{\text{\# incorrect predictions}}{\text{\# examples}}$$

Find the feature and the threshold that will minimize the classification error.

How would you generalized classification error to weighted data?

Learning weighted decision stumps

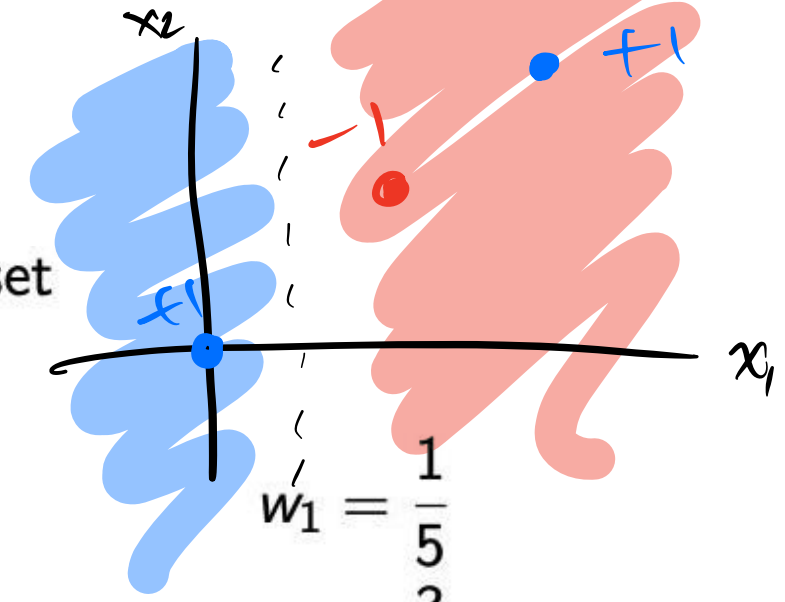
Weighted class err.

$$\theta^* = arg\min_{\theta} \frac{\sum_{i=1}^{N} w_i \mathbb{1}_{[y^{(i)} \neq T(x^{(i)};\theta)]}}{\sum_{i=1}^{N} w_i}$$

A fancy notation for weighted classification error

Exercise

$\frac{2/5}{6/5} = \frac{1}{3}$



Consider the following weighted dataset

$$x^{(1)} = (0,0), y^{(1)} = 1, \qquad w_1 = \frac{1}{5}$$

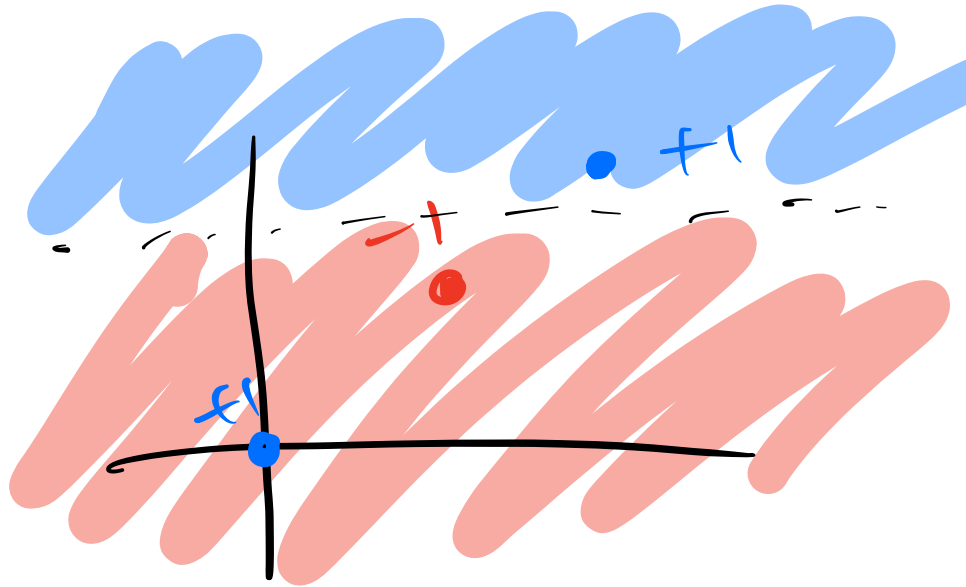
$$x^{(2)} = (1,1), y^{(2)} = -1, \qquad w_2 = \frac{3}{5}$$

$$x^{(3)} = (2,2), y^{(3)} = 1, \qquad w_3 = \frac{2}{5}$$

Find a stump that minimizes the weighted classification error.

Exercise

$\frac{1/5}{6/5} = 1/6$



1. Consider the following dataset.

$x = (x_1, x_2)$	y	weight
$(0, 0)$	-1	$\frac{2}{10}$
$(0, 1)$	-1	$\frac{2}{10}$
$(1, 1)$	-1	$\frac{3}{10}$
$(1, 3)$	1	$\frac{1}{10}$
$(2, 1)$	1	$\frac{2}{10}$

(a) Compute the weighted classification error of the following classifier. Which points are misclassified?

$$g(x) = \begin{cases} 1, & \text{if } x_2 \geq 0.5 \\ -1, & \text{otherwise} \end{cases}$$

(b) Find a stump that minimizes the weighted classification error. What is the best weighted classification error?

Algorithm 2 Adaboost

procedure ADABOOT

Initialize weights $w_i = \frac{1}{N}, i \in [1 \cdots N]$

for $m = 1$ to M **do**

Fit a tree classifier $T_m(x) = T(x; \theta_m)$ to training data using weights w_i

Compute

$$err_m = \frac{\sum_{i=1}^{N} w_i \mathbb{1}_{[y^{(i)} \neq T_m(x^{(i)})]}}{\sum_{i=1}^{N} w_i}$$

Compute $\alpha_m = \log(\frac{1-err_m}{err_m})$

Set $w_i = w_i \cdot \exp[\alpha_m \cdot \mathbb{1}_{[y^{(i)} \neq T_m(x^{(i)})]}], i = 1, \ldots, N$

end for

$f(x) = \sum_{m=1}^{M} \alpha_m T_m(x)$

Return $F(x) = sign\,(f(x))$

end procedure

Computing classifier weights

$$\alpha_m = \log(\frac{1 - err_m}{err_m})$$

error	alpha
0.01	$\log(\frac{.99}{.01}) = \log(99) =$
0.2	$\log(\frac{0.8}{0.2}) = \log 4 \approx$
0.5	$\log(\frac{0.5}{0.5}) = \log(1) = 0$

We want the classifier to have large weight if error is small and a small weight is error is not so good.

Reweighting data to focus on mistakes

Increase the weights on incorrectly classified observations

$$w_i = w_i \cdot \exp[\alpha_m \cdot 1_{[y^{(i)} \neq T_m(x^{(i)})]}]$$

$$w_i = \begin{cases} w_i & \text{if } y^{(i)} = T_m(x^{(i)}) \\ w_i \cdot \exp[\alpha_m] & \text{otherwise} \end{cases}$$

	T_m(x) =y	err of T_m	New weight
obs 1	Correct	~~0.1~~ 0.2	w_1
2	Correct	~~0.5~~ 0.2	w_2
3	Incorrect	0.2	$4w_3$

Exercise:

Run AdaBoost by hand on this dataset for 2 iterations

temperature (x_1)	wind (x_2)	play tennis (y)
hot	F	no
hot	T	no
hot	F	yes
mild	F	yes
cool	F	yes
cool	T	no

References

- Probabilistic Machine Learning, Ch. 18
- https://www.coursera.org/learn/ml-classification/lecture/3ywWA/the-boosting-question
- The Elements of Statistical Learning
- https://www.coursera.org/learn/competitive-data-science/lecture/br2ze/practical-guide
- https://www.utdallas.edu/~nrr150130/cs7301/2016fa/lects/Lecture_10_Ensemble.pdf.