NLP 220 Data Science and Machine Learning Fundamentals

Jalal Mahmud

Lecture 5

Fall 2022

Including slides adapted from Marilyn Walker, Dan Jurafsky, Mu Li and Alex Smola

Outline for Today

- Text Classification Review
- Review of Naïve Bayes & Decision Tree
- SVM
- K-nearest neighbors
- Linear Regression

- Readings:
  - Ch 16 of Manning and Schuetze book

Text Classification

- Sentiment Classification
  - Review sentiment classification: IMDB movie reviews. Download data
- Topic Classification
  - Categorization of news articles. Download data

Named Entity Recognition (NER)

- Named entities are phrases that contain the names of persons, organizations, locations, times and quantities.
  Example:

  [ORG U.N.] official [PER Ekeus] heads for [LOC Baghdad]

  .
- CoNLL 2003, language independent NER.
- Data from English and German newspaper articles.
- Download data

Semantic Parsing for Task Oriented Dialogue

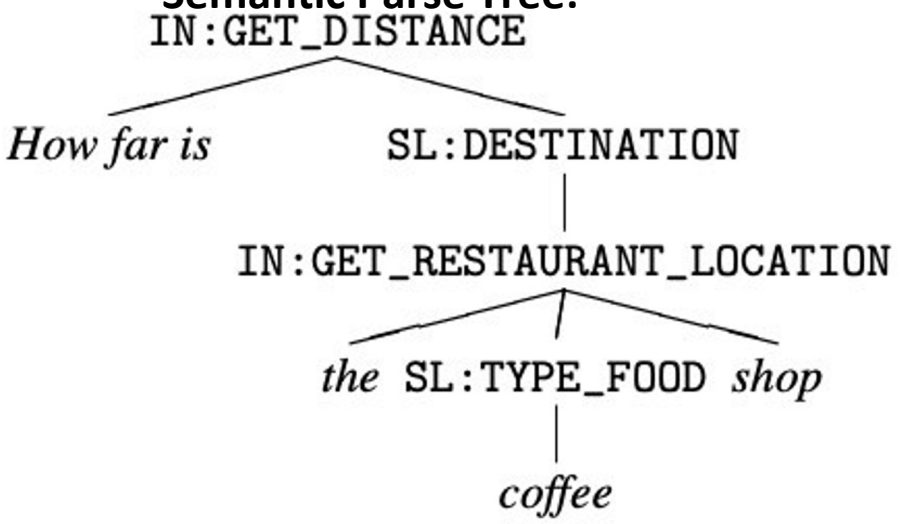
- Parsing natural language utterances into hierarchical structures: intent and slot labels
- Download data, Paper

Flat representation of the semantic parse tree:

[IN:GD How far is [SL:DEST [IN:GRL the [SL:TF coffee] shop]]]

Utterance: How far is the coffee shop?

Semantic Parse Tree:



Possible Approaches:

- Shift-reduce parser as described in the original paper.
- Sequence tagging: single pass or multi-pass.

Machine Reading Comprehension (MRC)

- **S**tanford **Qu**estion **A**nswering **D**ataset (SQuAD)

The first recorded travels by Europeans to China and back date from this time. The most famous traveler of the period was the Venetian Marco Polo, whose account of his trip to "Cambaluc," the capital of the Great Khan, and of life there astounded the people of Europe. The account of his travels, Il milione (or, The Million, known in English as the Travels of Marco Polo), appeared about the year 1299. Some argue over the accuracy of Marco Polo's accounts due to the lack of mentioning the Great Wall of China, tea houses, which would have been a prominent sight since Europeans had yet to adopt a tea culture, as well the practice of foot binding by the women in capital of the Great Khan. Some suggest that Marco Polo acquired much of his knowledge through contact with Persian traders since many of the places he named were in Persian.

How did some suspect that Polo learned about China instead of by actually visiting it?

Answer: through contact with Persian traders

Possible Approaches:

- Simple classifiers for answer sentence selection and then parsing to extract the answer.
- Neural networks with attention.
- Combination of the above.

Conversational QA

Similar to SQUAD, but a sequence of questions.

[CoQA]

[QuAC]

Possible Approaches:

- Similar ideas to MRC
- Coreference resolution at each step, and then similar ideas to MRC.

The Virginia governor's race, billed as the marquee battle of an otherwise anticlimactic 2013 election cycle, is shaping up to be a foregone conclusion. Democrat Terry McAuliffe, the longtime political fixer and moneyman, hasn't trailed in a poll since May. Barring a political miracle, Republican Ken Cuccinelli will be delivering a concession speech on Tuesday evening in Richmond. In recent ...

Q_1: What are the candidates **running** for?
A_1: Governor
R_1: The Virginia governor's race

Q_2: **Where**?
A_2: Virginia
R_2: The Virginia governor's race

Q_3: Who is the democratic candidate?
A_3: **Terry McAuliffe**
R_3: Democrat Terry McAuliffe

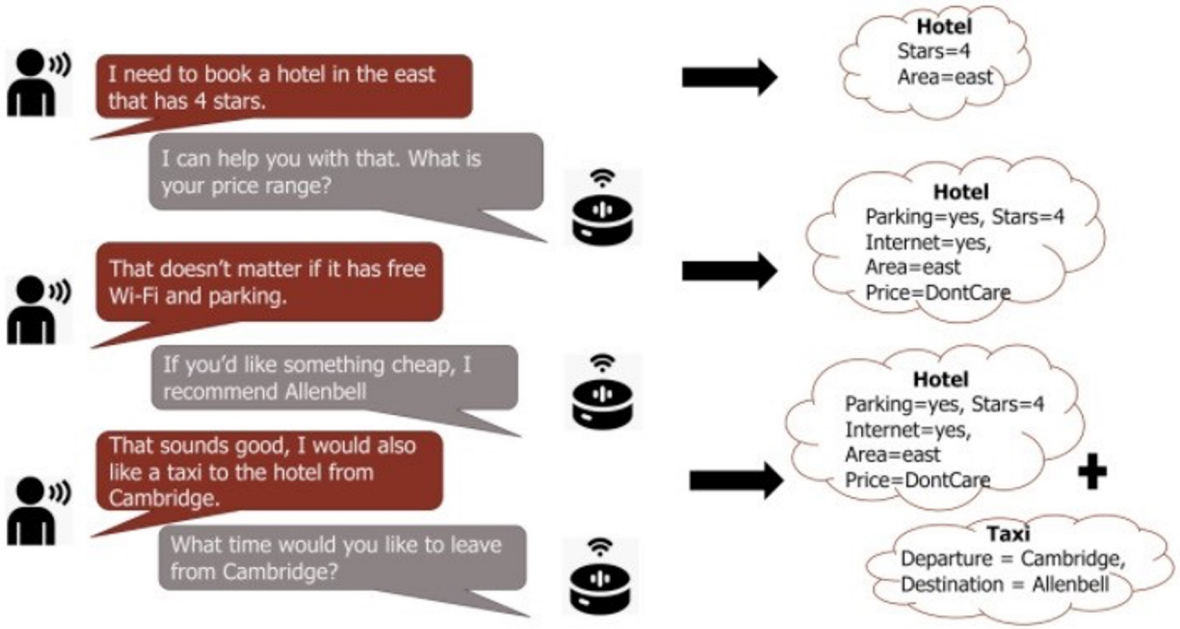
Q_4: Who is **his** opponent?
A_4: **Ken Cuccinelli**
R_4 Republican Ken Cuccinelli

Q_5: What party does **he** belong to?
A_5: Republican
R_5: Republican Ken Cuccinelli

Q_6: Which of **them** is winning?
A_6: Terry McAuliffe
R_6: Democrat Terry McAuliffe, the longtime political fixer and moneyman, hasn't trailed in a poll since May

MultiWOZ2.1

- Task-oriented dialogue system tasks, such as:
  - Dialogue state tracking
  - Next system action prediction



Dialogue states at several successive dialogue turns

Other NLP tasks

- One of more of GLUE tasks. Paper
- Single sentence tasks (i.e., sentiment classification)
- Similarity and Paraphrase tasks
- Inference tasks (i.e., relationship between two sentences, entailment, contradiction, neutral)

GLUE Tasks

Name	Download	More Info	Metric
The Corpus of Linguistic Acceptability			Matthew's Corr
The Stanford Sentiment Treebank			Accuracy
Microsoft Research Paraphrase Corpus			F1 / Accuracy
Semantic Textual Similarity Benchmark			Pearson-Spearman Corr
Quora Question Pairs			F1 / Accuracy
MultiNLI Matched			Accuracy
MultiNLI Mismatched			Accuracy
Question NLI			Accuracy
Recognizing Textual Entailment			Accuracy
Winograd NLI			Accuracy
Diagnostics Main			Matthew's Corr

Possible approaches for NLI Tasks

- Task: Given 2 sentences, do they entail or contradict each other or neither?
- You can learn a representation for each sentence (i.e., CNN or RNN), then use a similarity approach to determine entailment or not.
- You can also learn a classifier that uses the two representations and decides on one of the classes.

Other NLP Tasks

- One or more of SuperGLUE tasks

Name	Identifier	Download	More Info	Metric
Broadcoverage Diagnostics	AX-b			Matthew's Corr
CommitmentBank	CB			Avg. F1 / Accuracy
Choice of Plausible Alternatives	COPA			Accuracy
Multi-Sentence Reading Comprehension	MultiRC			F1a / EM
Recognizing Textual Entailment	RTE			Accuracy
Words in Context	WiC			Accuracy
The Winograd Schema Challenge	WSC			Accuracy
BoolQ	BoolQ			Accuracy
Reading Comprehension with Commonsense Reasoning	ReCoRD			F1 / Accuracy
Winogender Schema Diagnostics	AX-g			Gender Parity / Accuracy

Naïve Bayes – Linear or Non-Linear

Useful Discussions:

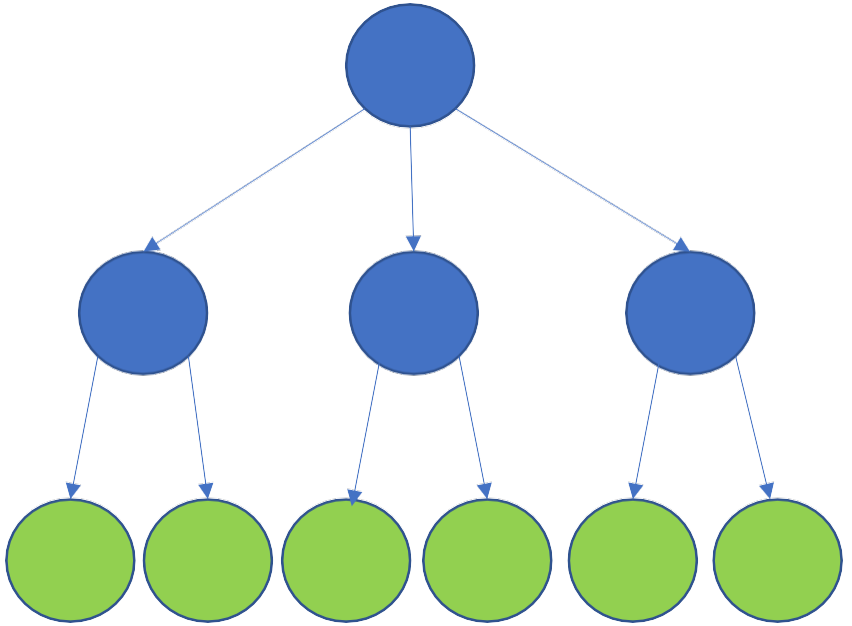
https://stats.stackexchange.com/questions/62816/why-is-naive-bayes-a-linear-model

https://stats.stackexchange.com/questions/142215/how-is-naive-bayes-a-linear-classifier

https://nlp.stanford.edu/IR-book/html/htmledition/linear-versus-nonlinear-classifiers-1.html

Decision Trees

- *Decision tree* is a classifier in the form of a tree structure, where each node is either:
  - *Decision node* - specifies some test to be carried out on a single attribute-value, with one branch and sub-tree for each possible outcome of the test
  - *Leaf node* - indicates the value of the target attribute (class) of examples
- A decision tree can be used to classify an example by starting at the root of the tree and moving through it until a leaf node, which provides the classification of the instance.



Decision Trees

- Class label denotes whether a tennis game was played

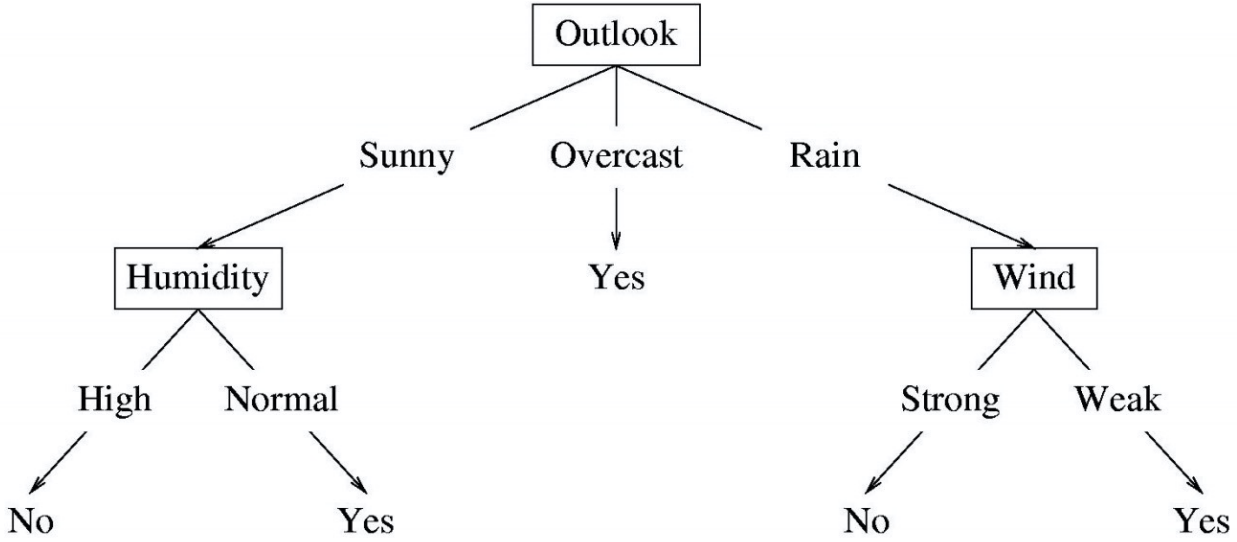
$\langle \boldsymbol{x}_i, y_i \rangle$

Predictors				Response
Outlook	**Temperature**	**Humidity**	**Wind**	**Class**
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Example from Eric Eaton, UPENN

Decision Trees (Examples)

- A possible decision tree for the data:

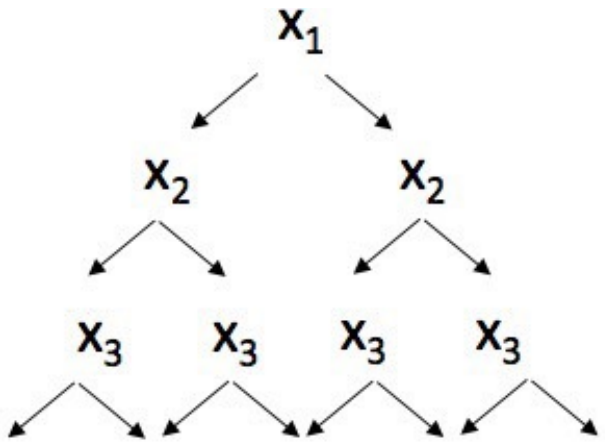


- Each internal node: test one attribute X_i
- Each branch from a node: selects one value for X_i
- Each leaf node: predict Y (or $p(Y \mid \boldsymbol{x} \in \text{leaf})$)

Example from Eric Eaton, UPENN

Decision Trees

x_1	x_2	x_3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

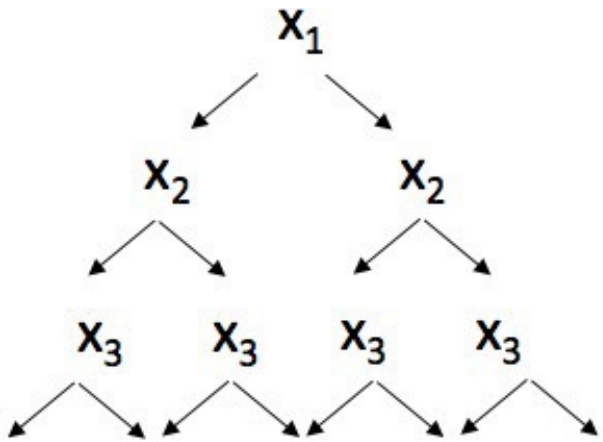


A tree can encode all possible combinations of feature values.

- Do you think decision tree learners explore all possible trees?

Decision Trees

x_1	x_2	x_3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

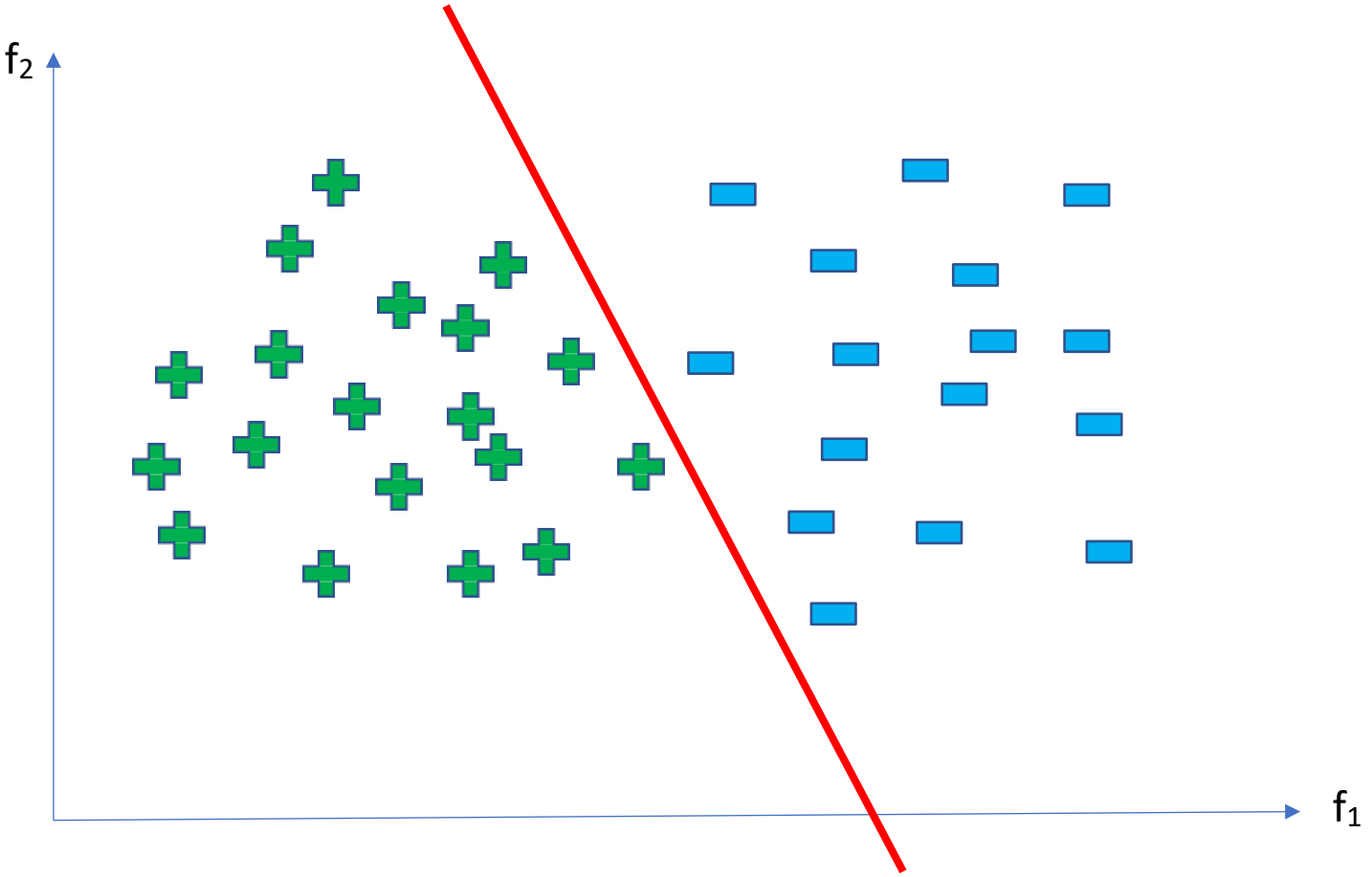


A tree can encode all possible combinations of feature values.

- Algorithm decides at each step what to split on next
- Usually greedy

What are these classifiers doing?

- An intuitive picture of the problem



Two features: f_1 and f_2
Two classes: + and -
Depiction of all examples in the training set.

What classifier to use?

- There may be a simple separator (e.g., a straight line in 2D or a hyperplane in general) or there may not
- There may be “noise” of various kinds
- There may be “overlap”
- Some classifiers explicitly represent separators (e.g., straight lines), while for other classifiers the separation is done implicitly
- Some classifiers just make a decision as to which class an object is in; others estimate class probabilities

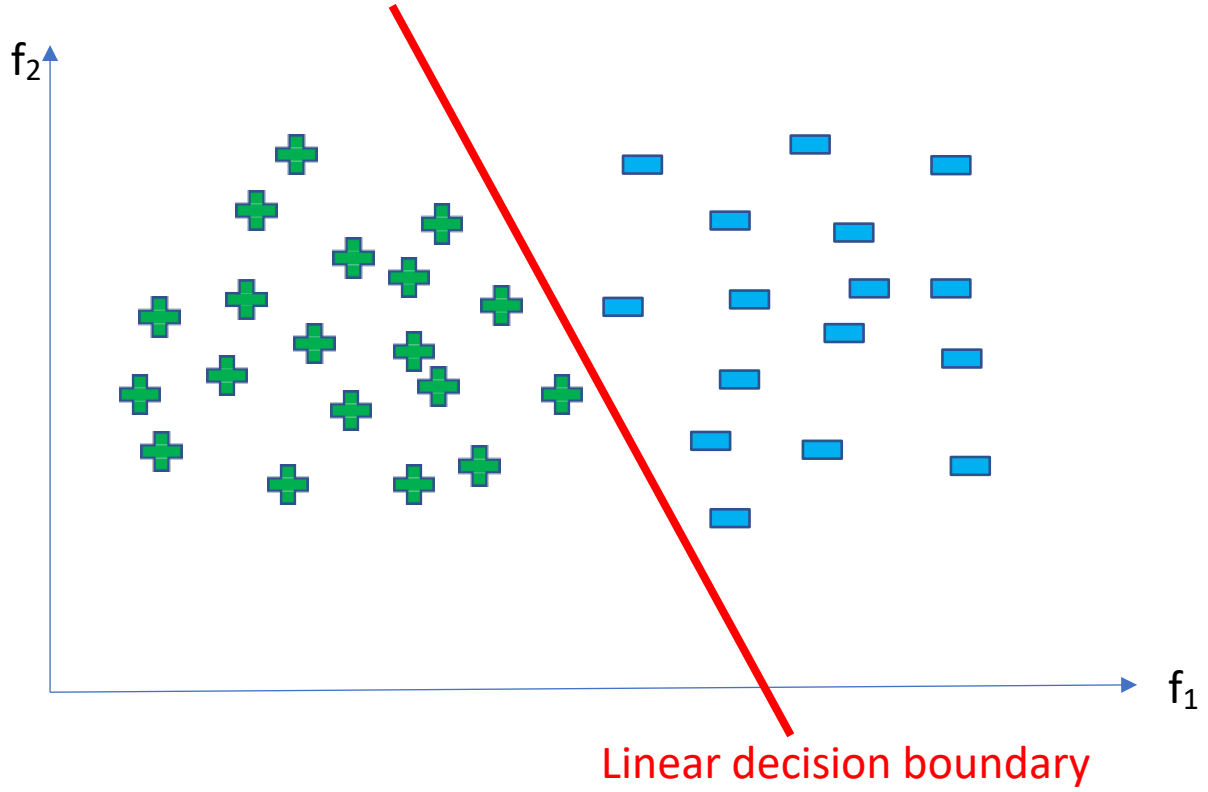
Many different classification methods

- **Probabilistic models**:
  - Naïve Bayes
- **Decision Models**:
  - Decision Trees
- **Linear Models**:
  - Support Vector Machine (SVM)
  - Linear regression
  - K-nearest neighbors

- And many many more

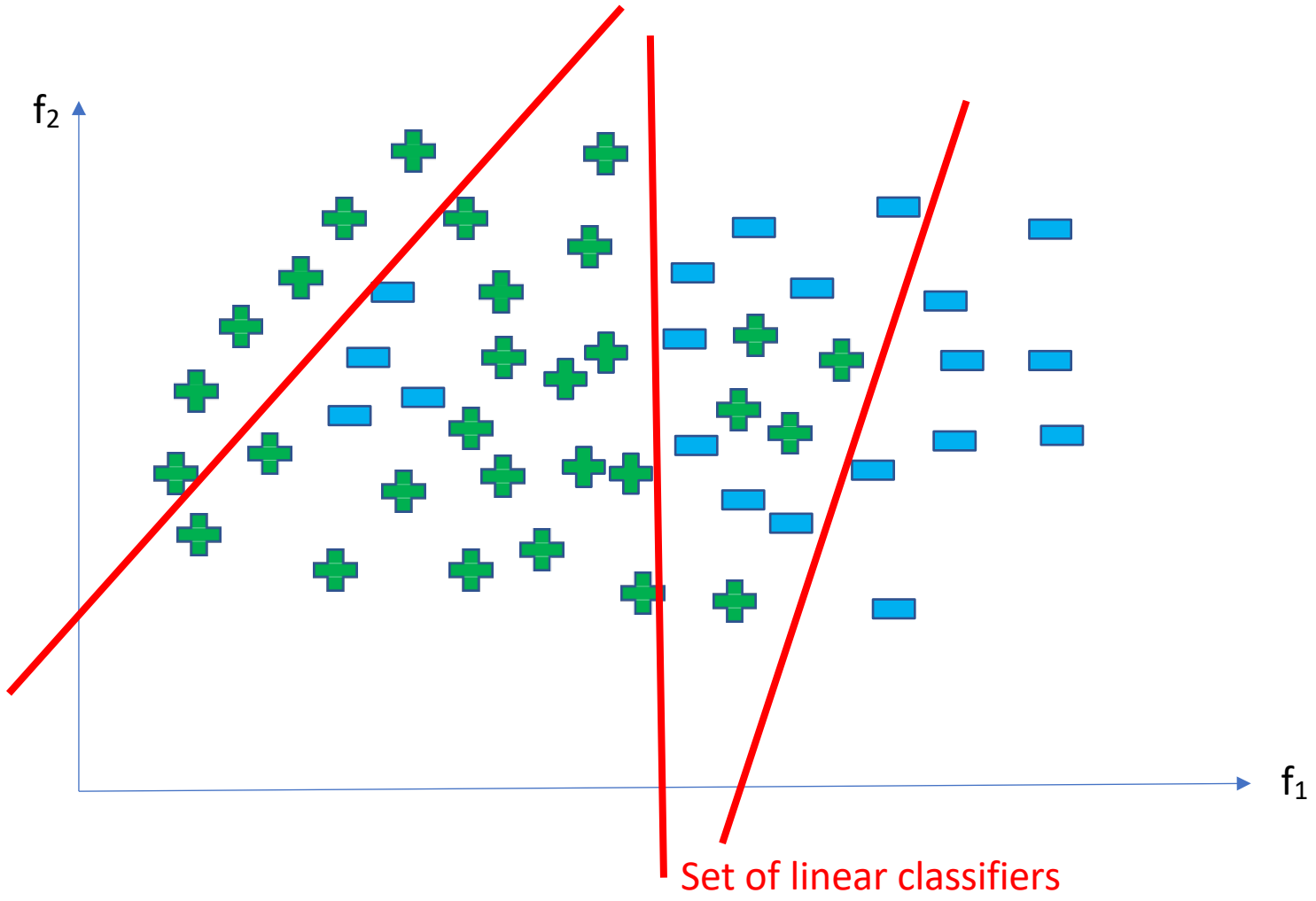
Linear versus non-linear algorithms

- **Linearly separable data**: if all the data points can be correctly classified by a linear (hyperplanar) decision boundary



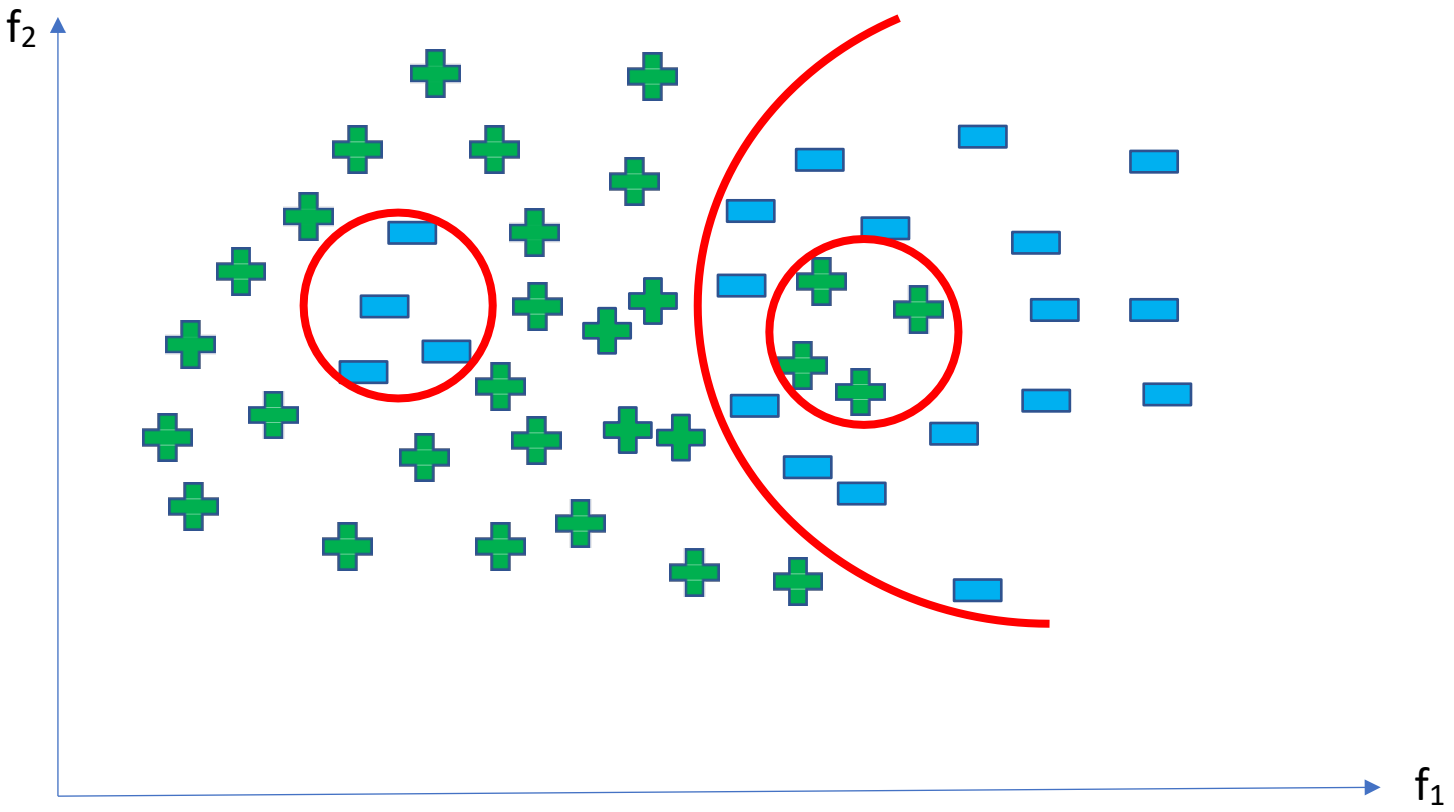
Two features: f_1 and f_2
Two classes: + and -
Depiction of all examples in the training set.

Non-linearly separable data



Two features: f_1 and f_2
Two classes: + and -
Depiction of all examples in the training set.

Non-linearly separable data

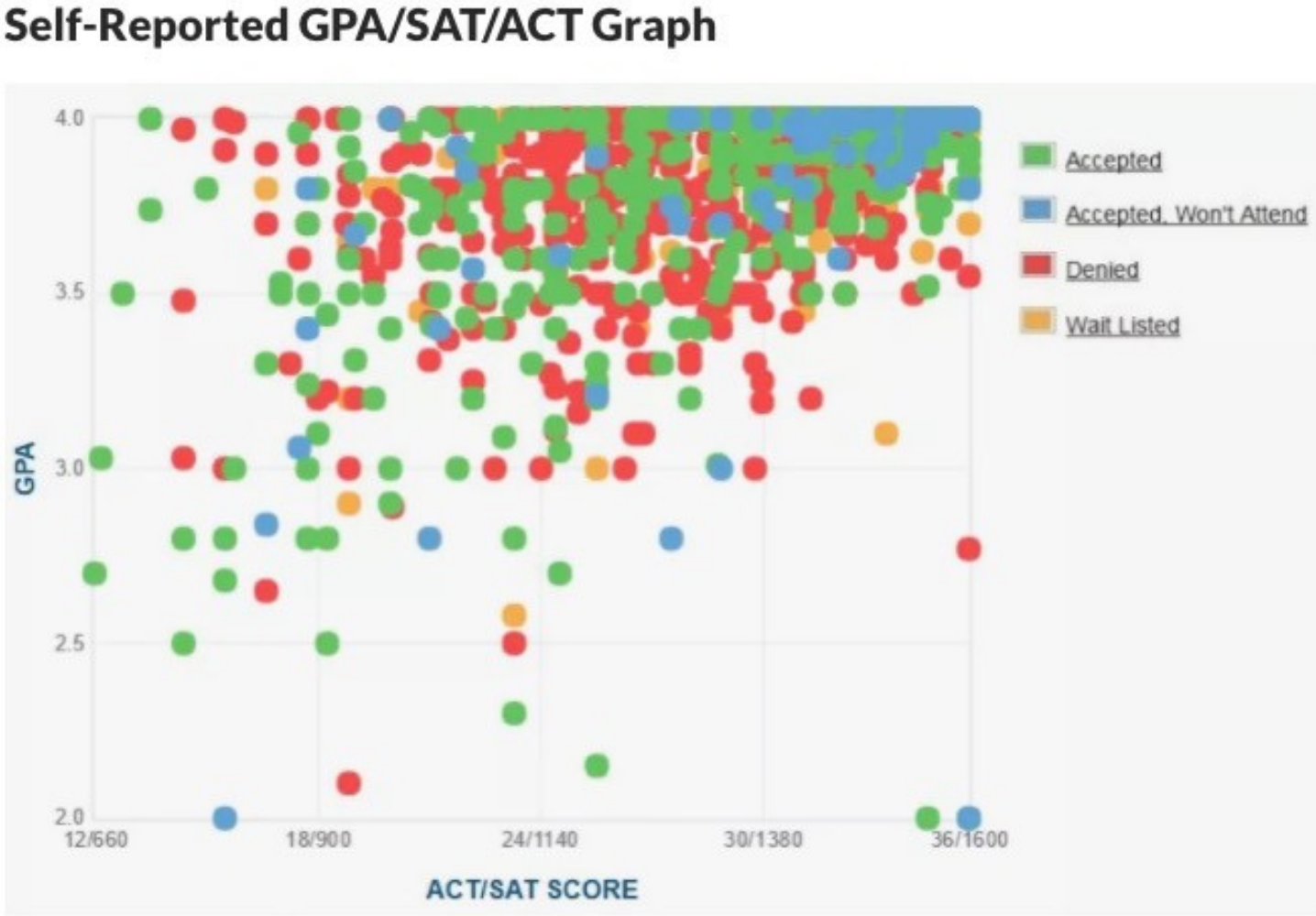


Non-linear classifiers

Two features: f_1 and f_2
Two classes: + and -
Depiction of all examples in the training set.

Example: College Admission

- Plot from Thoughtco.com



Harvard University Applicants' Self-Reported GPA/SAT/ACT Graph. Data courtesy of Cappex.

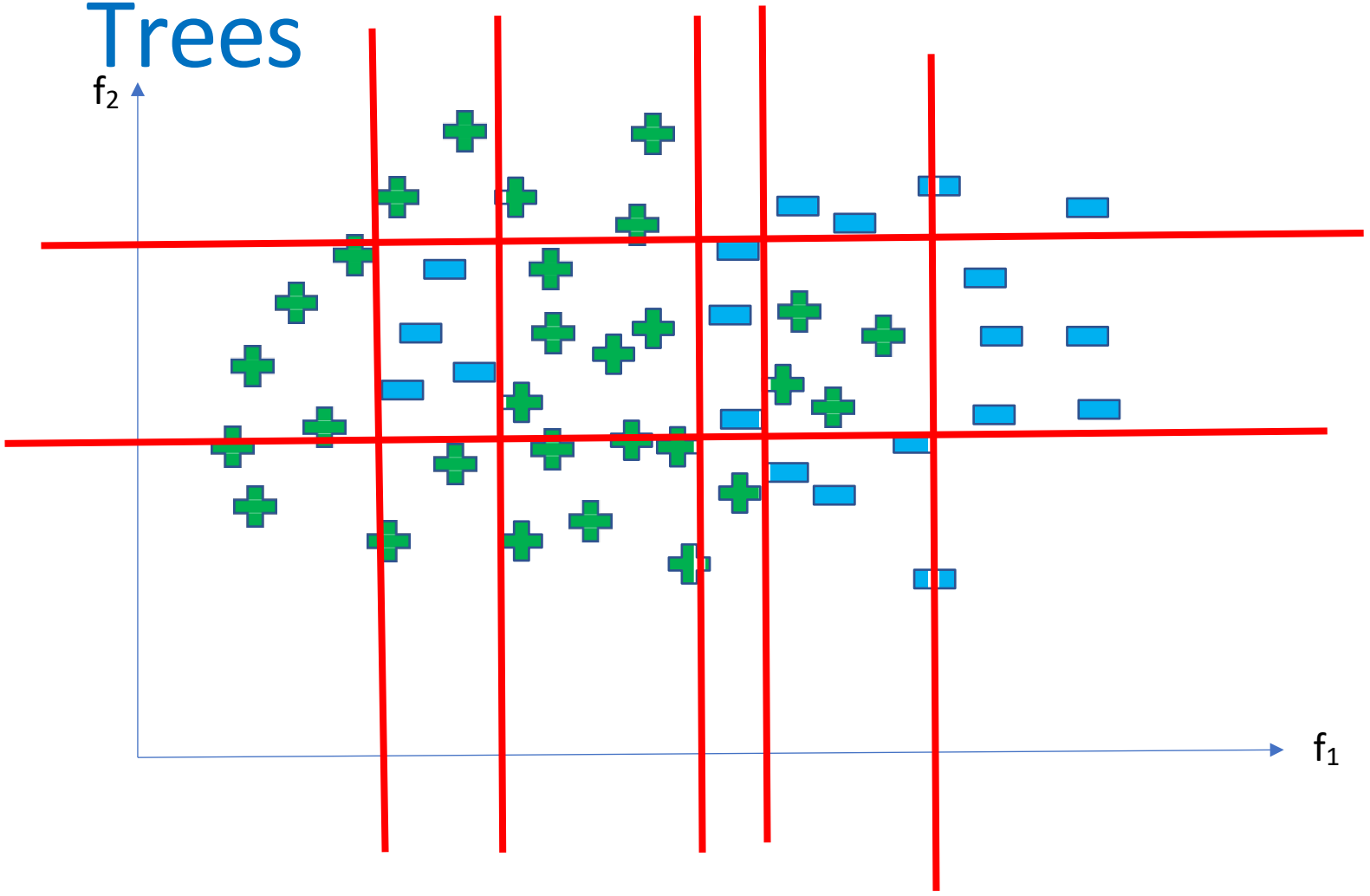
Linear versus Non-linear algorithms

- Linear or Non linear separable data?
  - We can find out empirically
- **Linear algorithms** (algorithms that find a linear decision boundary)
  - When we think the data is linearly separable
  - Advantages
    - Simpler, less parameters
  - Disadvantages
    - High dimensional data (like for NLP) is usually not linearly separable
  - Example: Linear Regression, SVM
  - Note: we can use linear algorithms also for non linear problems

Linear versus Non-linear algorithms

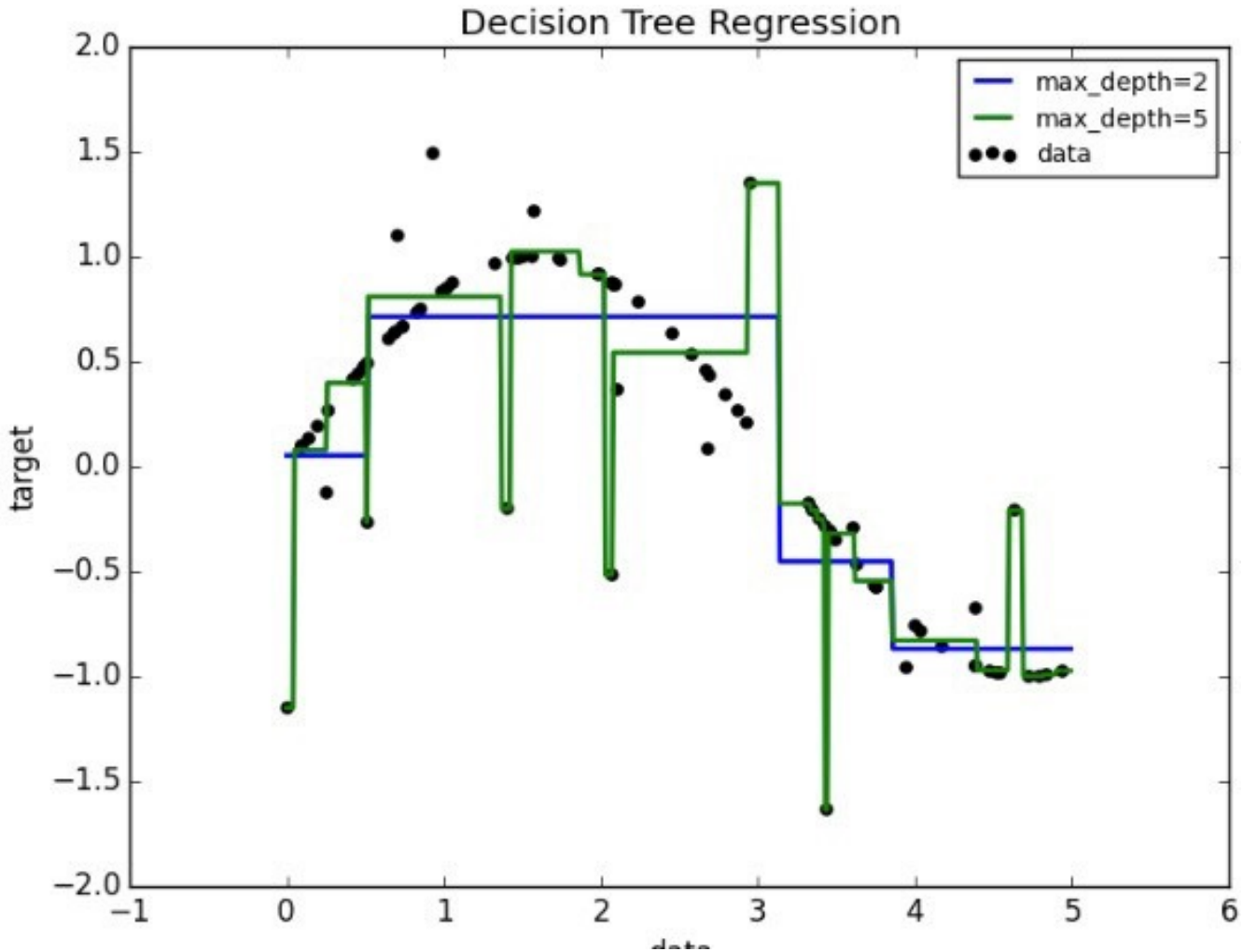
- **Non-linear algorithms**
  - When the data is non linearly separable
  - Advantages
    - More accurate
  - Disadvantages
    - More complicated, more parameters
  - Example: Decision Trees

Decision Trees



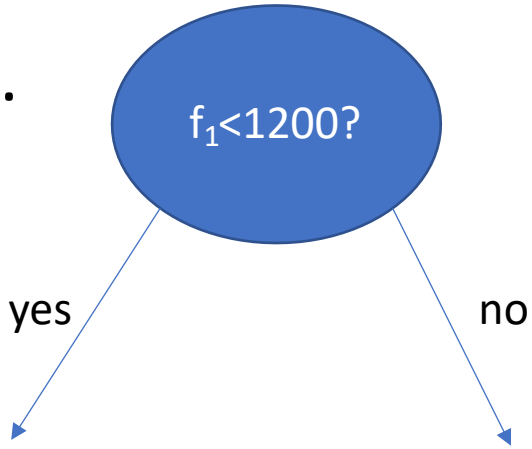
Two features: f_1 and f_2
Two classes: + and -
Depiction of all examples in the training set.

Decision tree can fit a nonlinear function



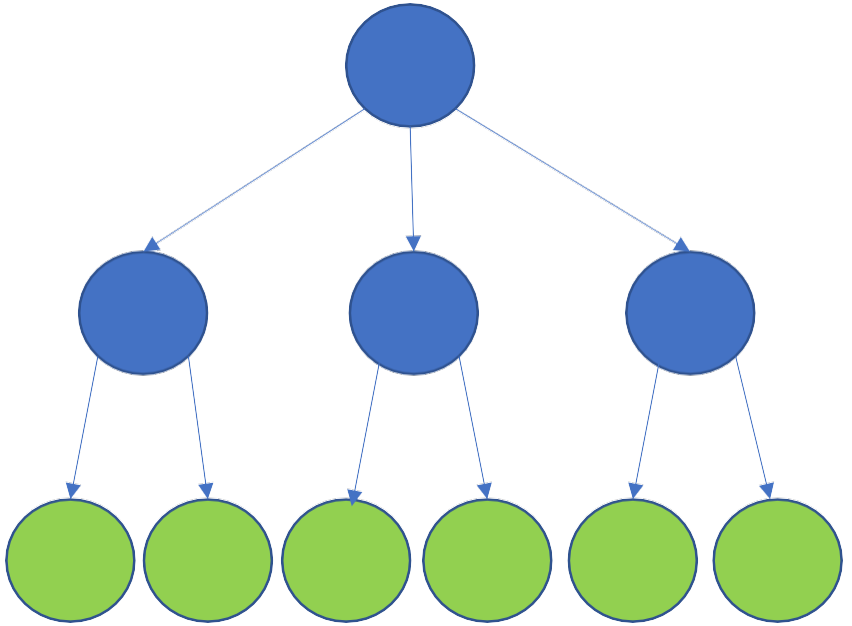
Building Decision Trees - Decision Stump

- Decision tree with a single node
- Decides how to classify examples with a single feature.
- Which feature should be used?
- Simplest solution:
  - Build a decision stump for each feature
  - Select according to accuracy on data



Building Decision Trees

- The central focus of the decision tree growing algorithm is selecting which attribute to test at each node in the tree. The goal is to select the attribute that is most useful for classifying examples.
- Top-down, **greedy** search through the space of possible decision trees.
  - That is, it picks the best attribute and never looks back to reconsider earlier choices.



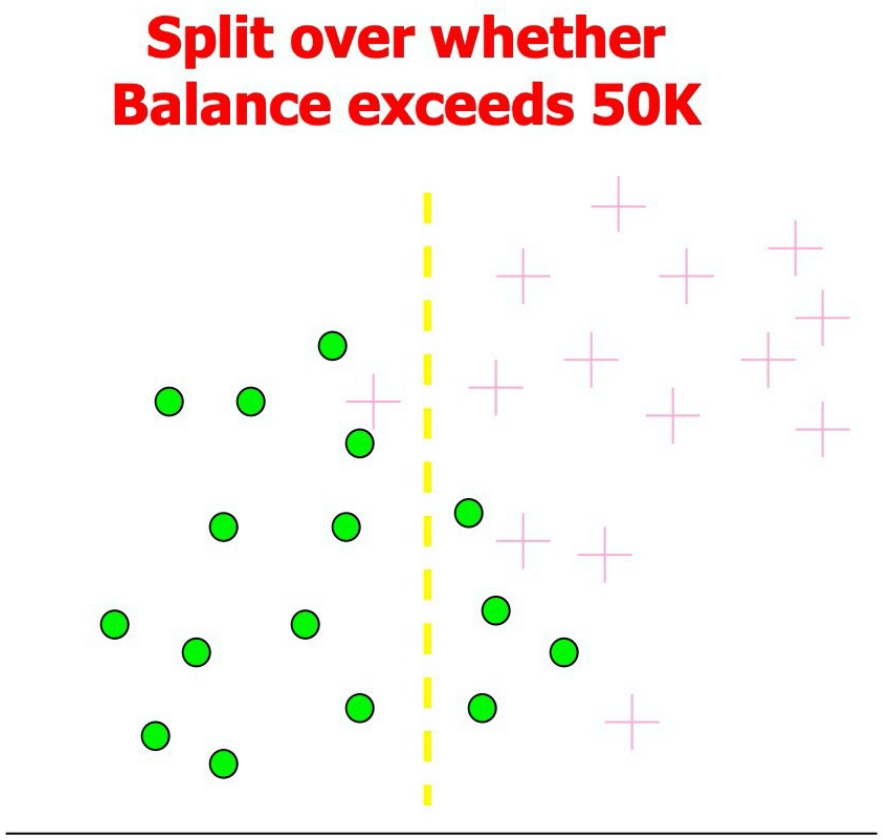
Building Decision Trees

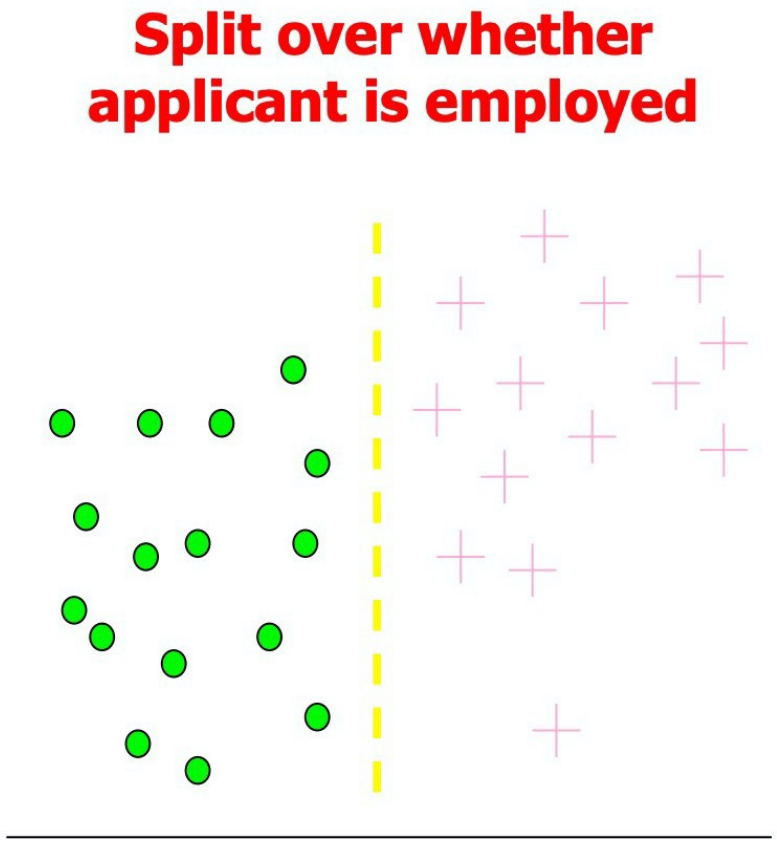
- Splitting criterion
  - Finding the features and the values to split on
    - for example, why test first "*past tense*" and not "negative emotion"?
  - Split that gives us the *maximum information gain* (or the *maximum reduction of uncertainty*)
- Stopping criterion
  - When all the elements at one node have the same class, no need to split further
- In practice, one first builds a large tree and then one prunes it back (to avoid overfitting)

Information Gain

Imagine you need to determine who a bank can safely give credit to?

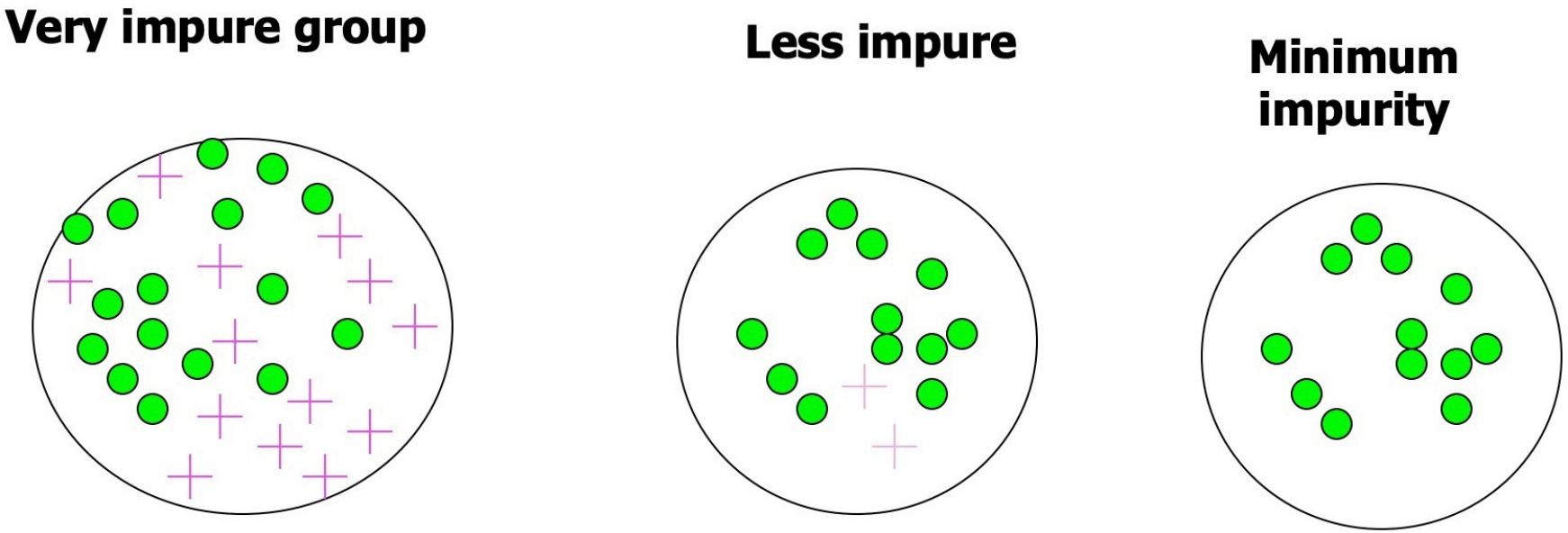
Which test is more informative?





Based on slide by Pedro Domingos, UW.

Impurity



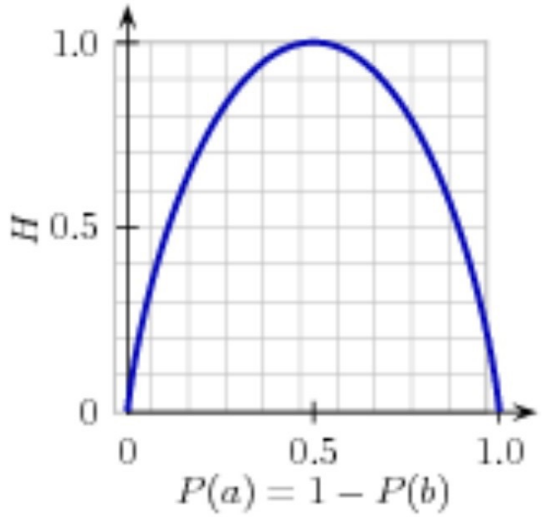
Based on slide by Pedro Domingos, UW.

Entropy: A way to measure impurity

- Entropy $H(X)$ of a random variable X:

$$H(X) = -\sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

Binary Classification example, classes are a and b:



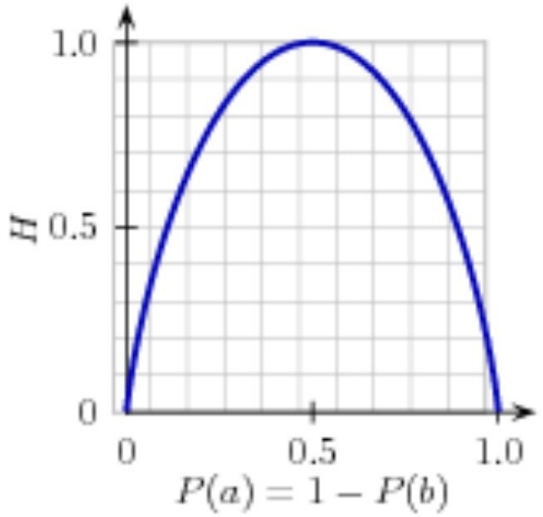
For a set of examples, P(a) is the proportion of examples of type a.

Entropy: A way to measure impurity

- Entropy $H(X)$ of a random variable X:

$$H(X) = -\sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

Binary Classification example, classes are a and b:



If P(a=0) => P(b) = 1

$H(X) = -(0 * \log_2 0 + 1 * \log_2 1) = 0$

Same for P(a=1)!

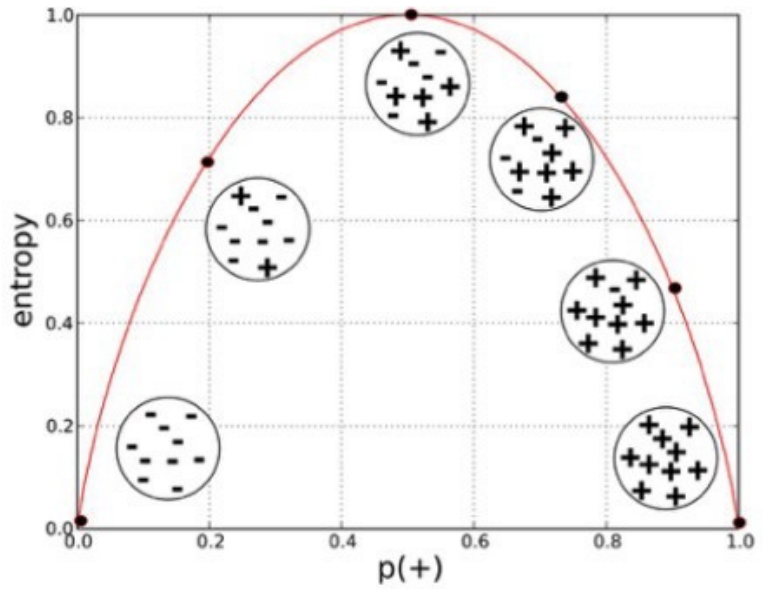
If P(a) = 0.5

$H(X) = -(½ * \log_2 ½ + ½ * \log_2 ½)$

$= 1 (½ * -1 + ½ * -1) = 1$

Entropy: A way to measure impurity

- Entropy $H(X)$ of a random variable X:



Provost, Foster; Fawcett, Tom. Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking

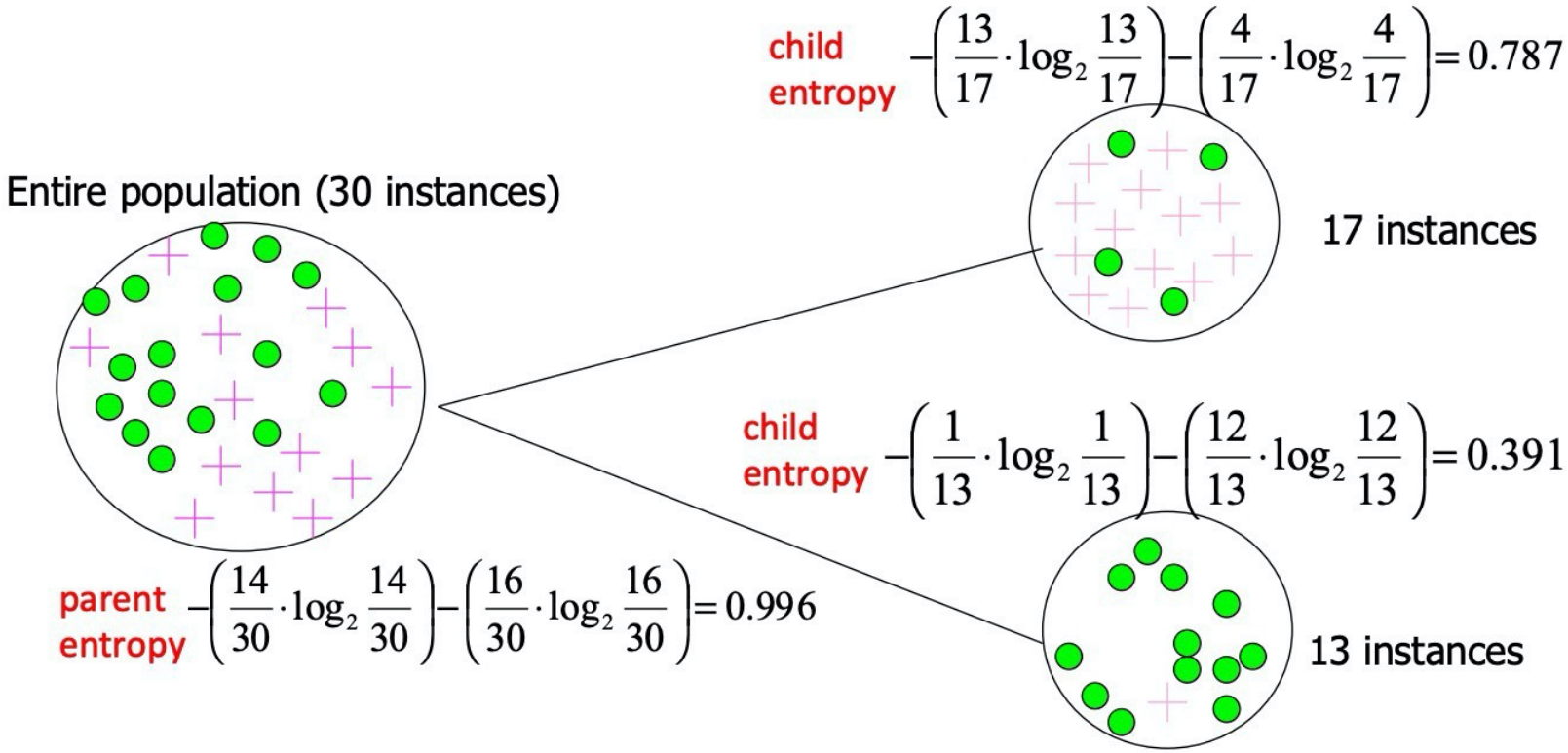
$$H(X) = -\sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

From entropy to information gain

- How much more organized the labels become once we apply the decision stump?
- We calculate the entropy for each of the decision stump's leaves, and take the average of those leaf entropy values (weighted by the number of samples in each leaf)
- The information gain is then equal to the original entropy minus this new, reduced entropy.

Calculating Information Gain

Information Gain = entropy(parent) – [average entropy(children)]



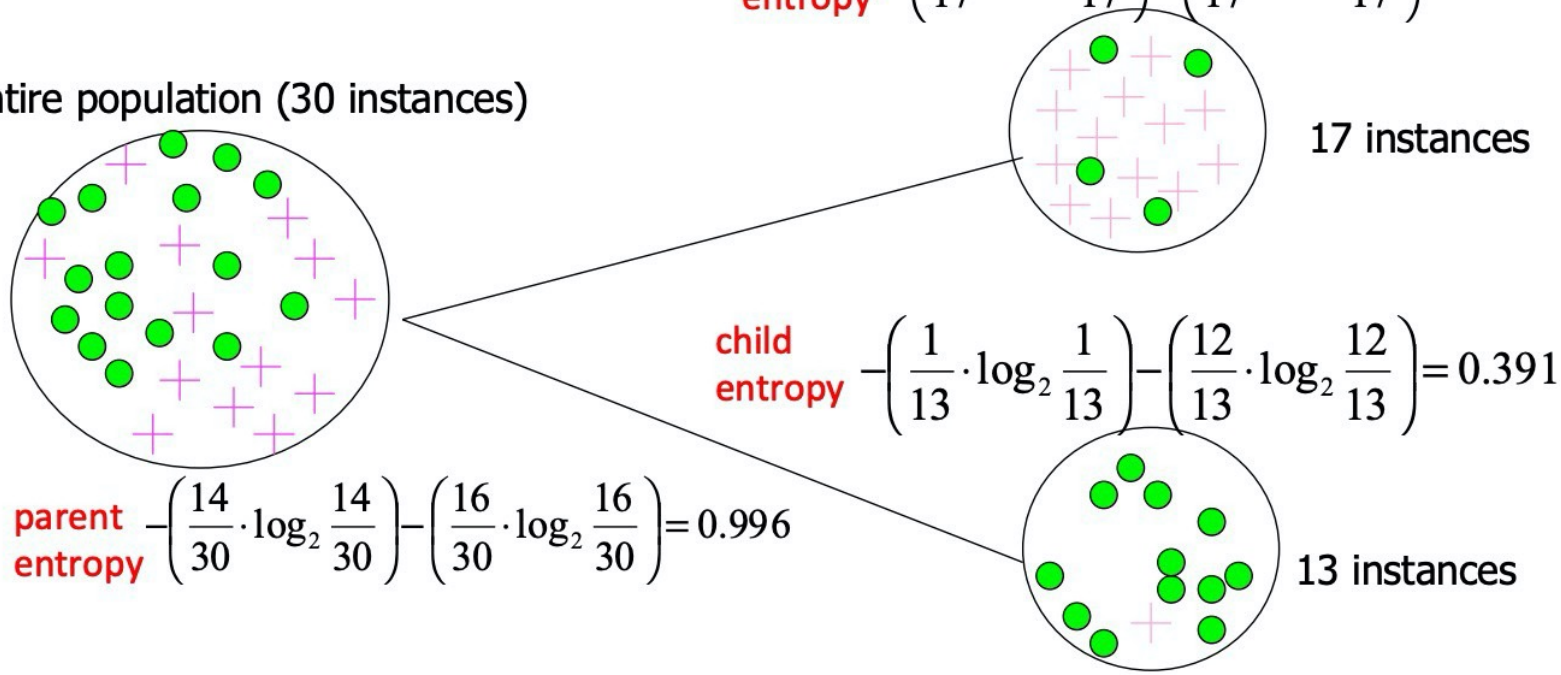
Based on slide by Pedro Domingos, UW.

Calculating Information Gain

Information Gain = entropy(parent) – [average entropy(children)]

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$

Entire population (30 instances)



child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$

parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

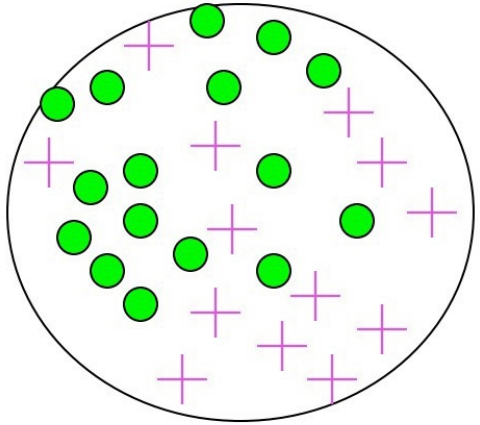
(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

Information Gain= 0.996 - 0.615 = 0.38

Based on slide by Pedro Domingos, UW.

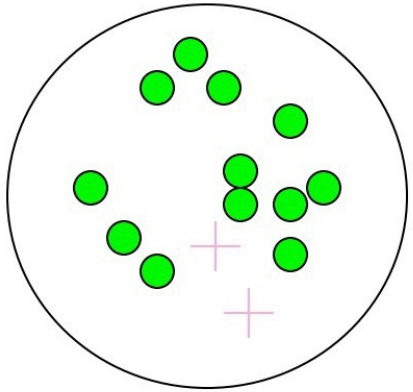
Impurity and Entropy

Very impure group



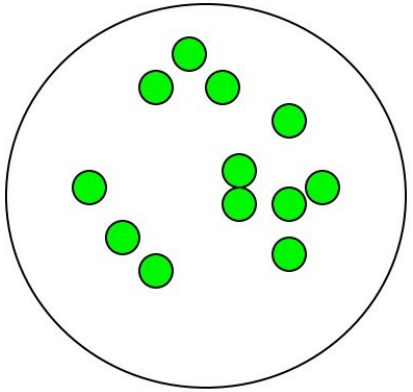
High Entropy

Less impure



Medium Entropy

Minimum impurity



Low Entropy

Decision Trees: Strengths

- Decision trees are able to generate understandable rules.
- Decision trees perform classification without requiring much computation.
- Decision trees are able to handle both continuous and categorical variables.
- Decision trees provide a clear indication of which features are most important for prediction or classification.

Decision Trees: Weaknesses

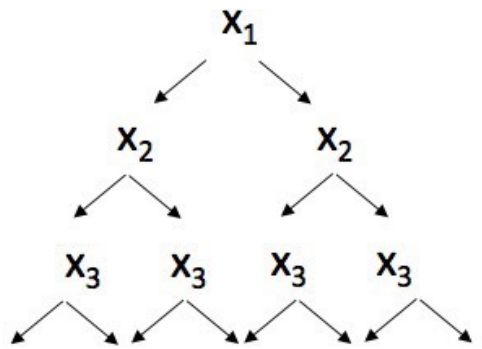
- Decision trees are prone to errors in classification problems with many classes and relatively small number of training examples.
  - Since each branch in the decision tree splits the training data, the amount of training data available to train nodes lower in the tree can become quite small.
- Decision tree can be computationally expensive to train.
  - Need to compare all possible splits

Decision Trees: Weaknesses

- Most decision-tree algorithms only examine a single field at a time. This leads to rectangular classification boxes that may not correspond well with the actual distribution of records in the decision space.
  - The fact that decision trees require that features be checked in a specific order limits their ability to exploit features that are relatively independent of one another.
  - Naive Bayes overcomes this limitation by allowing all features to act "in parallel."

Decision trees can easily overfit

x_1	x_2	x_3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



A tree can encode all possible combinations of feature values.

- Two common techniques to avoid overfitting:
  - Restrict the maximum depth of the tree
  - Restrict the minimum number of instances that must be remaining before splitting further
- If you stop creating a deeper tree before all instances at that node belong to the same class, use the majority class within that group as the final class for the leaf node.

Shallow Decision Tree

Shallow decision trees - trees that are too shallow might lead to overly simple models that can't fit the data.

A model that is underfit will have high training and high testing error. Hence, bad performance on training and test sets indicates underfitting which means the set of hypotheses are not complex enough (decision trees that are shallow) to include the true but unknown prediction function.

The shallower the tree the less variance we have in our predictions

Can we average the output of multiple decision trees?

Averaging out the predictions of multiple classifiers will drastically reduce the variance.

Averaging is not specific to decision trees; it can work with many different learning algorithms. But it works particularly well with decision trees.

Why averaging?

If two trees pick different features for the very first split at the top of the tree, then it's quite common for the trees to be completely different.

So decision trees tend to have high variance. To fix this, we can reduce the variance of decision trees by taking an average answer of a bunch of decision trees.

Pruning a Decision Tree?

The reason for pruning is that the trees prepared by the base algorithm can be prone to overfitting as they become incredibly large and complex.

Pruning is a technique in machine learning and search algorithms that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances.

Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting.

High Variance?

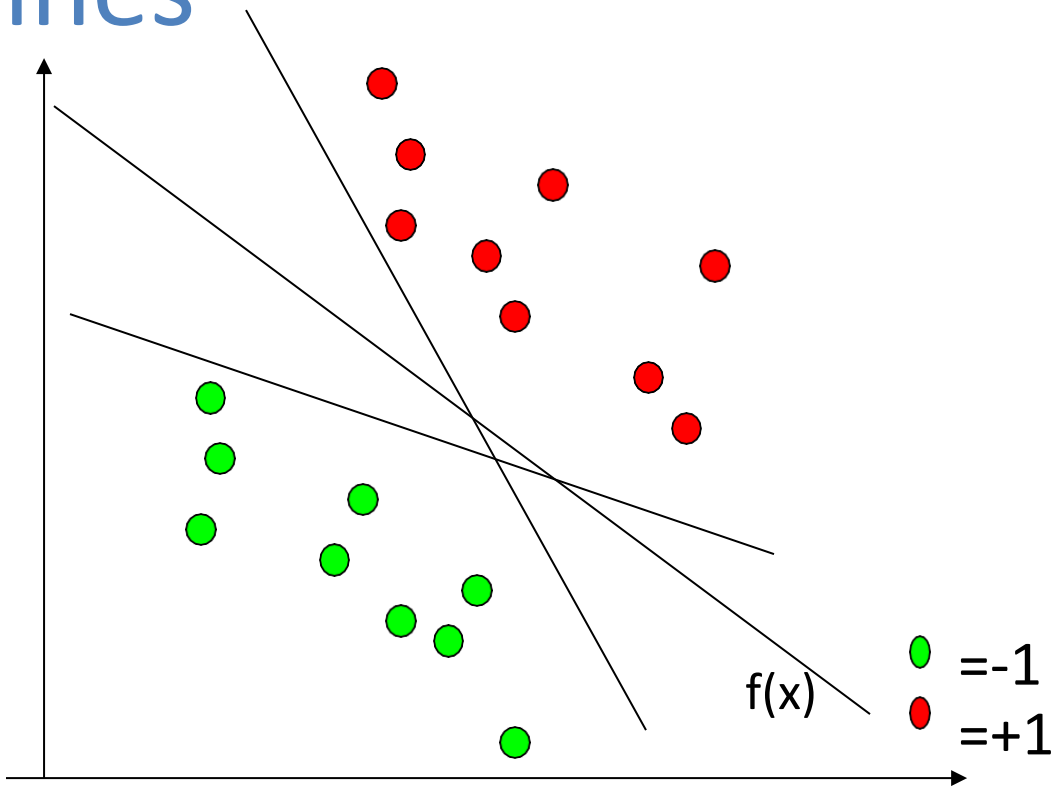
A model has high variance if it is very sensitive to (small) changes in the training data.

Decision trees are generally unstable considering that a small change in the data set can result in a very different set of splits. This results in high variance.

This is mainly due to the hierarchical nature of decision trees, since a change in split points in the initial stages will affect all the subsequent splits.

Support Vector Machines

Support Vector Machines



Data: $<\mathbf{x}_i, y_i>$, i=1,..,|D|

$\mathbf{x}_i \in R^d$

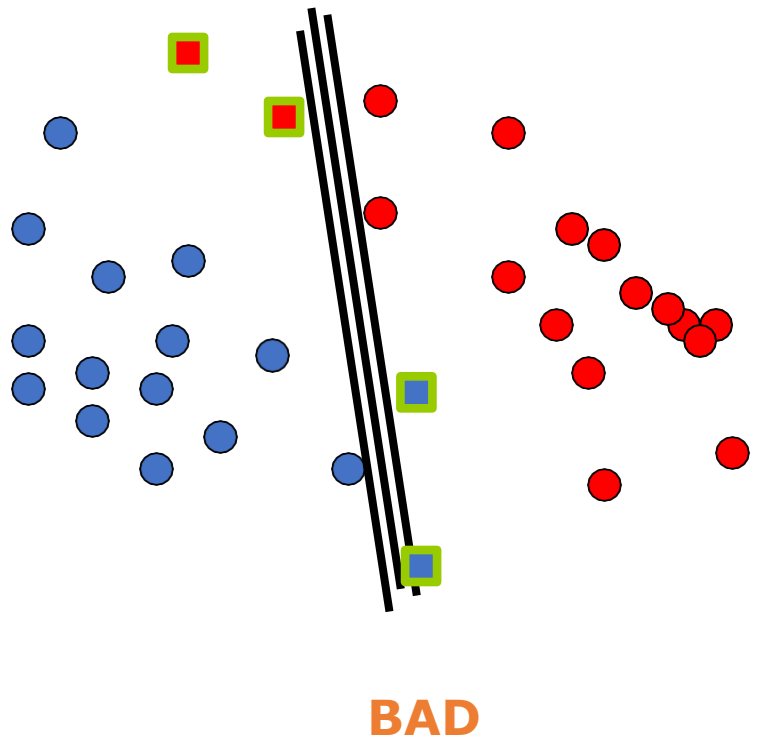
$y_i \in \{-1,+1\}$

All hyperplanes in R^d are parameterized by a vector ($\mathbf{w}$) and a constant b (bias). Hence, they can be expressed as $\mathbf{w} \bullet \mathbf{x} + b = 0$ (remember the equation for a hyperplane from algebra!)

Our aim is to find such a hyperplane $f(x) = sign(\mathbf{w} \bullet \mathbf{x} + b)$, that can correctly classify our data.

Support Vector Machines (SVM)

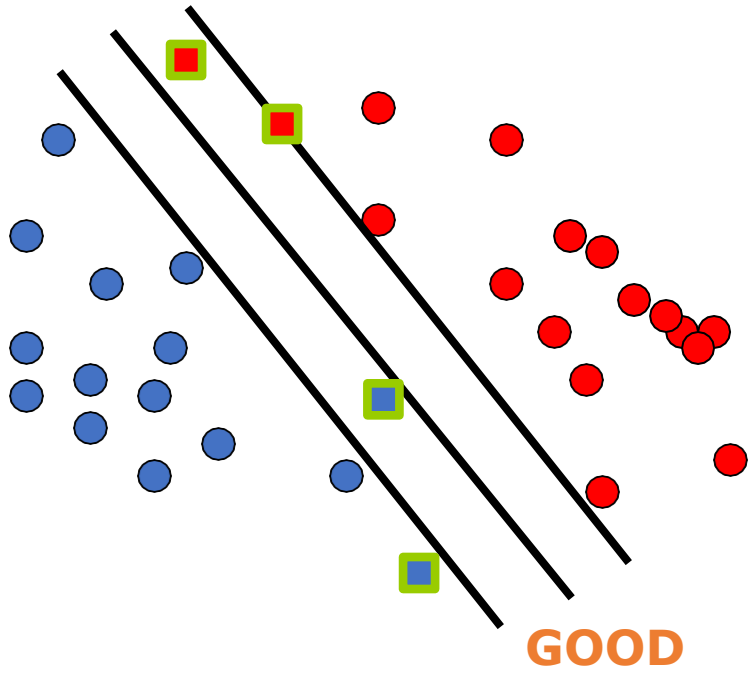
- **Another family of linear algorithms**
- **Intuition** (Vapnik, 1965)
- If the classes are linearly separable:
  - Separate the data
  - Place hyper-plane “far” from the data: **large margin**
  - Statistical results guarantee **good generalization**



From Gert Lanckriet, Statistical Learning Theory Tutorial

SVM: Large margin classifier

- **Another family of linear algorithms**
- **Intuition** (Vapnik, 1965)
- If the classes are linearly separable:
  - Separate the data
  - Place hyper-plane “far” from the data: **large margin**
  - Statistical results guarantee **good generalization**



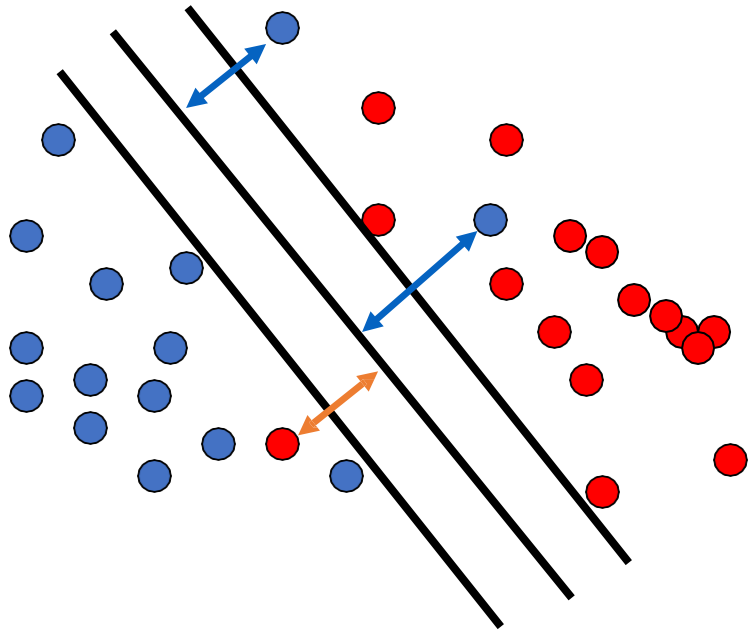
Maximal Margin Classifier

From Gert Lanckriet, Statistical Learning Theory Tutorial

SVM: Large margin classifier

If **not linearly separable**

- **Allow** some **errors**
- Still, try to place hyperplane “far” from each class

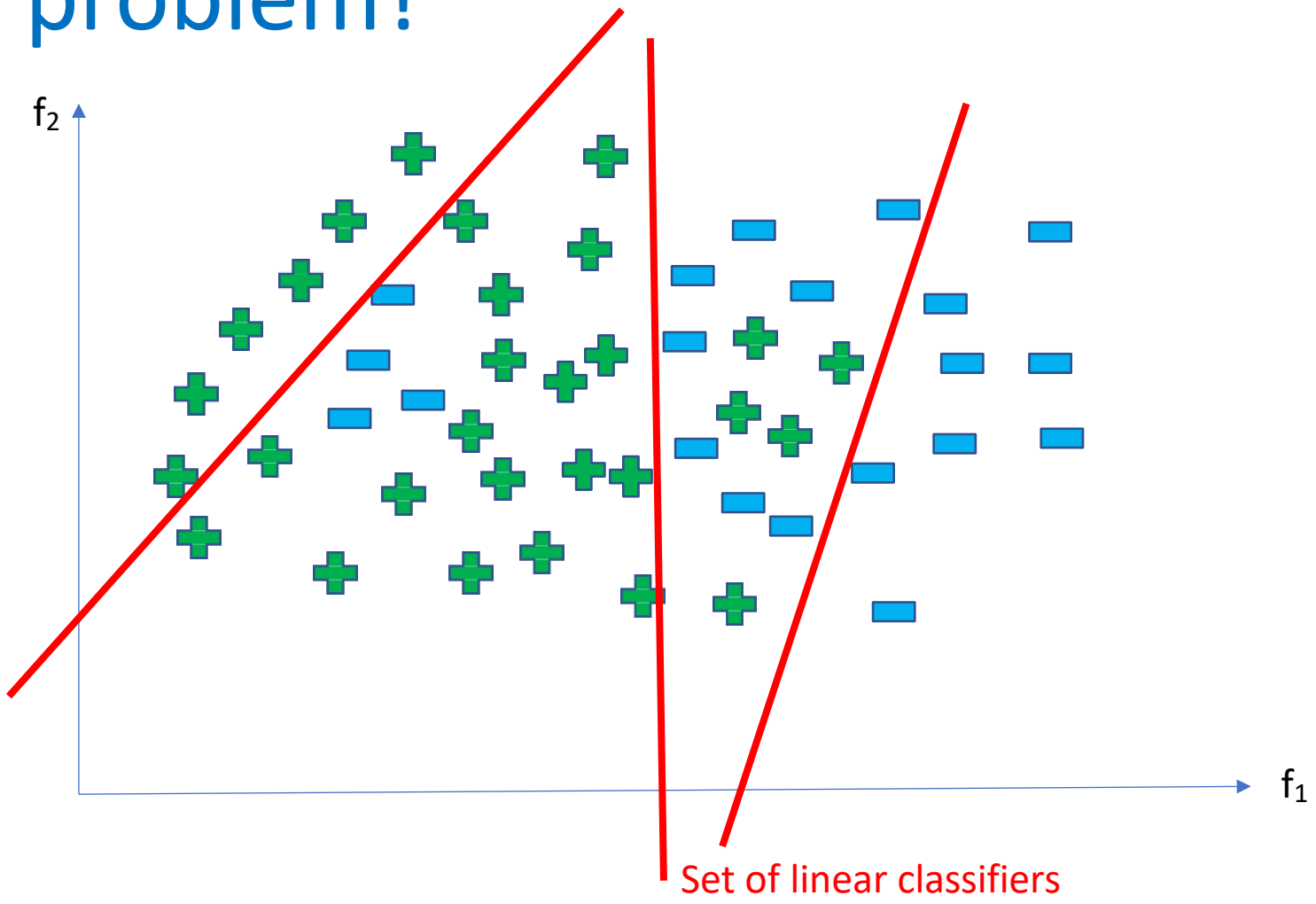


From Gert Lanckriet, Statistical Learning Theory Tutorial

SVM: Large Margin Classifiers

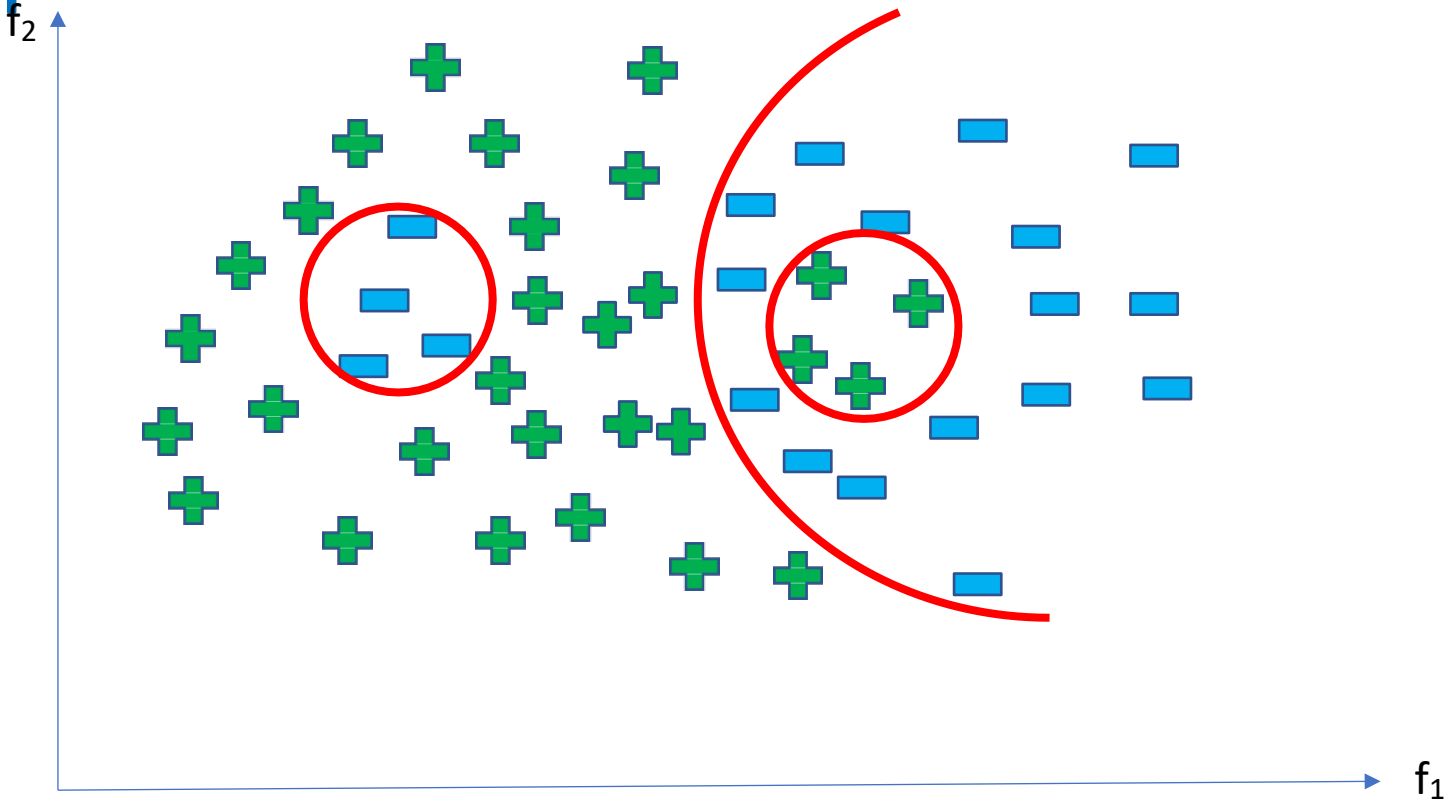
- Advantages
  - Theoretically better (better error bounds)
- Limitations
  - Computationally more expensive, large quadratic programming

So, what if it's a non-linear problem?



Two features: f_1 and f_2
Two classes: + and -
Depiction of all examples in the training set.

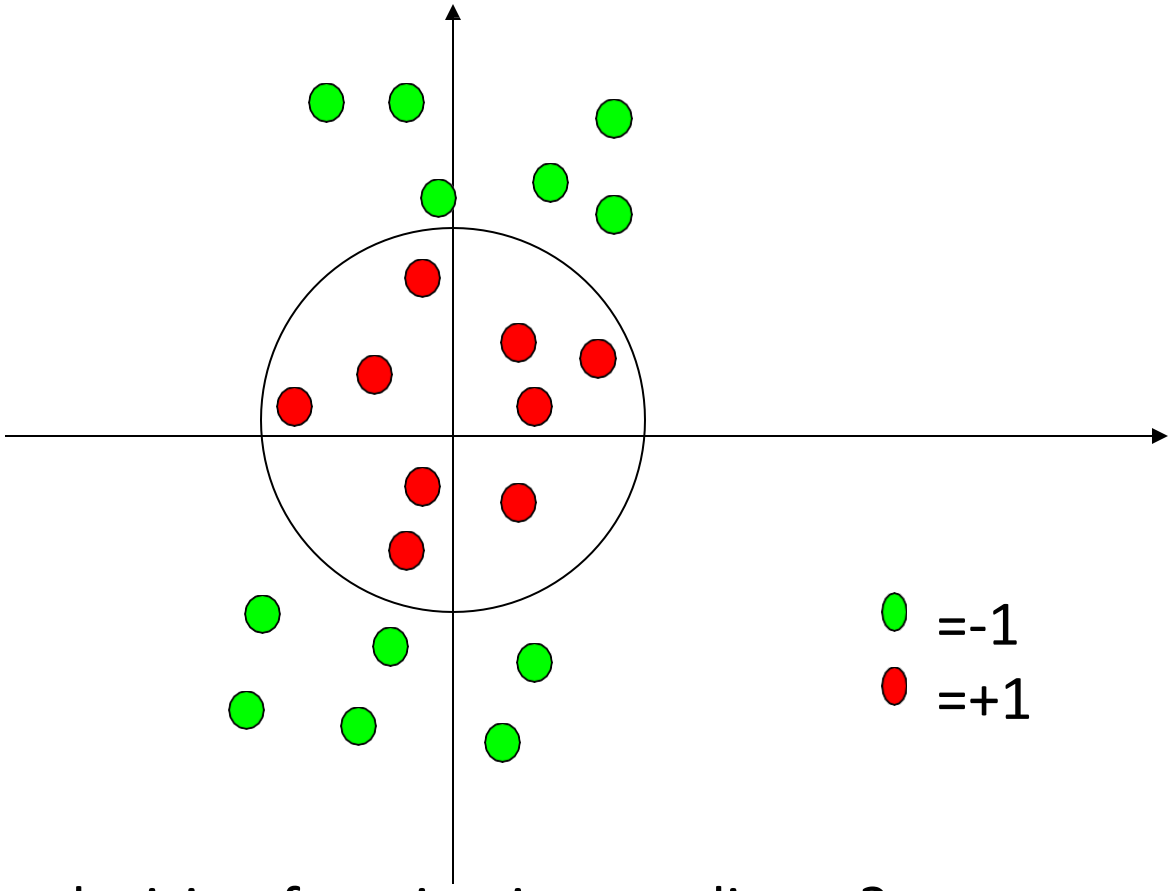
So, what if it's a non-linear problem?



Non-linear classifiers

Two features: f_1 and f_2
Two classes: + and -
Depiction of all examples in the training set.

So, what if it's a non-linear problem?

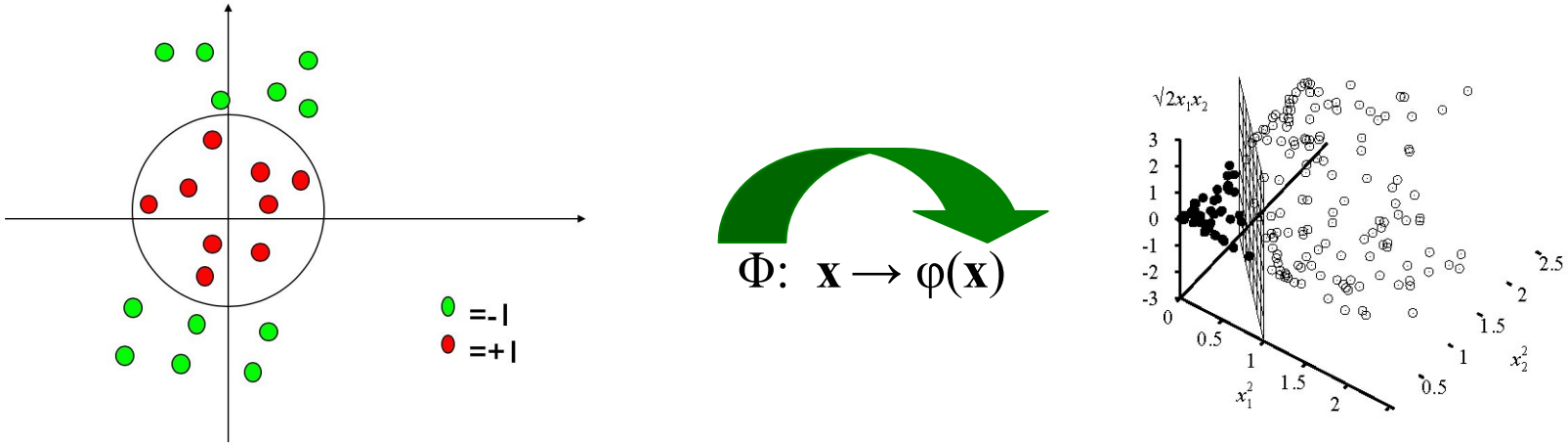


What if the decision function is not a linear?

Slide from Pierre Dönnes.

Kernel Trick

General idea: The original feature space can always be mapped to some higher-dimensional feature space where the training set is separable.



Slide from Pierre Dönnes.

So, what if it's a non-linear problem?

- Kernel methods: A family of non-linear algorithms
- Transform the non linear problem in a linear one (in a different feature space)
- Use linear algorithms to solve the linear problem in the new space
  - There are SVM learners in Scikit-Learn that have a kernel that does this kind of transform
  - http://scikit-learn.org/stable/modules/svm.html - svm-kernels
  - classifier = SklearnClassifier(svm.SVC()).train(train_sets)
  - class sklearn.svm.SVC(C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None)

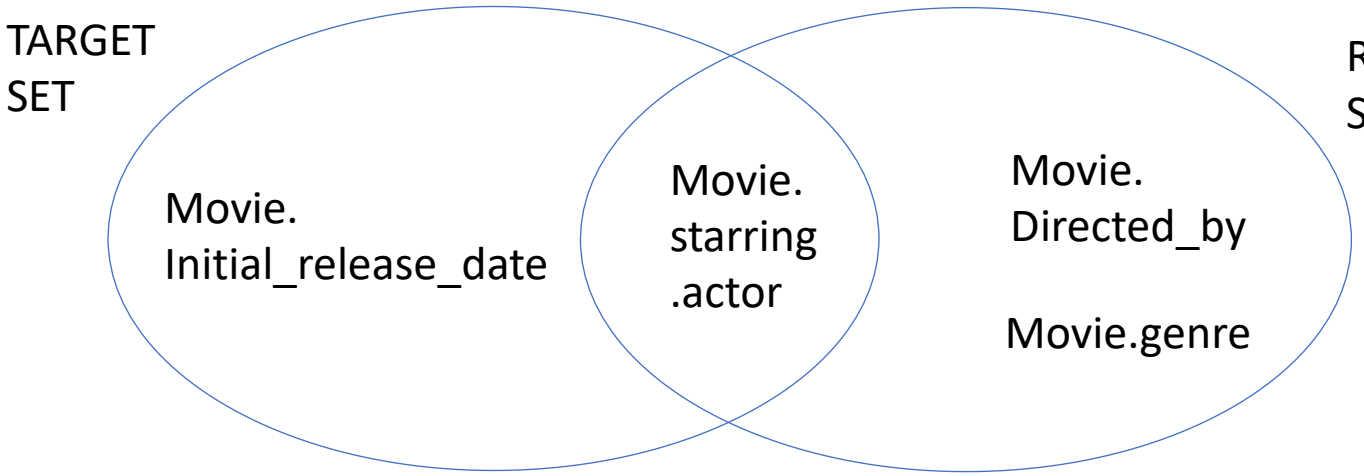
From Gert Lanckriet, Statistical Learning Theory Tutorial

Evaluation – F1-measure

- Harmonic mean of recall and precision
- For example, for the example utterance:

  What movies has Tom Cruise has been in recently

- If the classifier returns: movie.starring.actor, movie.genre, and movie.directed_by



True Positives = {movie.starring.actor}
False Positives = {movie.directed_by, movie_genre}
False Negatives = {movie.initial_release_date}

Evaluation – F1-measure (cont.)

- In NLP we are always dealing with these kinds of errors.
- Reducing the error rate for an application often involves two antagonistic efforts:
  - Increasing precision (minimizing false positives)
  - Increasing recall (minimizing false negatives)

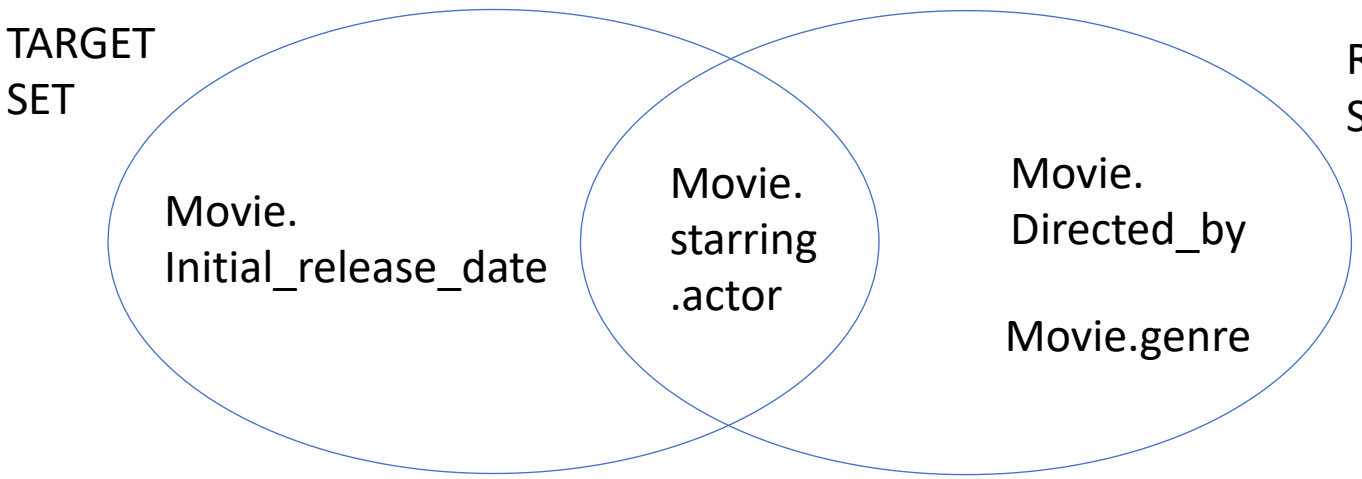
- Precision = TP / TP + FP
- Recall = TP / TP + FN
- F1 = (2 X Precision X Recall) / (Precision + Recall)

Evaluation – F1-measure (cont.)

Recall = ½ = 0.50

Precision = 1/3 = 0.33

F1 = 0.40



True Positives = {movie.starring.actor}

False Positives = {movie.directed_by, movie_genre}

False Negatives = {movie.initial_release_date}

Nearest Neighbor Methods

- Simplest
- Instance-based learning (also called memory-based learning)
  - Similar examples have the same label
  - We can classify an example with the label of a similar one.

- Image processing example:

→ cat

Example training set

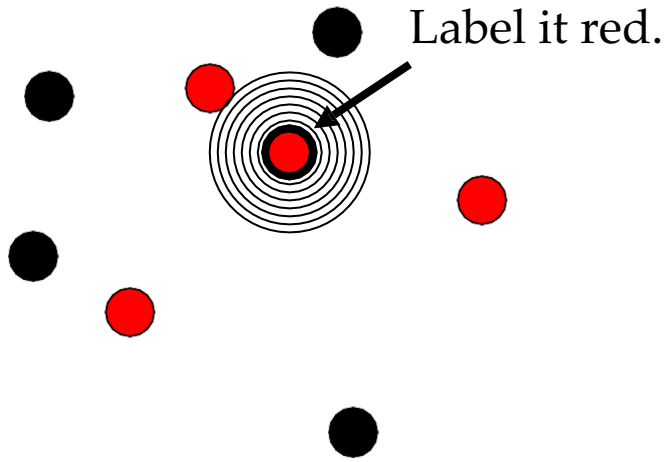
Images from CS231n by Fei-fei Li, Andrej Karpathy, and Justin Johnson

Nearest Neighbor Methods

- Algorithm:
  - Given a new example x for which we need to predict its label y
  - Find the most similar training examples
  - Classify x “like” these most similar examples
- **Lazy** learning algorithm
  - Processing of training examples postponed until inference time
  - Training ≈ gathering and storing examples

1-Nearest Neighbor

- Use only one neighbor
- Label an example with the label of the nearest example



Algorithm

- **Training:**

  for i = 1, ..., n in the training dataset:

  store training examples (x_i, y_i)

- **Inference:** label x_q

  closest point := None

  closest distance := ∞

  for i = 1, ..., n: # sequential search

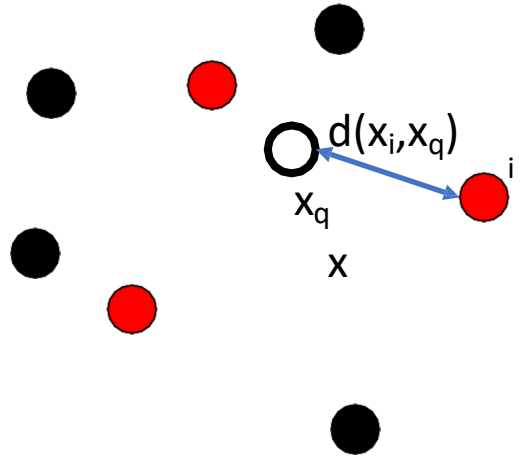
  current distance := $d(x_i, x_q)$

  if current distance < closest distance:

  closest distance := current

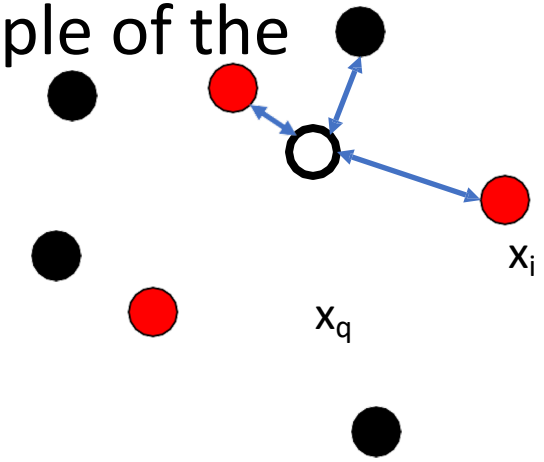
  distance closest point := x_i

  assign x_q is the label of closest point



k-Nearest Neighbors (kNN)

- Make the labeling decision based on k-nearest neighbors.
- Assign to a test sample the majority category label of its *k* nearest training samples
- In practice, *k* is usually chosen to be odd (for binary classification), so as to avoid ties
- For multi-class classification, k shouldn't be a multiple of the number of classes.



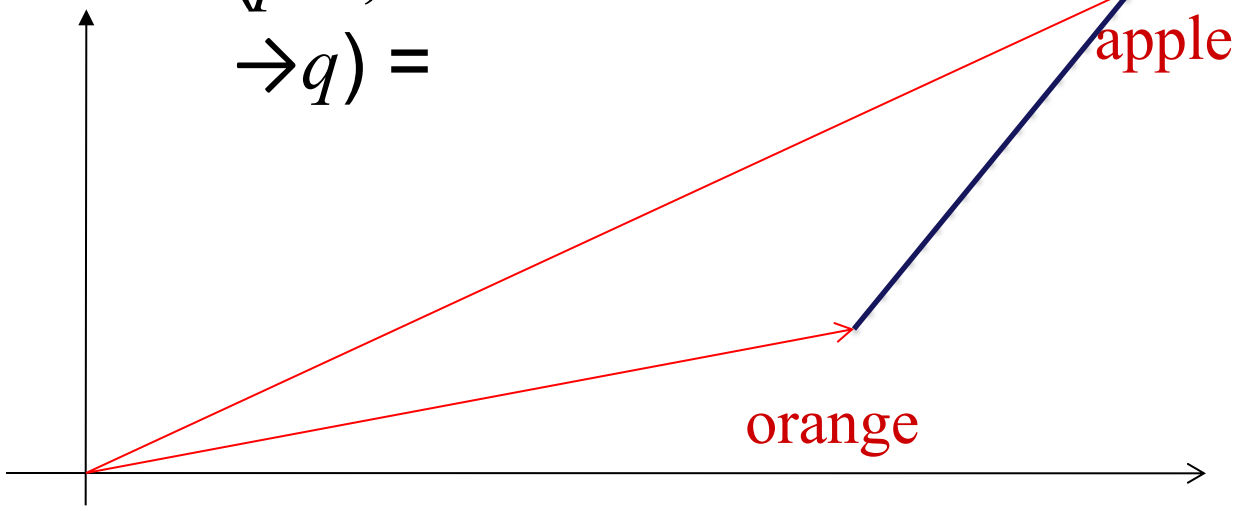
Questions

- How to represent examples?
- How to determine similarity?
- How many “similar training examples” to include?
- How to resolve inconsistencies between the training examples?

Euclidean (L2) Distance

- a dissimilarity measure:

$dist\ (p\rightarrow, \rightarrow q) =$



Distance

Minkowsky:

$$D(\boldsymbol{x},\boldsymbol{y}) = \left(\sum_{i=1}^{m} |x_i - y_i|^r\right)^{1/r}$$

Euclidean:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

Manhattan / city-block:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} |x_i - y_i|$$

Camberra:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i + y_i|}$$

Chebychev:

$$D(\boldsymbol{x},\boldsymbol{y}) = \max_{i=1}^{m} |x_i - y_i|$$

Quadratic:

$$D(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T Q(\boldsymbol{x} - \boldsymbol{y}) = \sum_{j=1}^{m}\left(\sum_{i=1}^{m}(x_i - y_i)q_{ji}\right)(x_j - y_j)$$

Q is a problem-specific positive definite $m \times m$ weight matrix

Mahalanobis:

$$D(\boldsymbol{x},\boldsymbol{y}) = [\det V]^{1/m}(\boldsymbol{x} - \boldsymbol{y})^{\mathrm{T}} V^{-1}(\boldsymbol{x} - \boldsymbol{y})$$

V is the covariance matrix of $A_1..A_m$, and A_j is the vector of values for attribute j occuring in the training set instances $1..n$.

Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{\sum_{i=1}^{m}(x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_{i=1}^{m}(x_i - \overline{x_i})^2 \sum_{i=1}^{m}(y_i - \overline{y_i})^2}}$$

$\overline{x_i} = \overline{y_i}$ and is the average value for attribute i occuring in the training set.

Chi-square:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{1}{sum_i}\left(\frac{x_i}{size_{\boldsymbol{x}}} - \frac{y_i}{size_{\boldsymbol{y}}}\right)^2$$

sum_i is the sum of all values for attribute i occuring in the training set, and $size_{\boldsymbol{x}}$ is the sum of all values in the vector $\boldsymbol{x}$.

Kendall's Rank Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = 1 - \frac{2}{n(n-1)}\sum_{i=1}^{m}\sum_{j=1}^{i-1}\mathrm{sign}(x_i - x_j)\mathrm{sign}(y_i - y_j)$$

$\mathrm{sign}(x)$=-1, 0 or 1 if $x < 0$, $x = 0$, or $x > 0$, respectively.

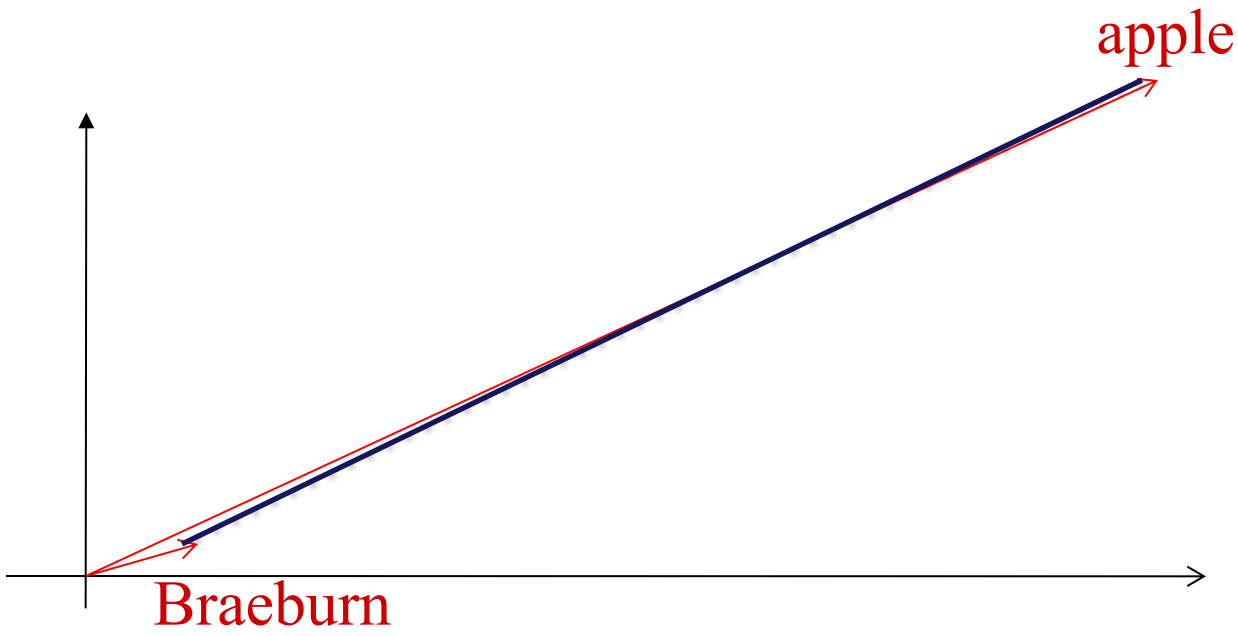
Figure 1. Equations of selected distance functions. ($\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of m attribute values).

Possible Issues:

- Most distance measures were designed for linear/real-valued attributes
- How to best handle discrete/nominal values?

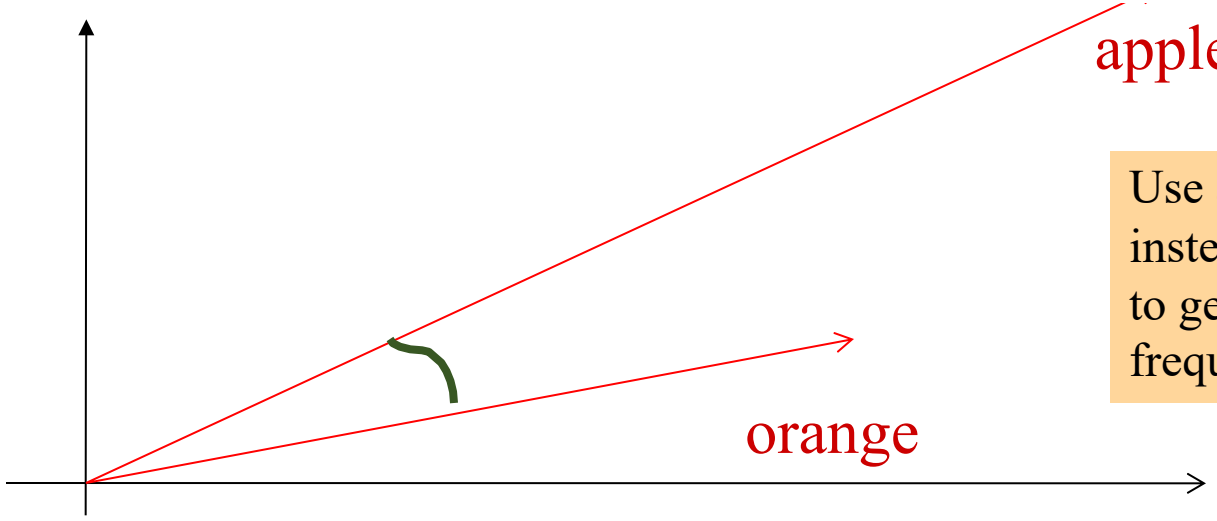
Euclidean (L2) Distance

- Problem: If features are context word counts, very sensitive to word frequency!



Cosine Similarity

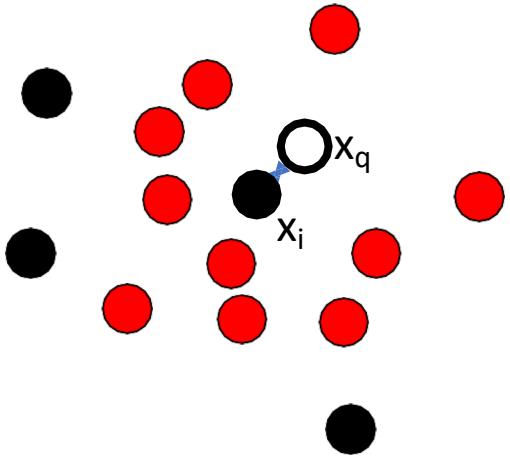
$$\text{cosine similarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$



Use angle between vectors instead of point distance to get around word frequency issues

Deciding on k

- Large k:
  - Makes KNN less sensitive to noise
- Small k:
  - Allows capturing finer structure of space



Pick k not too large, but not too small (depends on data)

Curse of dimensionality

- Prediction accuracy can quickly degrade when number of attributes grows.
  - Irrelevant attributes easily “swamp” information from relevant attributes
  - When many irrelevant attributes, similarity/distance measure becomes less reliable

- Remedy
  - Try to remove irrelevant attributes in pre-processing step
  - Weight attributes differently
  - Increase k (but not too much)

kNN Pros and Cons

- **Storage:** all training examples are stored in memory
  - A decision tree or linear classifier is much smaller
- **Time:** to classify *x*, you need to loop over all training examples *(x',y')* to compute distance between *x* and *x'*.
  - You get predictions for every class *y*
    - kNN is nice when there are many many classes
  - Actually, there are some tricks to speed this up…especially when data is sparse (e.g., text)
    - Need distance/similarity measure and attributes that "match" target function.

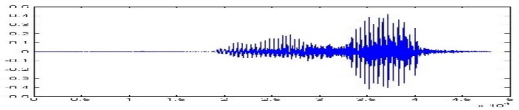
Other Classification Methods

- Include
  - Random Forests
  - Boosting
  - Maximum Entropy

Linear Regression

Learning ≈ Looking for a Function

- Speech Recognition

f  $) =$ "the book"

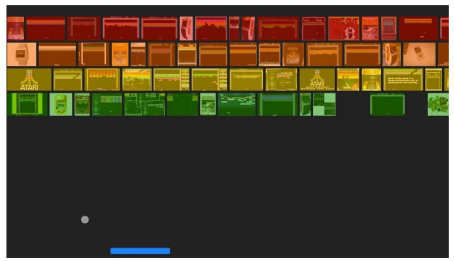
- Handwriting Recognition

f  $) =$ "2"

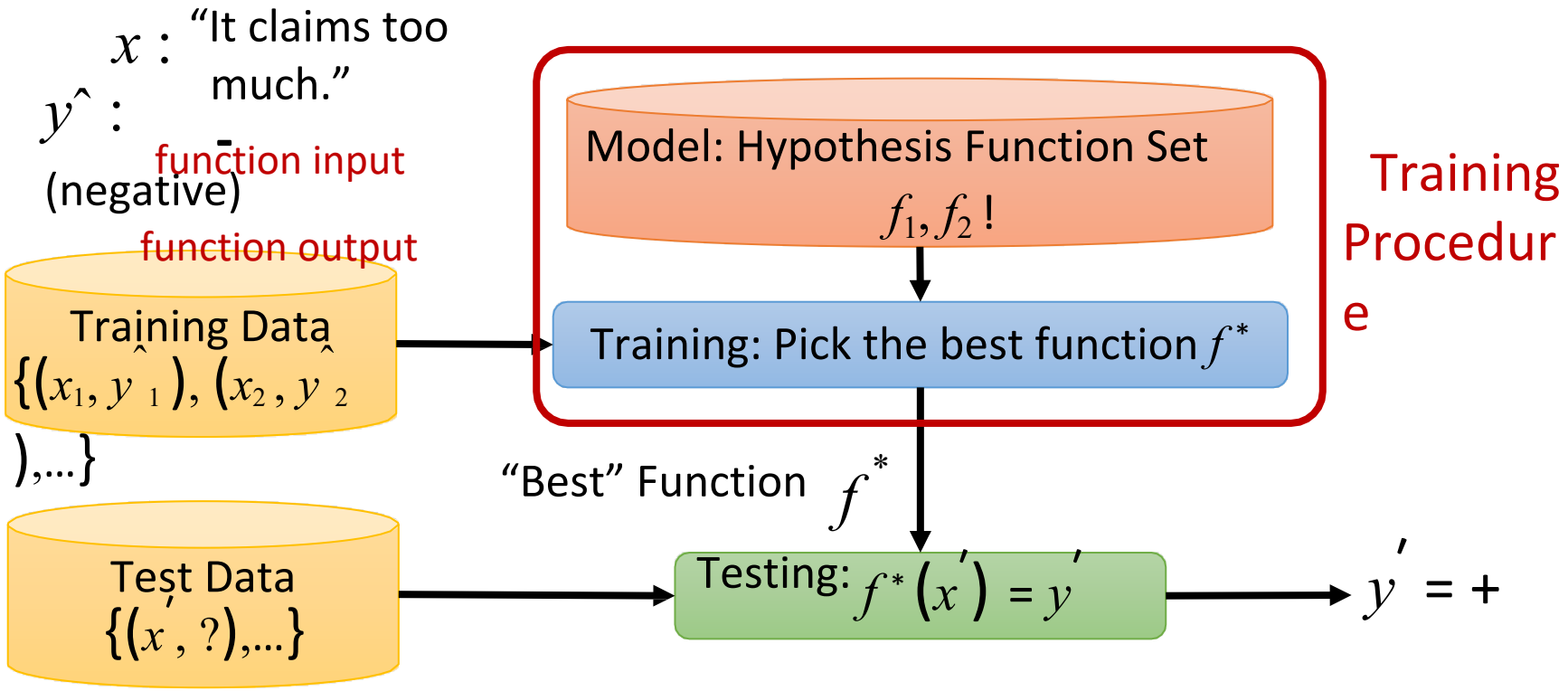
- Weather forecast

f Thursday $) =$  Saturday"

- Play video games

f  $) =$ "move left"

Machine Learning Framework



Training aims to pick the best function given the observed examples Testing predicts the label using the learned function

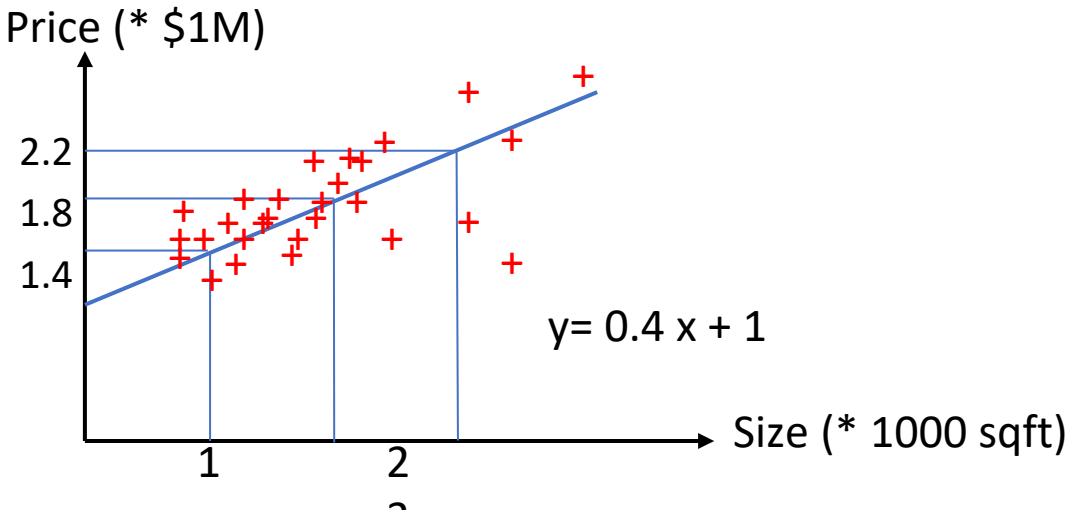
Linear Regression

- **Regression**: modeling the relationship between data points $\mathbf{x}$ and corresponding real-valued targets y
  - Example: Predicting prices of homes
  - Label: Price, $y^{(i)}$
  - Features: Size, Age

- $x^{(i)}=[x^{(i)}_1, x^{(i)}_2]$

2019 Sales:

Size (sqft)	Price
2100	$1.90M
1600	$1.56M
3200	$2.40M
…	…



Linear Regression (cont.)

- Assumption:
  - the relationship between the *features* $\mathbf{x}$ and targets y is linear,
  - i.e., y can be expressed as a weighted sum of the inputs $\mathbf{x}$, give or take some noise on the observations

$$\text{price} = w_1 \cdot x_1 + w_2 \cdot x_1 + b$$

weights

Bias (offset or intercept)

Linear Regression: Modeling bivariate data

- Bivariate data: (x_1,y_1), (x_2, y_2), …, (x_n,y_n)
- Model: $y_i = f(x_i) + e_i$ Random error

- Supervised approach!
- Model allows us to predict the value of y for any given value of x.
  - x is called the independent or predictor variable.
  - y is the dependent or response variable.

Examples of $f(x)$

- lines: $f(x) = mx + b$
- polynomials: $f(x) = ax^2 + bx + c$
- others: $f(x) = a/x + b$

  $f(x) = a\ sin(x) + b$

  $f(x) = a\ sqrt(x)$

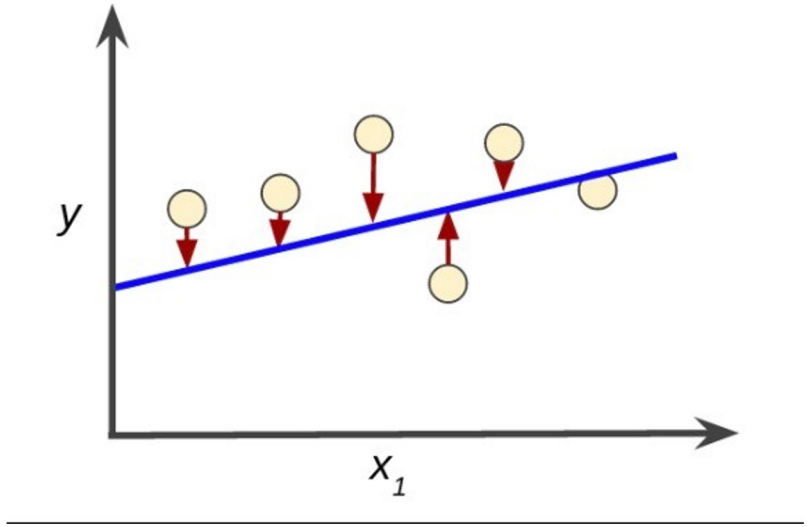
Assumptions for Linear Regression

- The relationship between *x* and *y* is linear
- *y* is distributed normally at each value of *x*, and the variance of *y* at every value of *x* is the same
- The observations are independent

Mean Squared Error

- y_i is the actual value, $f(x_i)$ is the prediction, and their difference is the error.
- Commonly used function, (mean) squared error:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

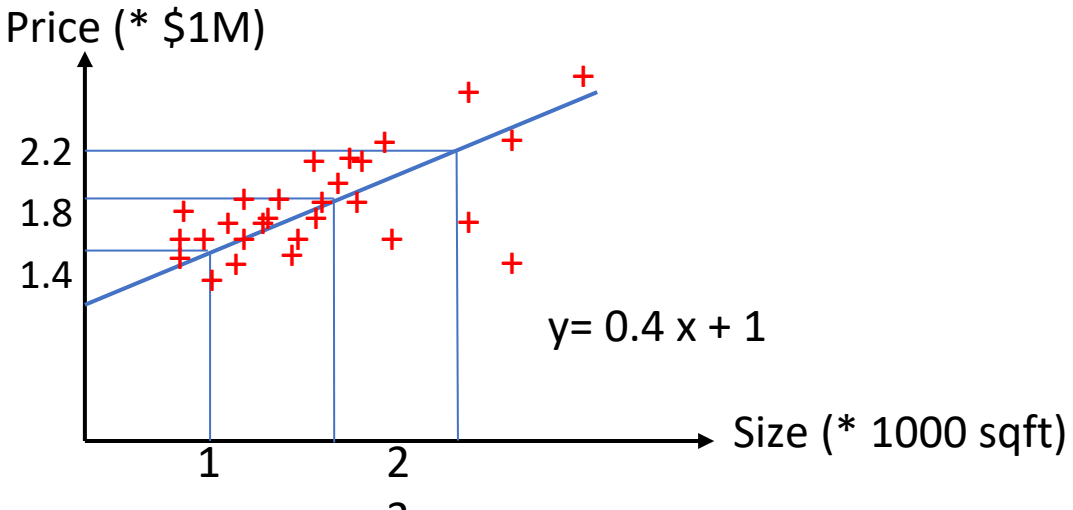


Linear Regression (cont.)

- **Regression**: modeling the relationship between data points
- $\mathbf{x}$ and corresponding real-valued targets y
  - Example: Predicting prices of homes
  - Label: Price, $y^{(i)}$
  - Features: Size, Age
    - $x^{(i)}=[x^{(i)}_1, x^{(i)}_2]$

2019 Sales:

Size (sqft)	Price
2100	$1.90M
1600	$1.56M
3200	$2.40M
…	…



Linear Regression (cont.)

- Assumption:
  - the relationship between the *features* $\mathbf{x}$ and targets y is linear,
  - i.e., y can be expressed as a weighted sum of the inputs $\mathbf{x}$, give or take some noise on the observations

$$\text{price} = w_1 . x_1 + w_2 . x_1 + b$$

weights

Bias (offset or intercept)

Linear Regression (cont.)

- Goal: choose the weights w and bias b to best fit the true values of y observed in the data
- When our inputs consist of d features, we express our prediction $\hat{y}$ as:

$$\hat{y} = w_1 \cdot x_1 + \ldots + w_d \cdot x_d + b$$

- Collecting all features into a vector $\mathbf{x}$ and all weights into a vector $\mathbf{w}$, our model can be expressed compactly using a dot product:

$$\hat{y} = \mathbf{w}_T \mathbf{x} + b$$

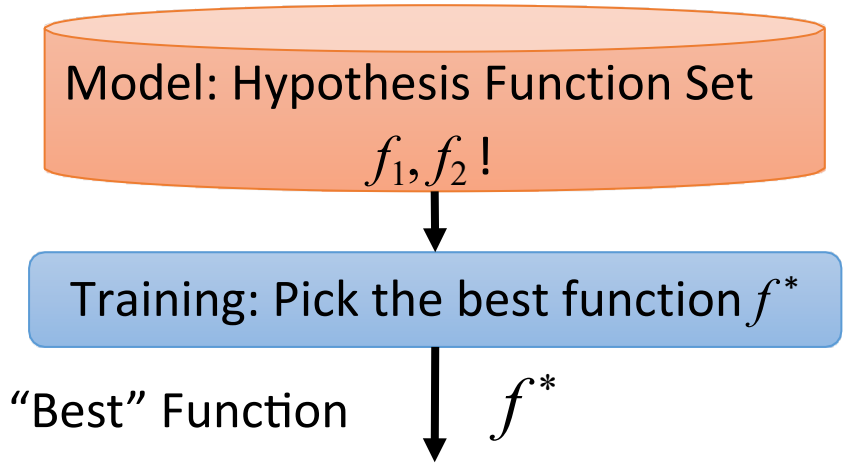
- $\mathbf{x}$: feature vector for a single data point
- $\mathbf{X}$: a collection of data points
- $\mathbf{y}$: vector of estimated predictions

$$\mathbf{y} = \mathbf{X}\mathbf{w} + b$$

Training Procedure:
Searching for the best parameters

$$\mathbf{y} = \mathbf{X}\mathbf{w} + b$$

Parameters: elements of $\mathbf{w}$ and b



- Q1. What is the model? (function hypothesis set)
- Q2. What does a "good" function mean?
- Q3. How do we find the "best" function?

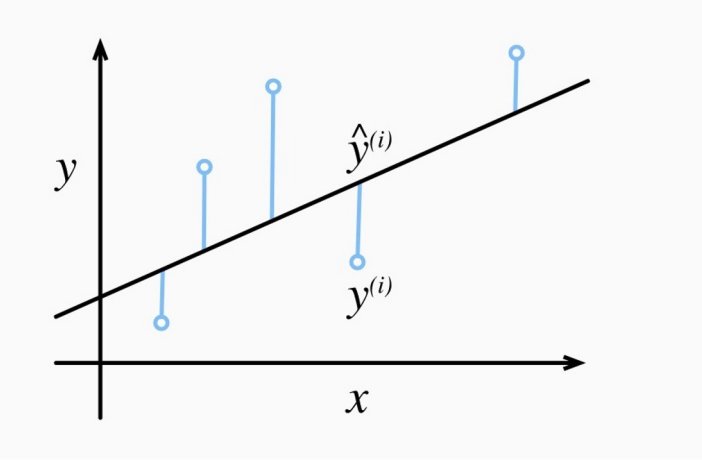
Loss Function

- Measure of *fitness*, quantifies the distance between $\mathbf{y}$ and $\mathbf{y}$.
- Sum of squared errors:

$$l^{(i)}(\mathbf{w}, b) = \frac{1}{2}(\hat{y}^{(i)} - y^{(i)})^2$$

- To measure quality over the entire training set,

  we average over all K training examples



Residual = truth - predicted

Training Procedure (cont.)

- Training the model ≈ search for parameters $(\mathbf{w}^*, b^*)$ that minimize the total loss across all training samples:

$$\mathbf{w}^*, b^* = \text{argmin}_{\mathbf{w},b} L(\mathbf{w}, b)$$