

Lecture 8: A brief on supervised methods

Dr. Greg Chism

Outline

- Supervised Learning
 - **Logistic Regression**
 - SVMs
- Decision Trees
- Random Forests

Logistic Regression

- We made a distinction earlier between regression (predicting a real value) and classification (predicting a discrete value).
- Logistic regression is designed as a **binary classifier** (output say {0,1}) but actually **outputs the probability** that the input instance is in the “1” class.
- A logistic classifier has the form:

$$p(X) = \frac{1}{1 + \exp(-X\beta)}$$

where $X = (X_1, \dots, X_n)$ is a vector of features.

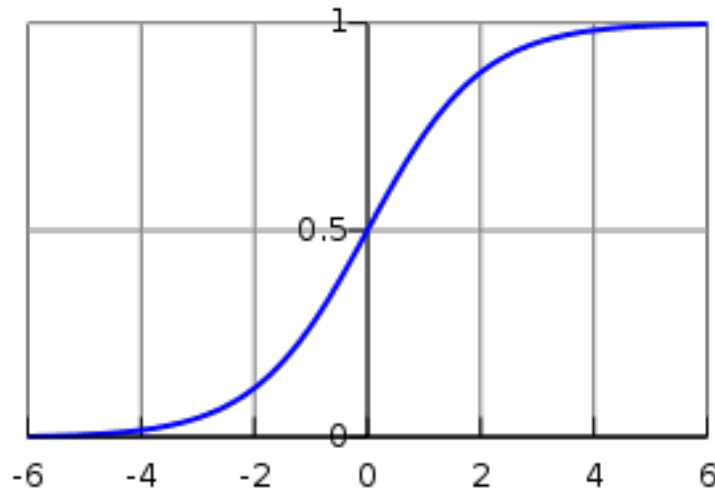
Logistic Regression

- Logistic regression is probably the **most widely used general-purpose classifier**.
- Its **very scalable** and can be **very fast** to train. It's used for
 - Spam filtering
 - News message classification
 - Web site classification
 - Product classification
 - Most classification problems with large, sparse feature sets.
- The only caveat is that **it can overfit** on very sparse data, so its often used with Regularization

Logistic Regression

- Logistic regression maps the “regression” value $-X\beta$ in $(-\infty, \infty)$ to the range $[0,1]$ using a “logistic” function:

$$p(X) = \frac{1}{1 + \exp(-X\beta)}$$



- i.e. the logistic function maps any value on the real line to a probability in the range $[0,1]$

Logistic Training

For training, we start with a collection of **input values** X^i and corresponding **output labels** $y^i \in \{0,1\}$. Let p^i be the **predicted output** on input X^i , so

$$p^i = \frac{1}{1 + \exp(-X^i\beta)}$$

The **accuracy** on an input X^i is

$$A^i = y^i p^i + (1 - y^i)(1 - p^i)$$

Logistic regression maximizes either the sum of the log accuracy, or the total accuracy, e.g.

$$A = \sum_{i=1}^N \log A^i$$

Outline

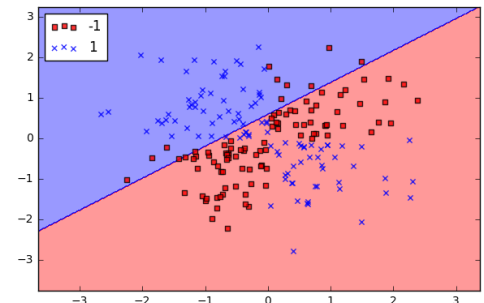
- Supervised Learning
 - Logistic Regression
 - **SVMs**
- Decision Trees
- Random Forests

Support Vector Machine Basics

- Supervised learning technique used for classification and regression analysis
 - Input of train data (with known response variable)
 - Creates a decision boundary between data points
 - Separates data into distinct sets

Support Vector Machine Basics

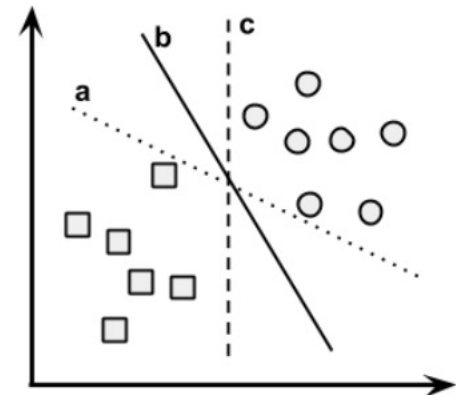
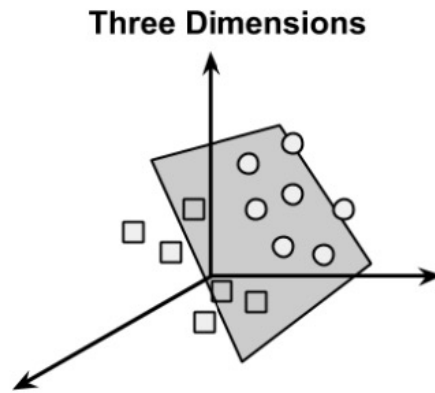
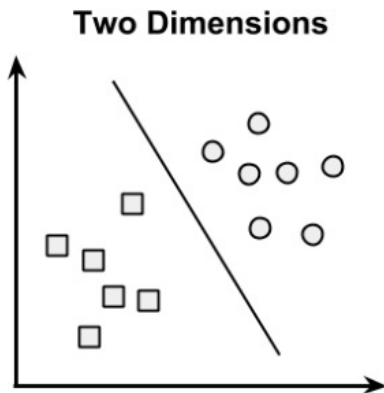
- Generally understood for binary classification.
- However, SVMs are extremely flexible
 - Classification (e.g. gene expression, bioinformatics, text categorization, etc)
 - Numeric prediction (...)
- We're going to focus on the classification side of SVMs.



Support Vector Machine Basics

- SVMs use a linear boundary called a hyperplane to partition data into groups of similar elements.

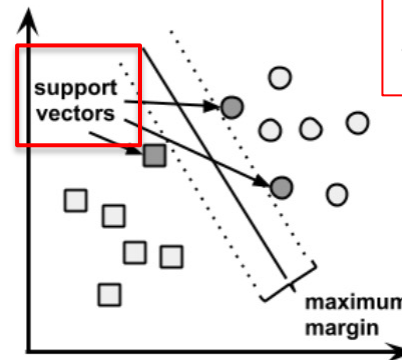
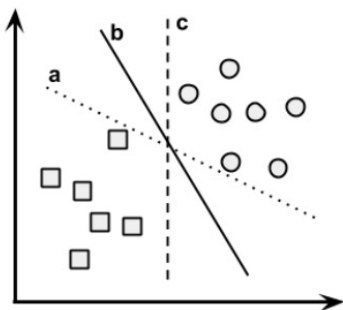
The task (SVM)?
Finding the plane(s)!



Support Vector Machine Basics

- How to choose between different planes?
- Some jargon: This step involves search for the **Maximum Margin Hyperplane (MMH)**.
- The MMH creates the greatest separation between the two classes.

This algorithm depends on vector geometry

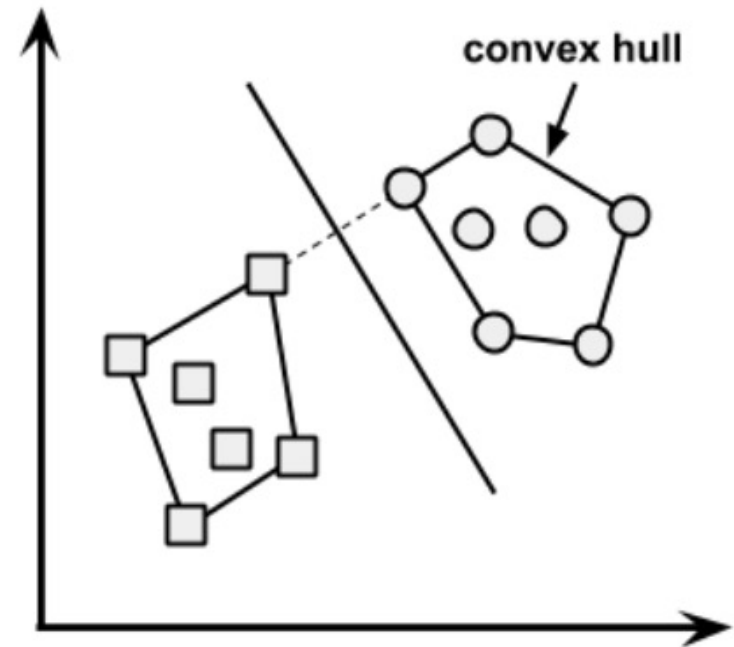


The closest points within each group to the MMH!

SVMs – linearly separable data

Option 1

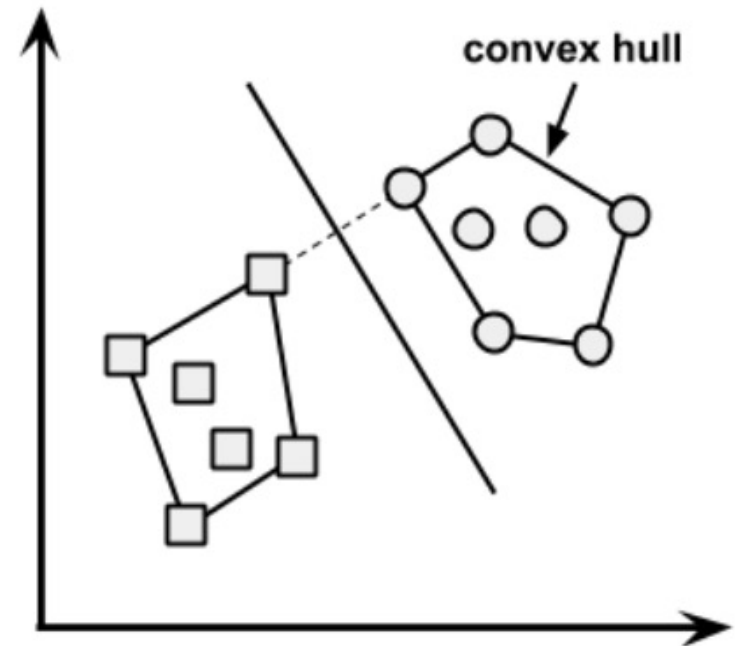
- Based on outer boundaries of each group in the data (aka Convex hulls)
- **MMH** is the perpendicular bisector of the shortest line between the two convex hulls.
- Quadratic optimization!
- This is largely a geometrical solution... MMH is then the perpendicular bisector of the shortest line between the two convex hulls



SVMs – linearly separable data

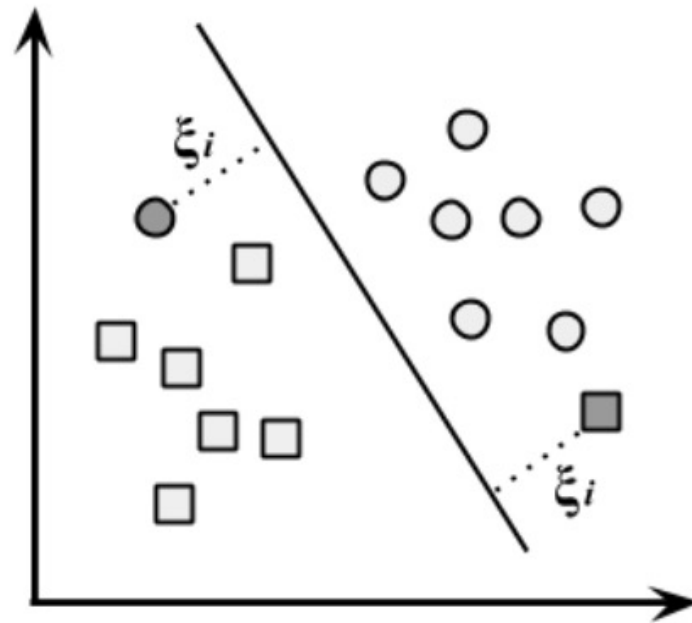
Option 2

- Extensive search across all potential hyperplanes
- Set of two parallel planes which divide the points into homogeneous groups yet themselves are as far apart as possible



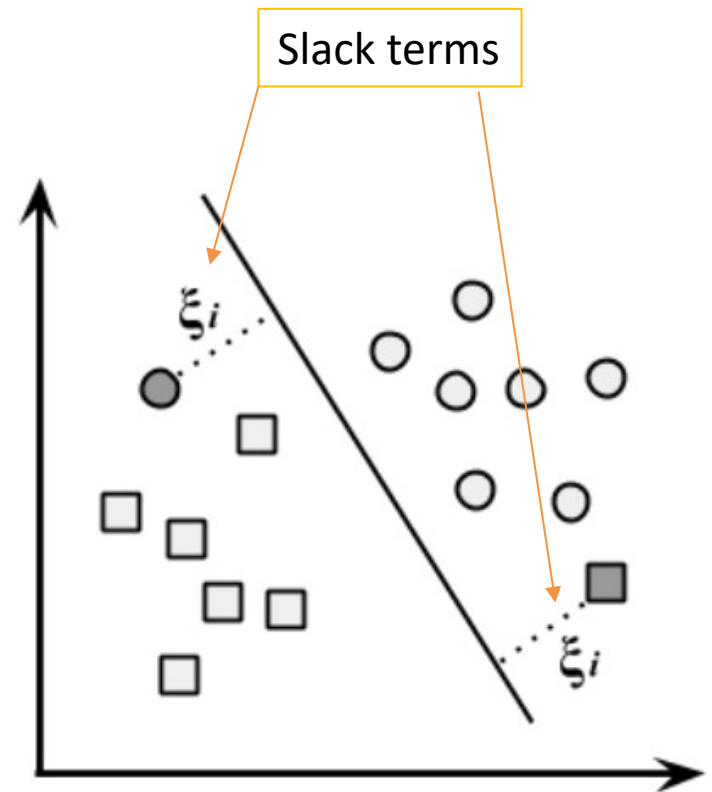
SVMs – non-linearly separable data

For some datasets, linear separation between groups is not possible.



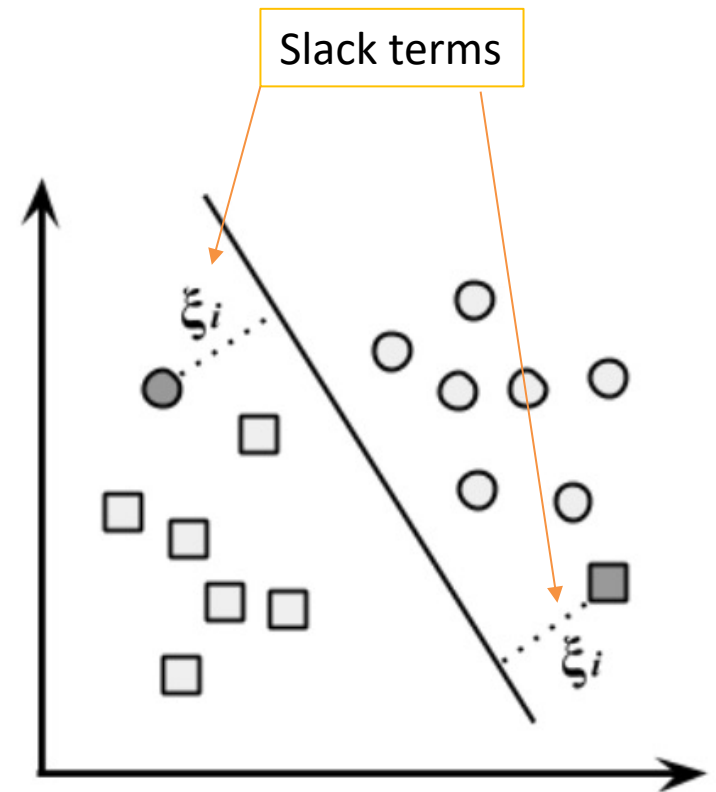
SVMs – non-linearly separable data

- **Slack variables!**
- Creates a soft margin that allows some points to fall on the incorrect side of the margin.
- In the figure: two points falling on the wrong side of the line
- Each of these points has a corresponding slack terms.



SVMs – non-linearly separable data

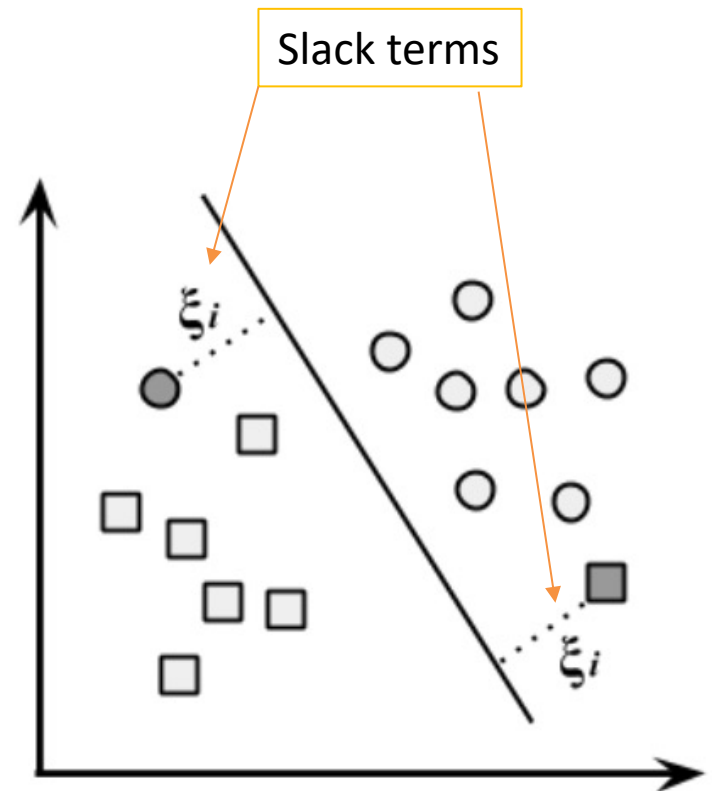
- A cost value (denoted as C) is applied to all points that violate the constraints.
- Therefore, the algorithm, rather than finding the maximum margin, will attempt to minimize the total cost.



SVMs – non-linearly separable data

The C parameter

- Modifying C will adjust the penalty for observations that fall on the wrong side of the hyperplane.
- Large C: The greater the cost parameter, the harder the optimization will try to achieve 100 percent separation.
- Small C: A lower cost parameter will place the emphasis on a wider overall margin.
- The goal: a balance between over- and under-fit!

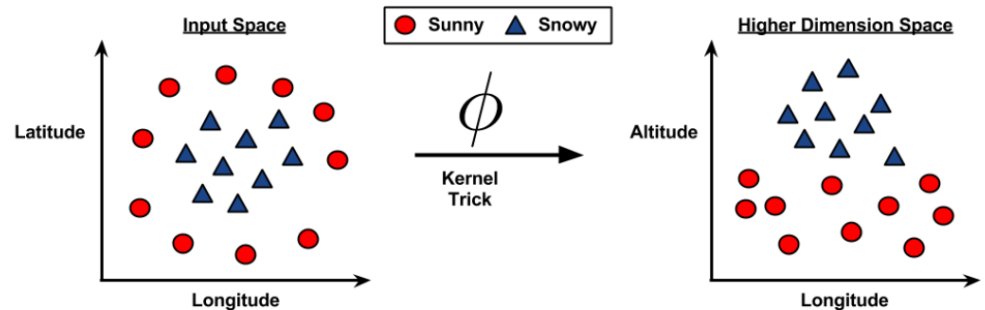


Using kernels for non-linear spaces

- Another approach to account for non-linearity using SVMs!
- The kernel trick: map the problem into a higher dimension space.
- A graphical explanation!

The kernel trick

- SVMs with non-linear kernels add additional dimensions to the data in order to create separation in this way.
- The kernel trick involves a process of adding new features that express mathematical relationships between measured characteristics



For this example, altitude feature can be expressed mathematically as an interaction between latitude and longitude

Using kernels for non-linear spaces

- SVM to learn concepts that were not explicitly measured in the original data.
- New perspectives on the data...

Strengths	Weaknesses
<ul style="list-style-type: none">• Can be used for classification or numeric prediction problems• Not overly influenced by noisy data and not very prone to overfitting• May be easier to use than neural networks, particularly due to the existence of several well-supported SVM algorithms• Gaining popularity due to its high accuracy and high-profile wins in data mining competitions	<ul style="list-style-type: none">• Finding the best model requires testing of various combinations of kernels and model parameters• Can be slow to train, particularly if the input dataset has a large number of features or examples• Results in a complex black box model that is difficult if not impossible to interpret

Outline

- Supervised Learning
 - Logistic Regression
 - SVMs
- **Decision Trees**
- Random Forests

Decision trees

- Decision tree learners build a model in the form of a tree structure.
 - Decision nodes: Decision to make on an attribute.
 - Branches: Decision's choice.
 - Leaf nodes: The outcome of a decision combination.
 - Root node: The data that is being classified.

Decision trees – when to use them?

- Essentially a flowchart
- Lots of applications and their performance is generally unparalleled.
- Appropriate for applications in which the classification mechanism needs to be explicit.
 - Credit scoring models in which the criteria that causes an applicant to be rejected need to be well-specified
 - Marketing studies of customer churn or customer satisfaction that will be shared with management or advertising agencies
 - Diagnosis of medical conditions based on laboratory measurements, symptoms, or rate of disease progression

Decision trees – when NOT to use them?

- Will the data set result in a very large number of decisions?
 - A large number of nominal features with many levels...
 - A large number of numeric features...

Decision trees – divide and conquer

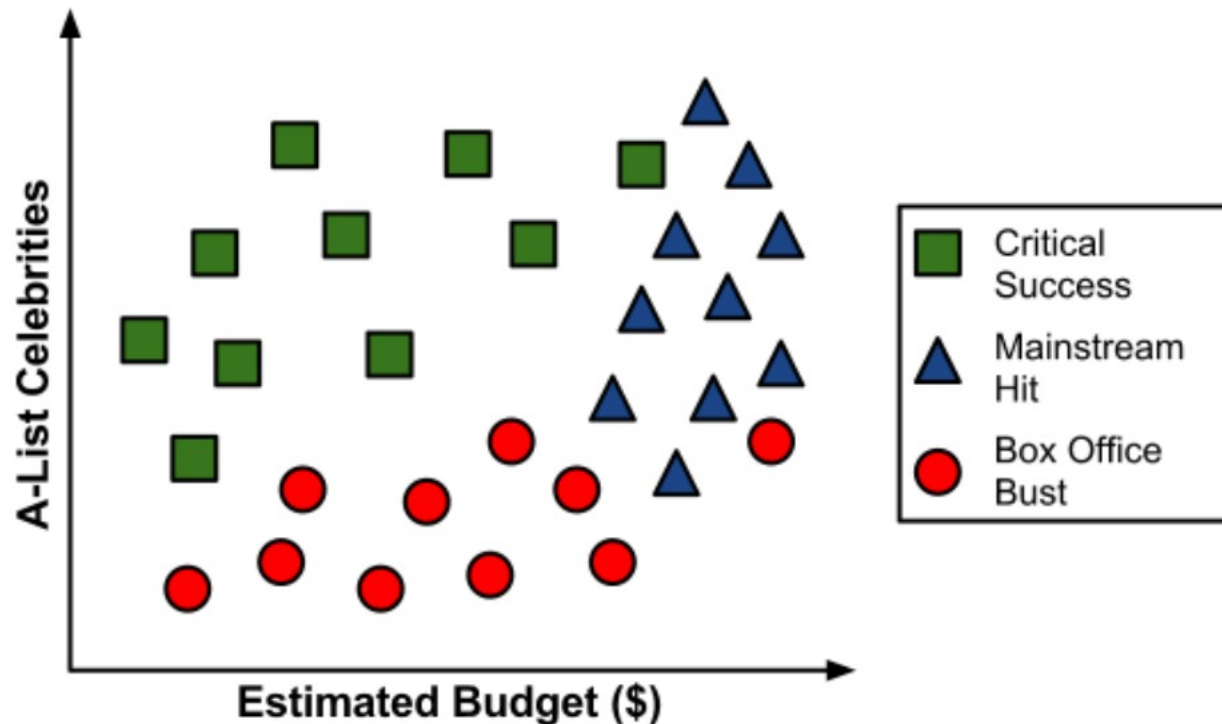
- Decision trees are built using recursive partitioning (heuristic).
 - Splits de data into smaller and smaller subsets of similar classes.
- General steps:
 - First, the algorithm chooses a feature that is the most predictive of the target class.
 - Second, the algorithm continues to divide-and-conquer the nodes, choosing the best candidate feature each time
 - Finally, – stopping criterion.

Decision trees – divide and conquer

- Situation: your desk is piled with 1000 screenplays.
- You decide to develop a decision tree algorithm to predict whether a potential movie would fall into
 - mainstream hit
 - critic's choice,
 - or box office bust.

Decision trees – divide and conquer

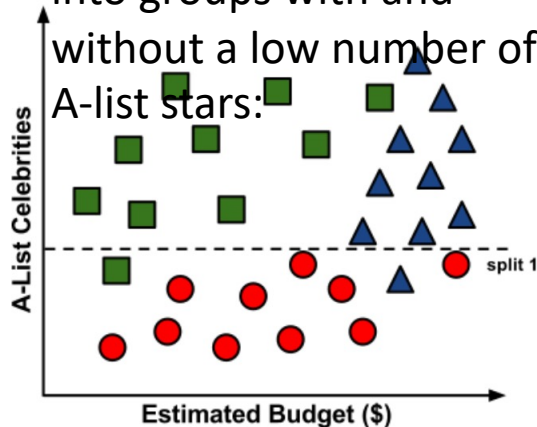
- After reviewing data for 30 movies...



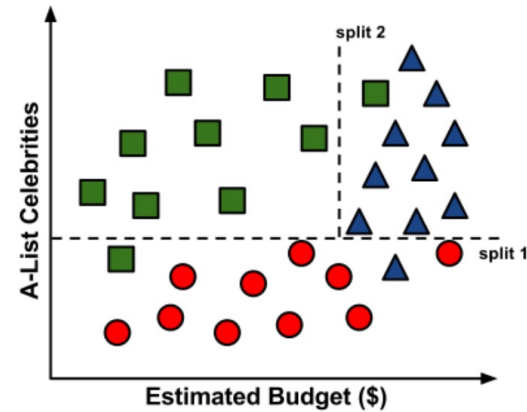
Decision trees – divide and conquer

- After reviewing data for 30 movies...

Let's first split the feature indicating the number of celebrities, partitioning the movies into groups with and without a low number of A-list stars:



among the group of movies with a larger number of celebrities, we can make another split between movies with and without a high budget

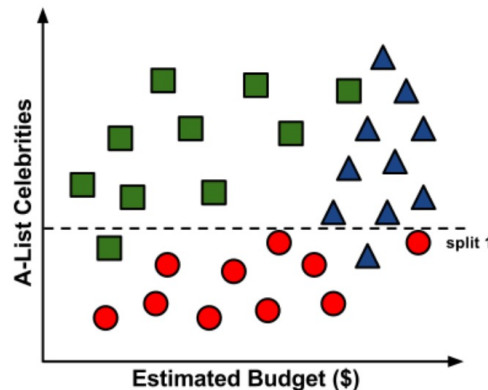


We stop the algorithm when more than 80 percent of the data points in each group are from a single class

Decision trees – divide and conquer

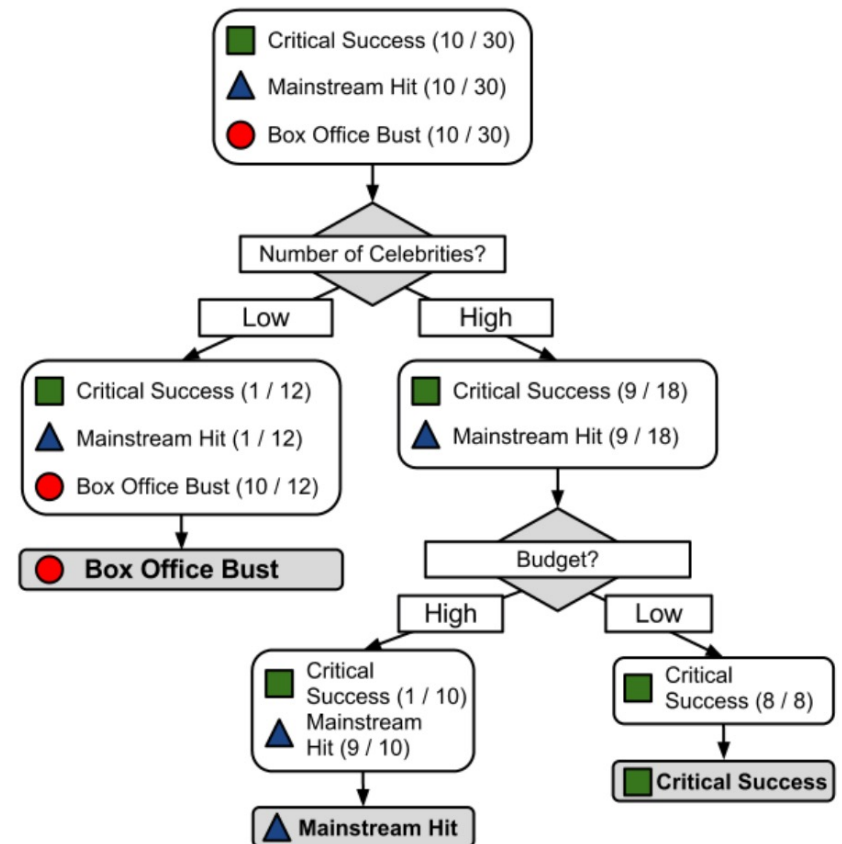
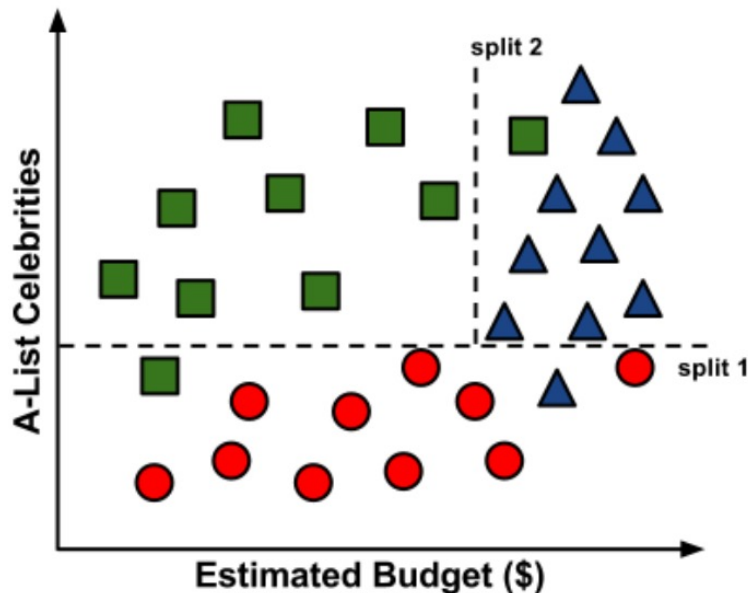
Why didn't we use a diagonal first?

- A limitation of decision trees:
 - **axis-parallel splits.**
 - Why? It prevents complex decisions to be made. Only one axis changes at a time.



Decision trees – divide and conquer

Let's take a look at our model!



Decision trees – entropy

Central to this part is the concept of “Purity”

If the segments of data contain only a single class, they are considered **pure**

Entropy of a data set (or sample): indicates how mixed the class values are [2 classes: 0–1*]

0 bits: sample is completely homogenous

1 bits: maximum amount of disorder

*Entropy range = $0 - \log_2(n)$

Decision trees – entropy

What is the main goal of using entropy:

- Again, entropy is a measure of randomness
- Sets with high entropy are very diverse and provide little information about other items that might belong to the set (i.e. not a lot in common).
- Sets with low entropy are homogeneous.

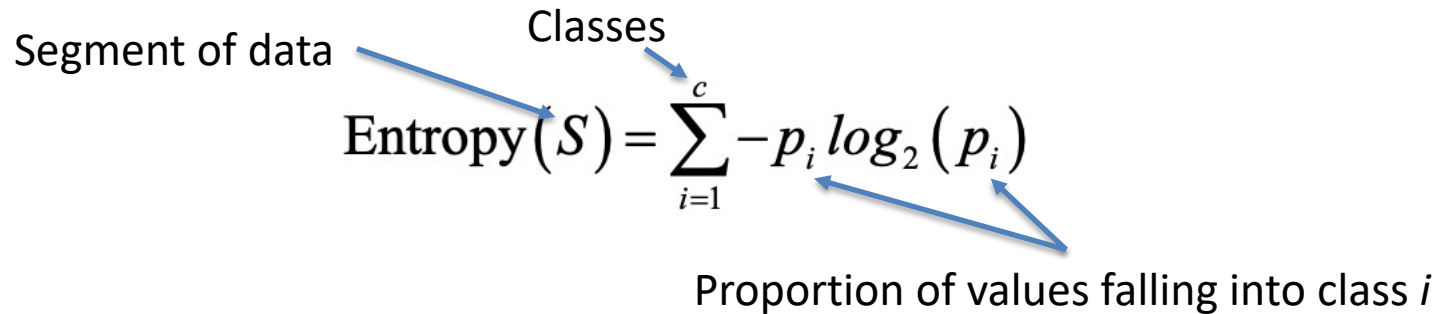
Decision trees – entropy

Segment of data

Classes

$$\text{Entropy}(S) = \sum_{i=1}^c -p_i \log_2(p_i)$$

Proportion of values falling into class i



Estimating entropy of a given split (2 groups)

- A partition of data with two classes: class red (60 percent), and class blue (40 percent):

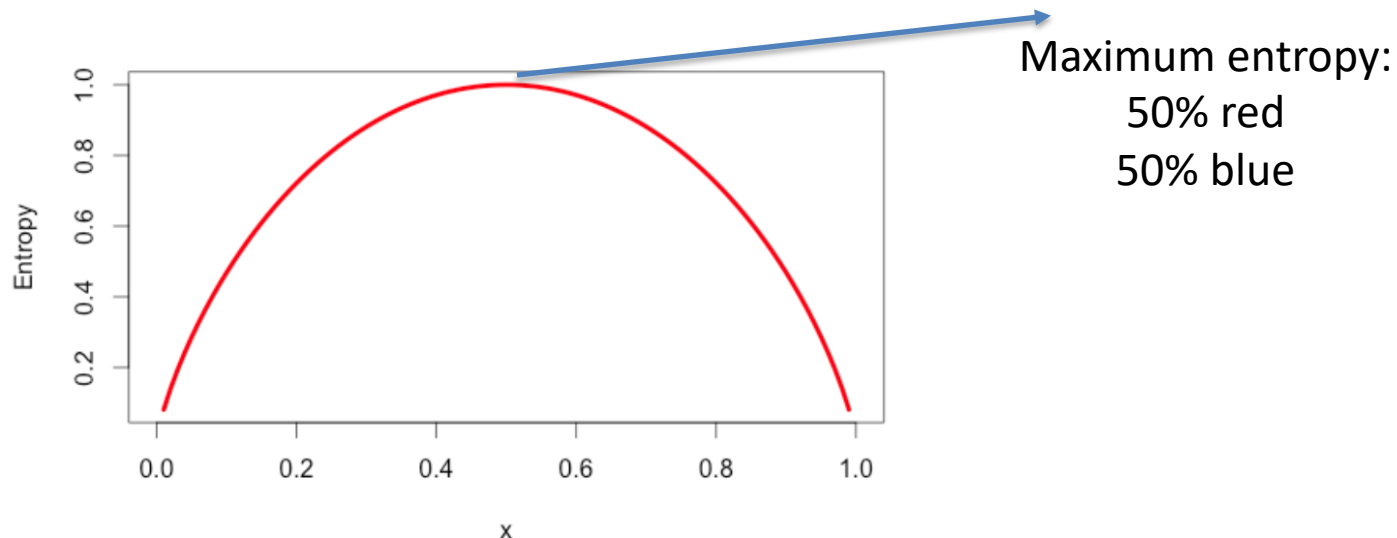
$$\begin{aligned} \text{Entropy} &= -0.60 * \log_2(0.60) - 0.40 * \log_2(0.40) \\ &= 0.9709 \end{aligned}$$

Decision trees – entropy

Estimating entropy of a given split (2 groups)

- A partition of data with two classes: class red (60 percent), and class blue (40 percent):

$$\text{Entropy} = -0.60 * \log_2(0.60) - 0.40 * \log_2(0.40) \\ = 0.9709$$



Decision trees – Information gain

- Information gain: the change in homogeneity (entropy) resulting from a split on each possible feature.
- Calculated as the difference in entropy before the split (S_1) and after the split (S_2).
 - It can also account for the proportion of observations falling into a given partition (weights).

$$\text{InfoGain}(F) = \text{Entropy}(S_1) - \text{Entropy}(S_2)$$

Decision trees – Information gain

The higher the information gain, the better a feature is at creating homogeneous groups after a split on that feature

Outline

- Supervised Learning
 - Logistic Regression
 - SVMs
- Decision Trees
- **Random Forests + boosted decision trees**

Random forest

Ensembles of decision trees

Bagging + random feature selection

Easier to use and less prone to over-fitting

Random Forests

Principles: we want to take a **vote between different learners** so we don't want the models to be too similar. These two criteria ensure **diversity** in the individual trees:

- Draw K bootstrap samples of size N :
 - Each tree is trained on different data.
- Grow a Decision Tree, by selecting a **random set of m out of p features** at each node, and choosing the best feature to split on.
 - Corresponding nodes in different trees (usually) can't use the same feature to split.

Random Forests

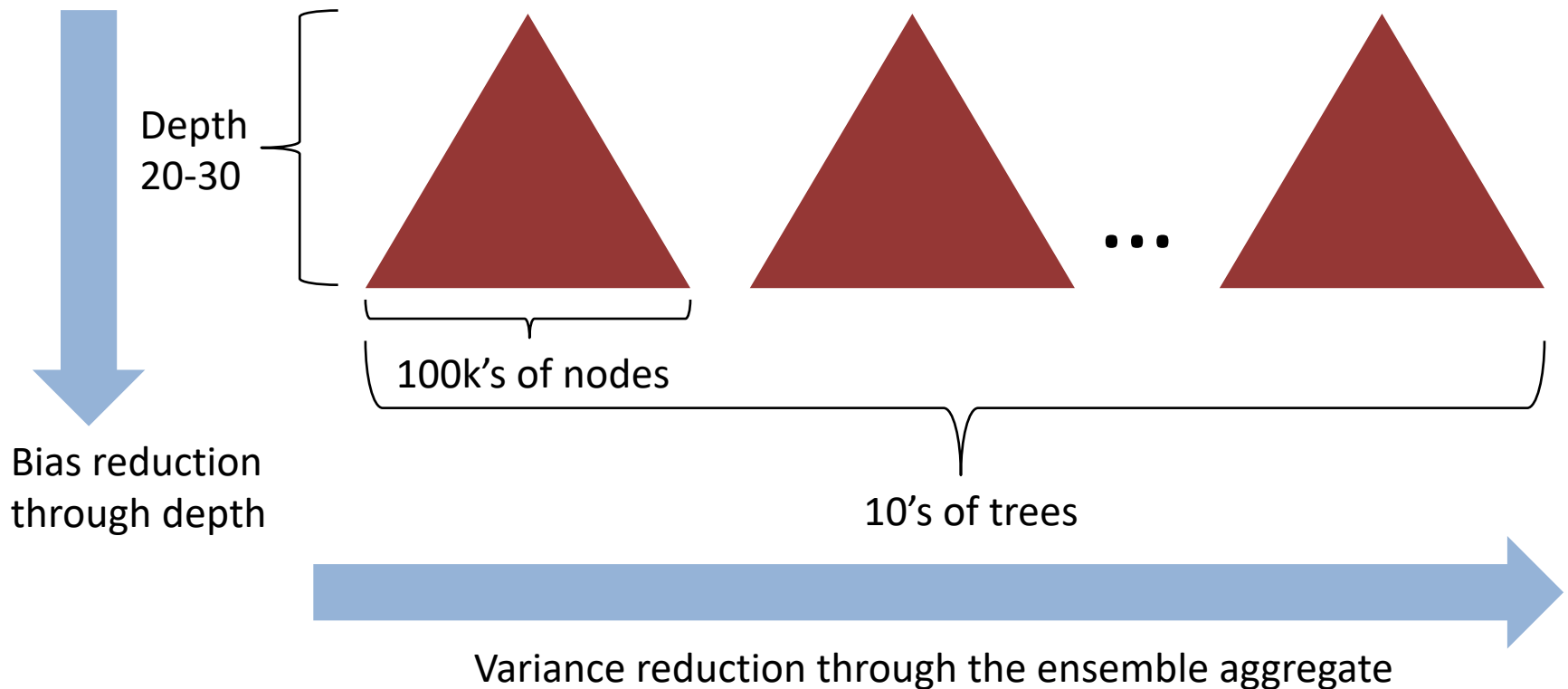
- **Very popular in practice**, probably the most popular classifier for dense data (\leq a few thousand features)
- **Easy to implement** (train a lot of trees). Good match for MapReduce.
- **Parallelizes easily** (but not necessarily efficiently).
- **Not quite state-of-the-art accuracy** – DNNs generally do better, and sometimes gradient boosted trees.
- **Needs many passes over the data** – at least the max depth of the trees. (\ll boosted trees though)
- **Easy to overfit** – hard to balance accuracy/fit tradeoff.

Boosted Decision Trees

- A recently-developed alternative to random Forests:
- In contrast to RFs whose trees are trained **independently**, BDT trees are trained **sequentially** by **boosting**: Each tree is trained on weighted data which emphasizes incorrectly-labeled instances by the previous trees.
- Both methods can produce very high-quality models. Superiority of one method or the other is very dataset-dependent.
- Resource requirements are very different as well, so its actually non-trivial to compare the methods (what resources do you fix during the experiment?).

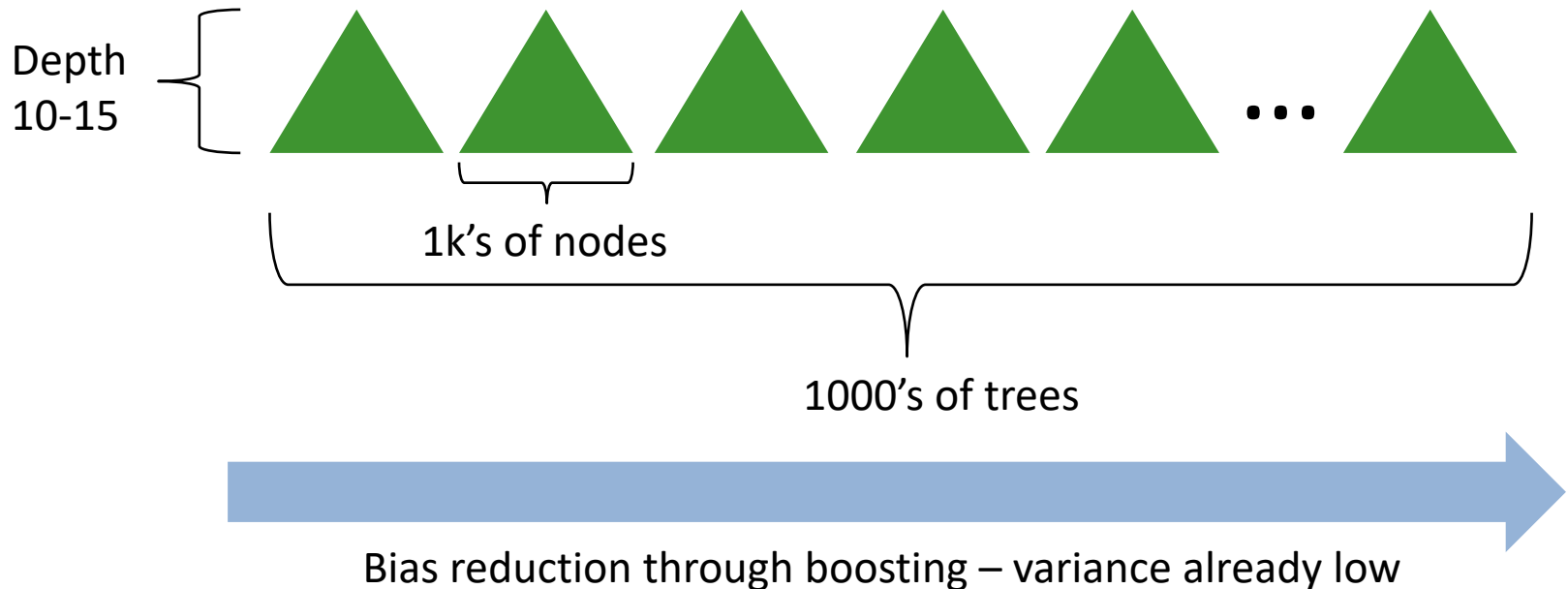
Random Forests vs Boosted Trees

- The “geometry” of the methods is very different (MNIST data):
- Random forest use 10’s of deep, large trees:



Random Forests vs Boosted Trees

- The “geometry” of the methods is very different (MNIST data):
- Boosted decision trees use 1000’s of shallow, small trees:



Random Forests vs Boosted Trees

- RF training embarrassingly parallel, can be very fast
- Evaluation of trees (runtime) also much faster for RFs

