Data Science

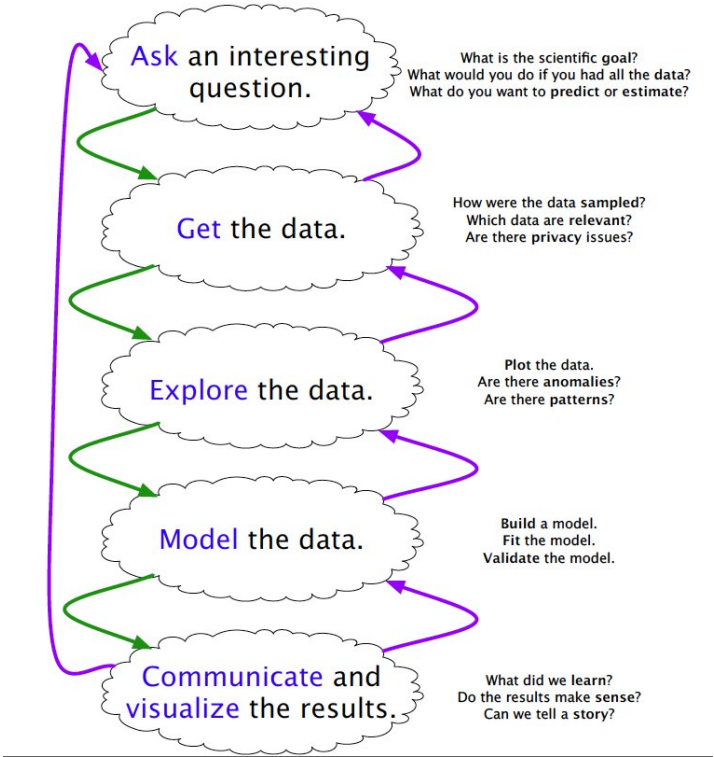
Lecture 3: Machine Learning and Big Data

Model the Data

The Data Science Analysis Pipeline

Modeling is the process of encapsulating information into a tool which can make forecasts/predictions.

The key steps are building, fitting, and validating the model.



Philosophies of Modeling

We need to think in some fundamental ways about modeling to build them in sensible ways.

- Occam’s Razor
- Bias-Variance tradeoffs

Occam’s Razor

This philosophical principle states that “the simplest explanation is best”.

When presented with competing hypothetical answers to a problem, one should select the answer that makes the fewest assumptions.

With respect to modeling, this often means minimizing the parameter count in a model.

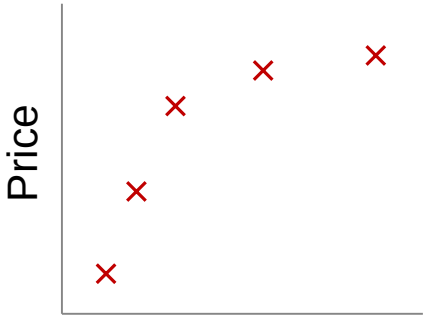
Bias-Variance Tradeoffs

"All models are wrong, but some models are useful."

– George Box (1919-2013)

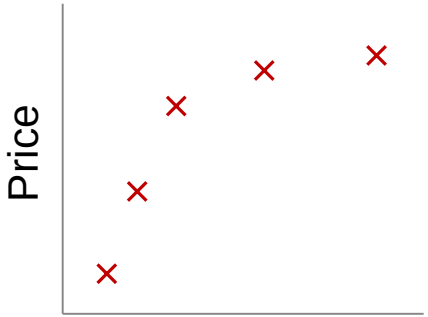
- *Bias* is error from erroneous assumptions in the model, like making it linear. (underfitting)
- *Variance* is error from sensitivity to small fluctuations in the training set. (overfitting)

Bias/variance



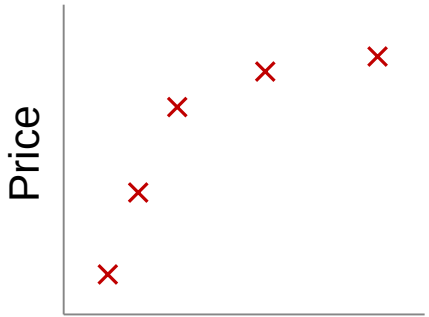
$\theta_0 + \theta_1 x$

High bias
(underfit)



$\theta_0 + \theta_1 x + \theta_2 x^2$

“Just right”



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

High variance
(overfit)

Taxonomy of Models

Models have different properties inherent in how they are constructed:

- Linear vs. nonlinear
- Black box vs. descriptive
- First-principle vs. data-driven
- Stochastic vs. deterministic
- Flat vs. hierarchical

Linear vs. Non-Linear Models

Linear models are governed by equations that weigh each feature variable by a coefficient reflecting its importance, and sum up these values to produce a score.

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Non-linear models involve higher-order polynomials, logarithms, and exponentials.

Blackbox vs. Descriptive Models

Ideally models are descriptive, meaning they explain why they are making their decisions.

Linear regression models are descriptive, because one can see which variables are weighed heaviest.

Neural network models are generally opaque.

Lesson: “Distinguishing cars from trucks.”

Deep Learning Models are Blackbox

Deep learning models for computer vision are highly-effective, but opaque as to how they make decisions.
They can be badly fooled by images which would never confuse human observers.

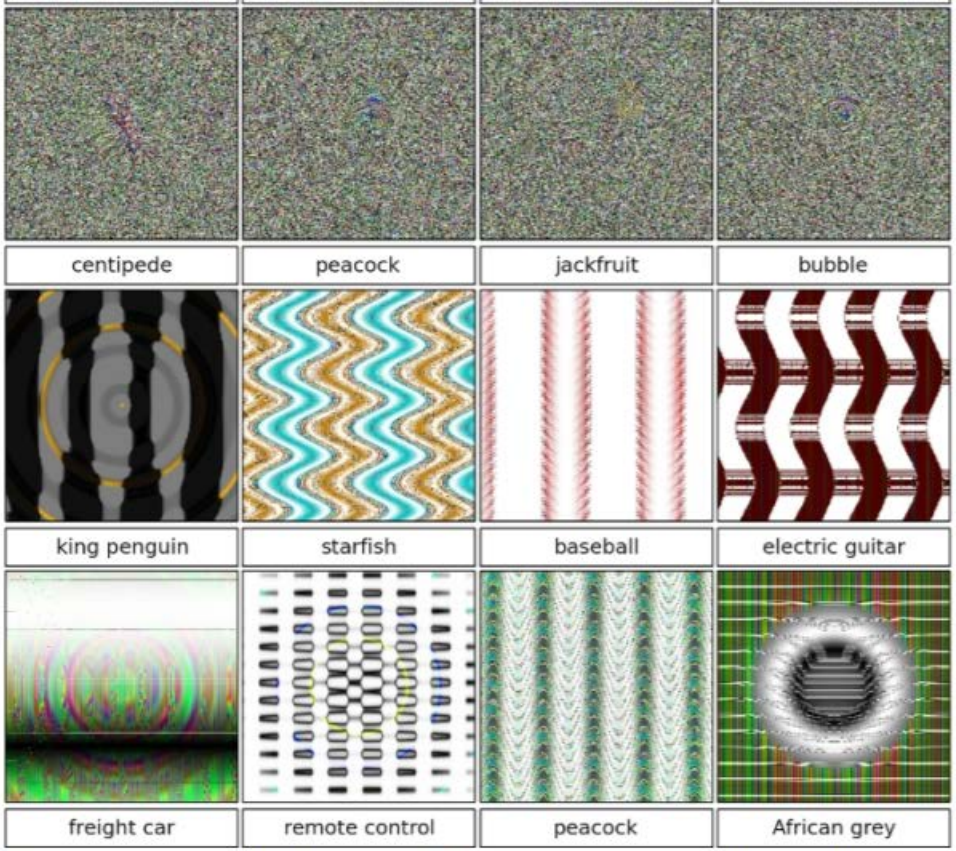


Figure 1. Evolved images that are unrecognizable to humans, but that state-of-the-art DNNs trained on ImageNet believe with $\geq$ 99.6% certainty to be a familiar object. This result highlights differences between how DNNs and humans recognize objects. Images are either directly (*top*) or indirectly (*bottom*) encoded.

General vs. Ad Hoc Models

Machine learning models for classification and regression are general, meaning they employ no problem-specific ideas, only specific data.

Ad hoc models are built using domain-specific knowledge to guide their structure and design.

Evaluating Models: Baselines

The first step to assess whether your model is any good is to build baselines: the simplest *reasonable* models to compare against.

Only after you decisively beat your baselines can your models be deemed effective.

Representative Baseline Models

- Uniform or random selection among labels.
- The most common label in the training data.
- The best performing single-variable model.
- Same label as the previous point in time.
- Mean or median.
- Linear regression.
- Existing models.

Baseline models must be fair: they should be simple but not stupid.

Evaluating Classifiers

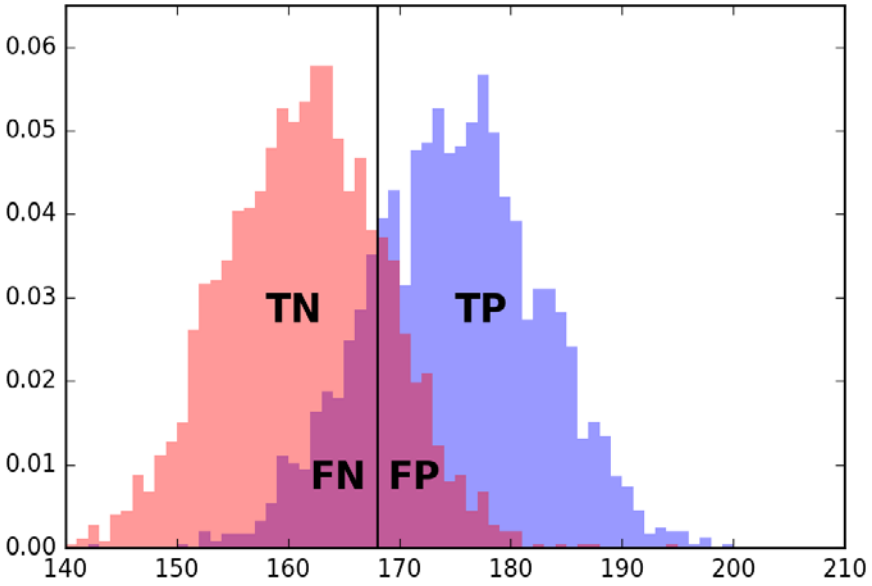
There are four possible outcomes for a binary classifier:

- True positives (TP) where + is labeled +
- True negative (TN) where - is labeled -
- False positives (FP) where - is labeled +
- False negatives (FN) where + is labeled -

		Predicted Class: Yes	Predicted Class: No
Actual Class	Yes	TP	FN
	No	FP	TN

Threshold Classifiers

Identifying the best threshold requires deciding on an appropriate evaluation metric.



Accuracy

The accuracy is the ratio of correct predictions over total predictions:

$$accuracy = \frac{TP + TN}{TP + TN + FN + FP}$$

The monkey would randomly guess P/N, with accuracy 50%.

Picking the biggest class yields >=50%.

Precision

When |*P*|<<|*N*|, accuracy is a silly measure.

e.g. only 5% of tests say cancer.

$$precision = \frac{TP}{(TP + FP)}$$

		Predicted Class: Yes	Predicted Class: No
Actual Class	Yes	TP	FN
	No	FP	TN

Recall

In the cancer case, we would tolerate some false positive (scares) to identify real cases:

$$recall = \frac{TP}{(TP + FN)}$$

		Predicted Class: Yes	Predicted Class: No
Actual Class	Yes	TP	FN
	No	FP	TN

F-Score

To get a meaningful single score balancing precision and recall use F-score:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

The harmonic mean is always <= arithmetic mean, making it tough to get a high F-score.

Sensitivity vs. Specificity

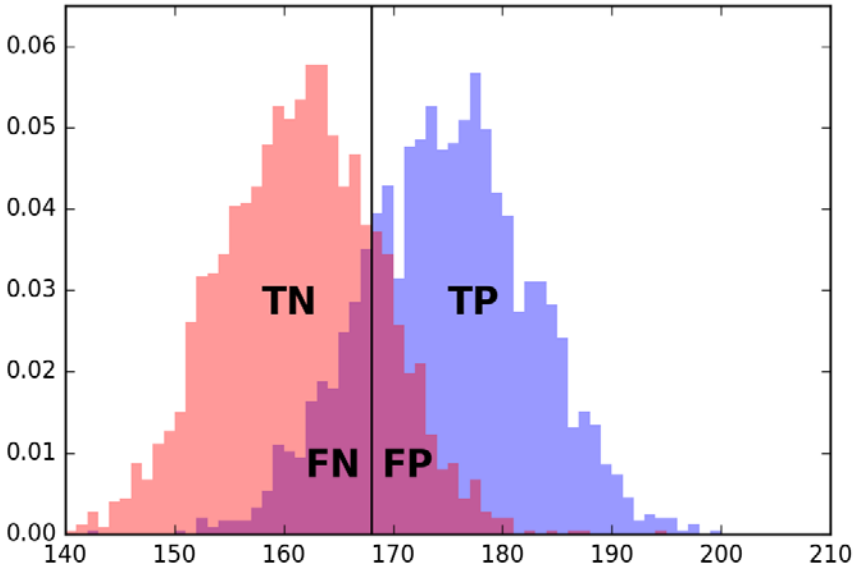
		Predicted Class: Yes	Predicted Class: No
Actual Class	Yes	TP	FN
	No	FP	TN

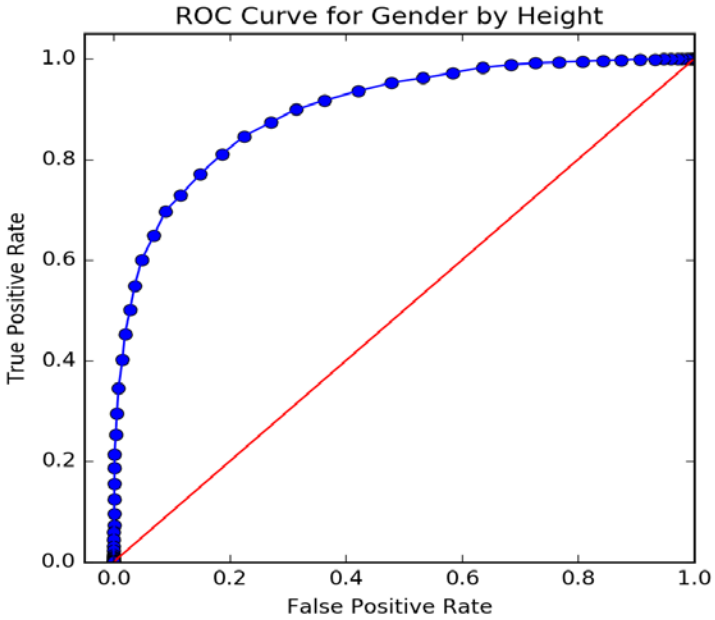
		Predicted Class: Yes	Predicted Class: No
Actual Class	Yes	TP	FN
	No	FP	TN

Receiver-Operator Characteristic (ROC) Curves

Varying the threshold changes recall/precision, TPR/FPR.

Area under ROC curve (AUC) is a measure of accuracy.





Summary Statistics: Numerical Error

For numerical values, error is a function of the delta between forecast *f* and observation *o*:

- Absolute error: *(f - o)*
- Relative error: *(f - o) / o* (typically better)

These can be aggregated over many tests:

- Mean or median squared error

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

- Root mean squared error

$$\mathrm{RMSD}(\hat{\theta}) = \sqrt{\mathrm{MSE}(\hat{\theta})} = \sqrt{\mathrm{E}((\hat{\theta} - \theta)^2)}.$$

Evaluation Data

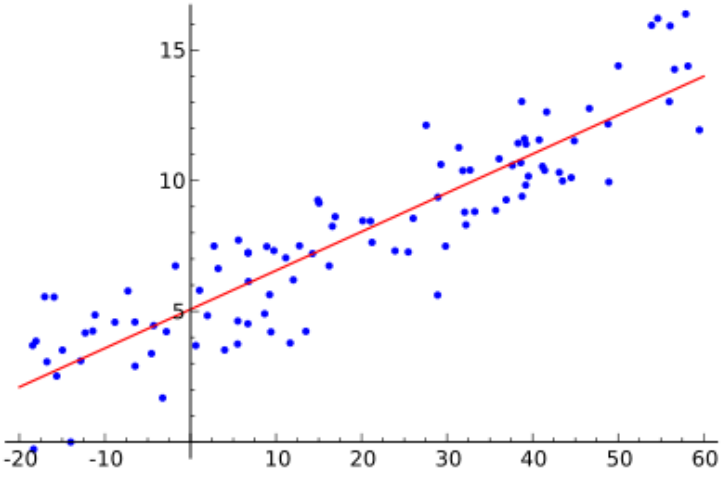
The best way to assess models involve out-of-sample predictions, results on data you never saw (or even better did not exist) when you built the model.

Partitioning the input into training (60%), validation (20%) and testing (20%) data works only if you never open testing data until the end.

Linear Regression

Linear Regression

Given a collection of *n* points, find the line which best approximates or fits the points.



One features(variable).

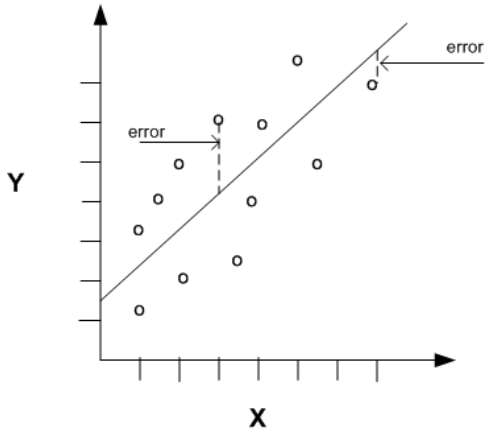
Size (feet²)	Price ($1000)
x	y
2104	460
1416	232
1534	315
852	178
…	…

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Error in Linear Regression

The residual error is the difference between the predicted and actual values:

Least squares regression minimizes the sum of the squares of the residuals of all points.
This metric is chosen because (1) it has a nice closed form and (2) it ignores the sign of the errors.



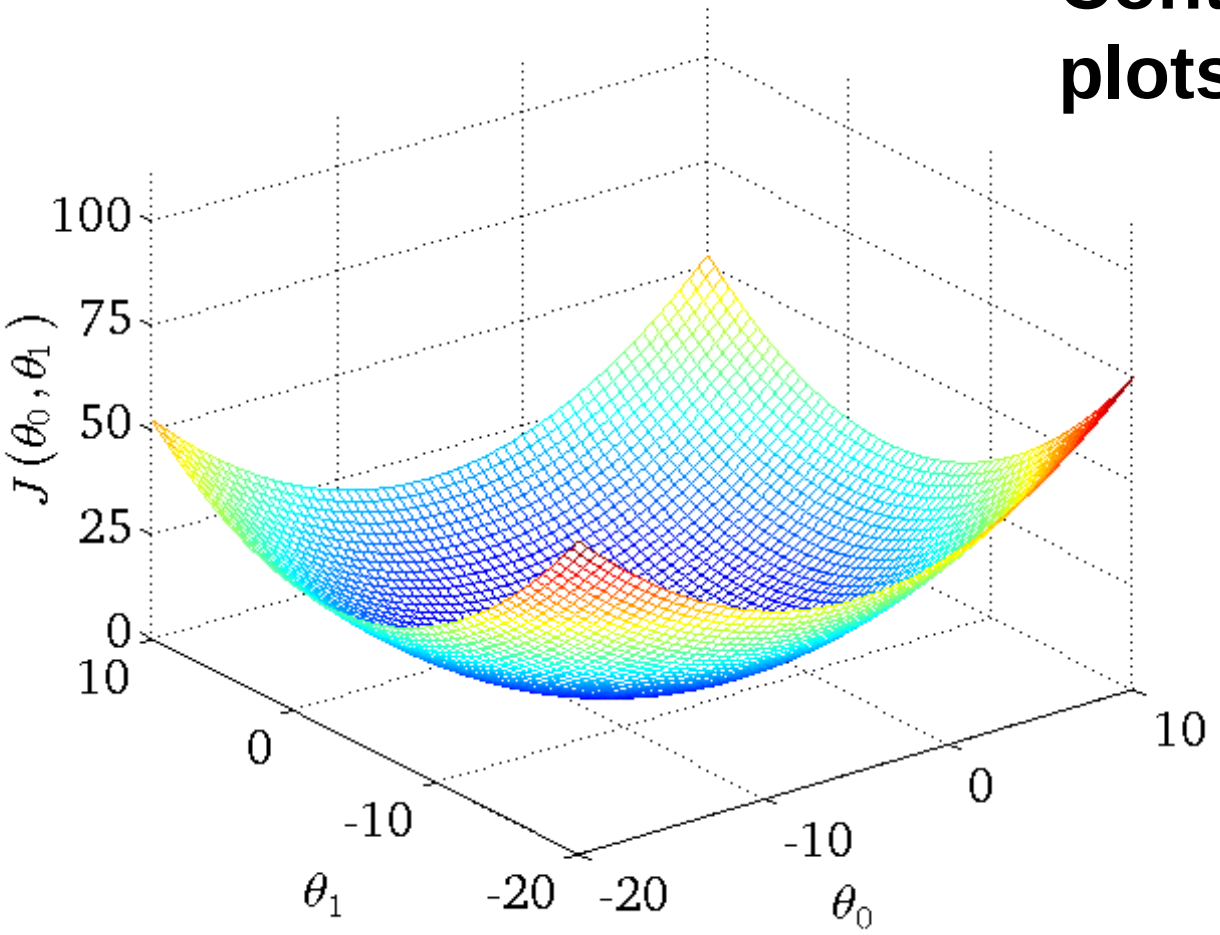
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

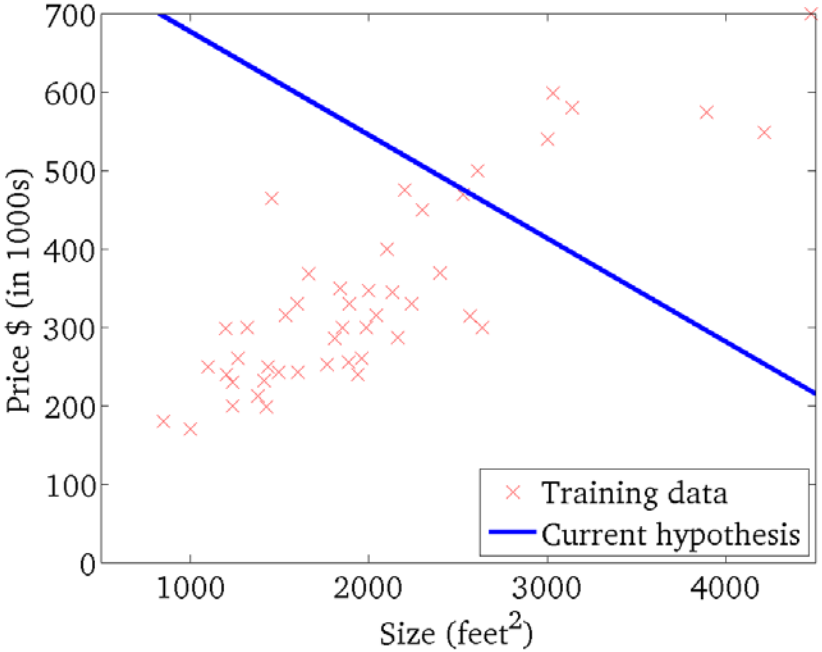
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Contour plots



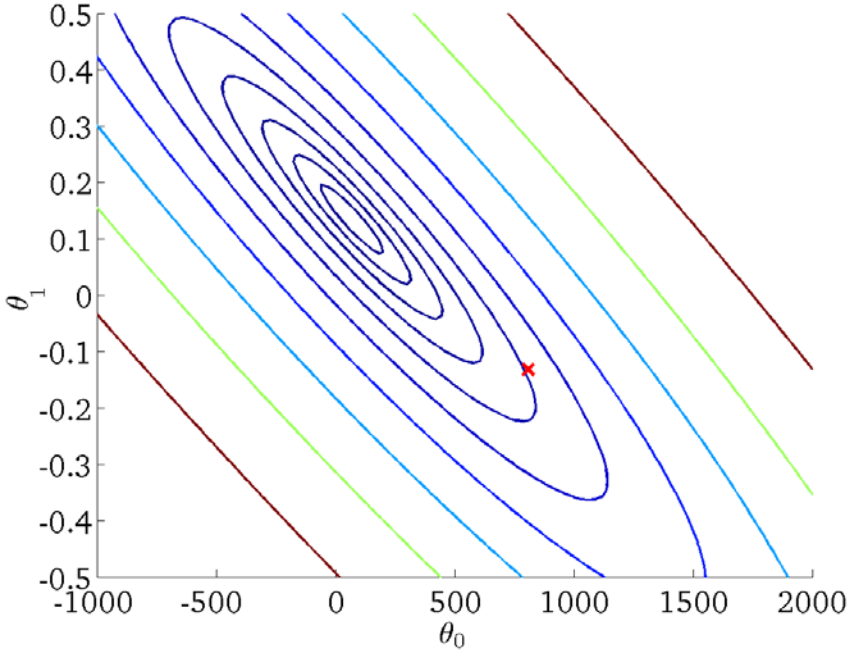
$h_\theta(x)$

(for fixed θ_0, θ_1 this is a function of x)



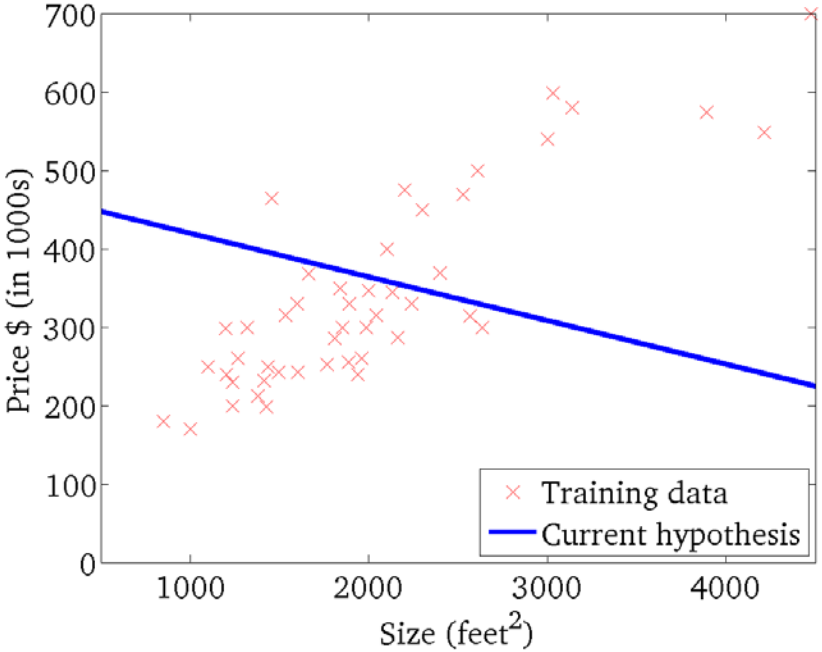
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



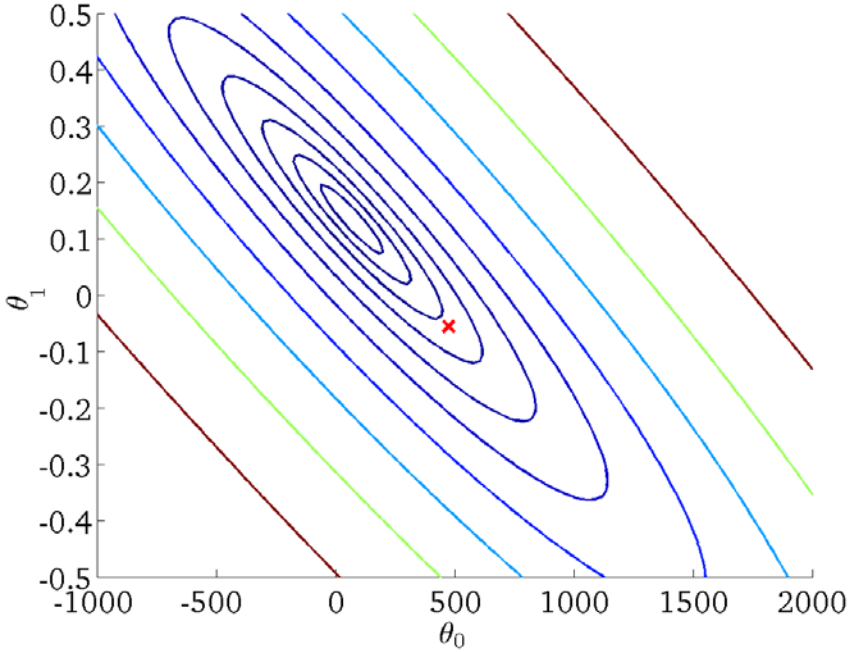
$h_\theta(x)$

(for fixed θ_0, θ_1 this is a function of x)



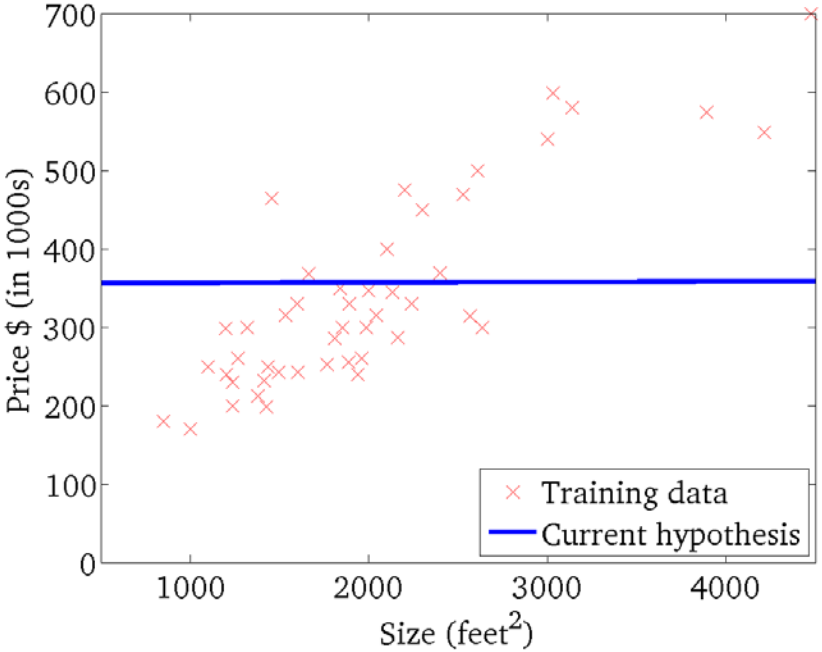
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



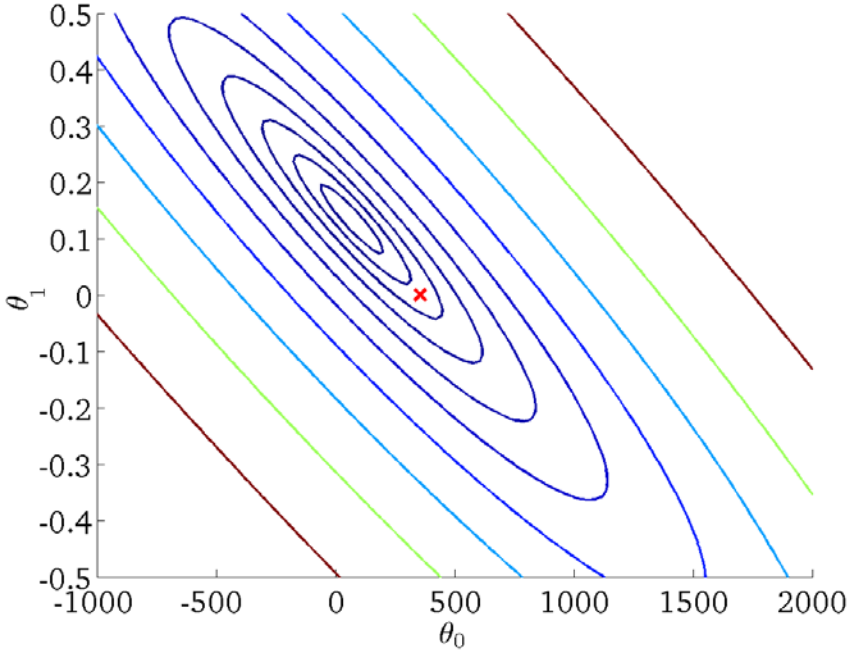
$h_\theta(x)$

(for fixed θ_0, θ_1 this is a function of x)



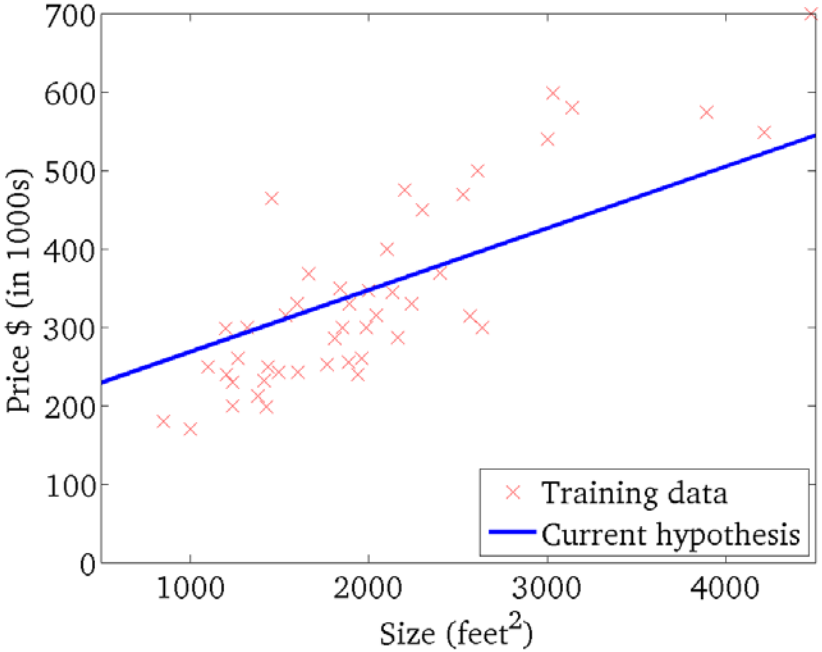
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



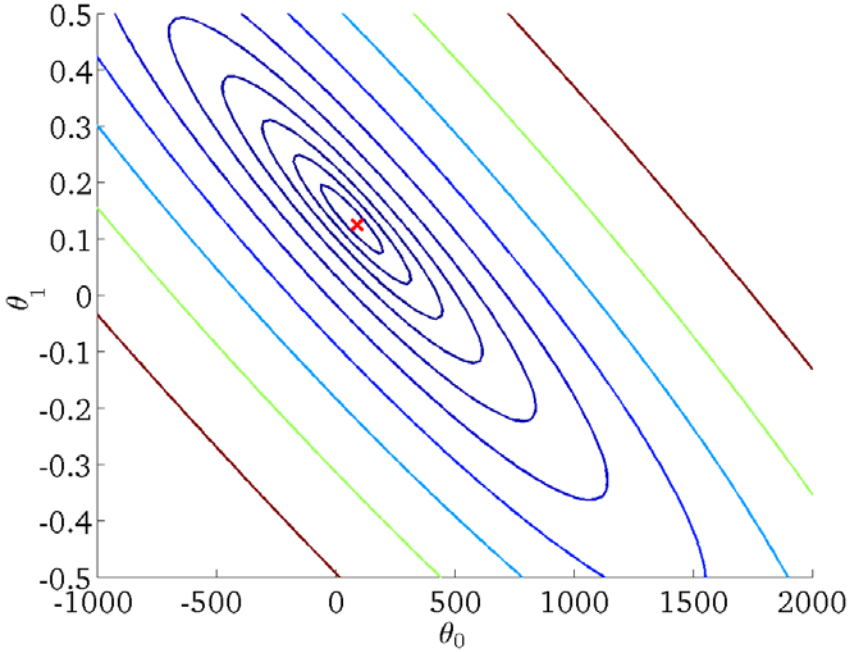
$h_\theta(x)$

(for fixed θ_0, θ_1 this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



Why Linear Functions?

Linear relationships are easy to understand, and *grossly* appropriate as a default model:

- Income grows linearly with time worked.
- Housing prices grow linearly with area.
- Weight increases linearly with food eaten.

Multiple features (variables).

Size (feet^2)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

Notation:

n = number of features

$x^{(i)}$ = input (features) of i^{th} training example.

$x_j^{(i)}$ = value of feature j in i^{th} training example.

Hypothesis:

For single variable:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Multivariate linear regression:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

For convenience of notation, define $x_0 = 1$.

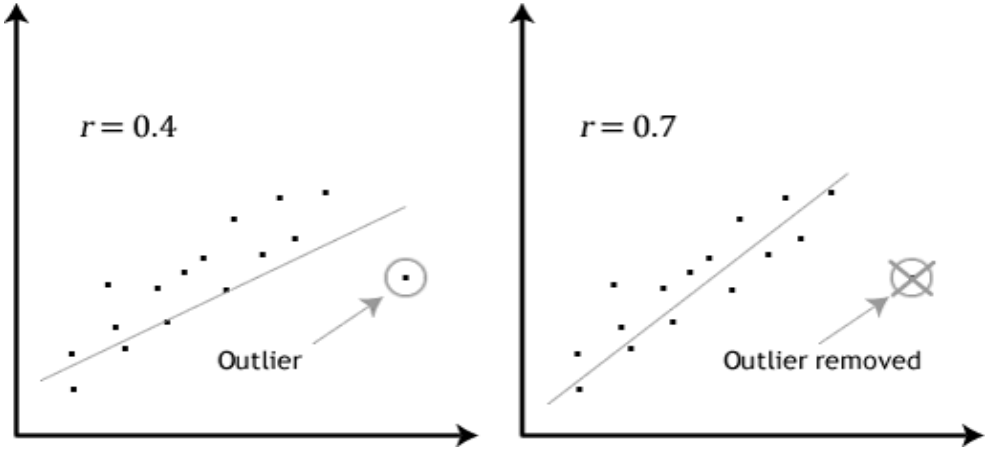
Better Regression Models

Proper treatment of variables yields better models:

- Removing outliers
- Fitting nonlinear functions
- Feature/target scaling
- Collapsing highly correlated variables

Outliers and Linear Regression

Because of the quadratic weight of residuals, outlying points can greatly affect the fit.



Identifying outlying points and removing them in a principled way can yield a more robust fit.

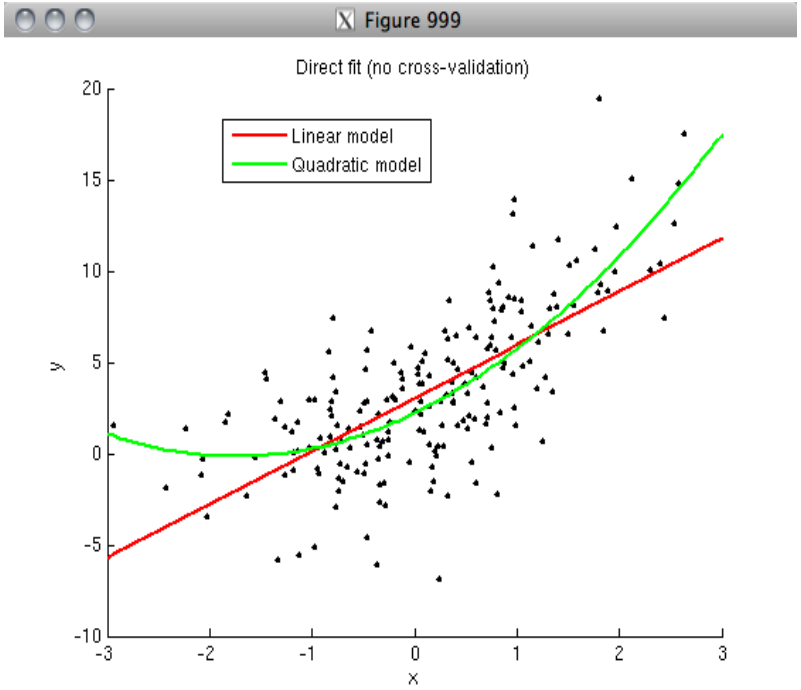
Fitting Non-Linear Functions

Linear regression fits lines, not high-order curves!

But we can fit quadratics by creating another variable with the value x^2 to our data matrix.

We can fit arbitrary polynomials (including square roots) and exponentials/logarithms by explicitly including the component variables in our data matrix: *sqrt(x)*, *lg(x)*, *x^3*, *1/x*.

However explicit inclusion of all possible non-linear terms quickly becomes intractable.



Feature Scaling: Z-scores

Features over wide numerical ranges (say national population vs. fractions) require coefficients over wide scales to bring together.

$$V = c1 * 300{,}000{,}000 + c2 * 0.02$$

Fixed learning rates (step size) will over/under shoot over such a range, in gradient descent.

Scale the features in your matrix to Z-scores!

Dominance of Power Law Features

Consider a linear model for years of education, which ranges from 0 to 12+4+5=19.

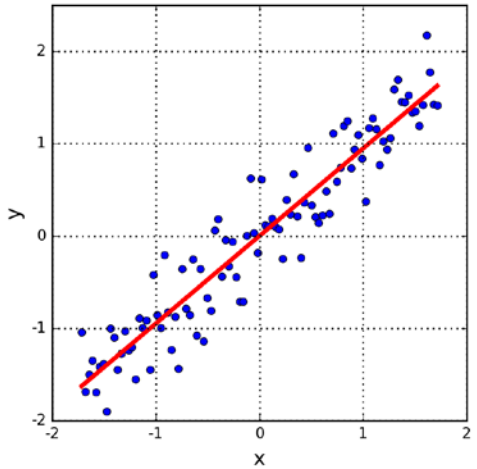
$$Y = c1 * income + c2$$

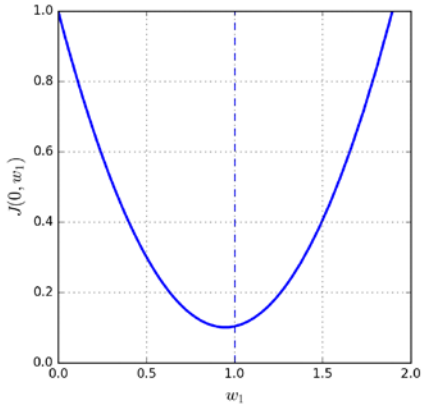
No such model can gives sensible answers for both my kids and Bill Gates' kids.

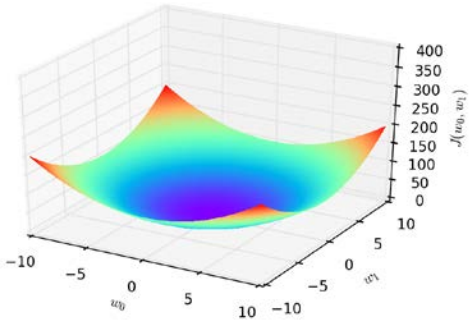
Z-scores of such power law variables don't help because they are just a linear transformation.

Lines in Parameter Space

The error function $J(\theta_0, \theta_1)$ is convex, making it easy to find the single local/global minima.







Gradient Descent for Regression

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

(for $j = 1$ and $j = 0$)

}

Linear Regression Model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Regularization

The trick is to add terms to the objective function seeking to keep coefficients small:

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

We pay a penalty proportional to the sum of squares of the coefficients, thus ignoring sign.

This rewards us for setting coefficients to zero.

LASSO (Least Absolute Shrinkage and Selection Operator)

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{i=1}^{n} |\theta_j|\right]$$

- Tends to select sparse solution
- Performs both variable selection and regularization
- Enhance interpretability

Logistic Regression

Classification Problems

Often we are given collections of examples *labeled* by class:

- male / female?
- democrat / republican?
- spam / non-spam?
- cancer / benign?

Classification assigns a label to an input record.

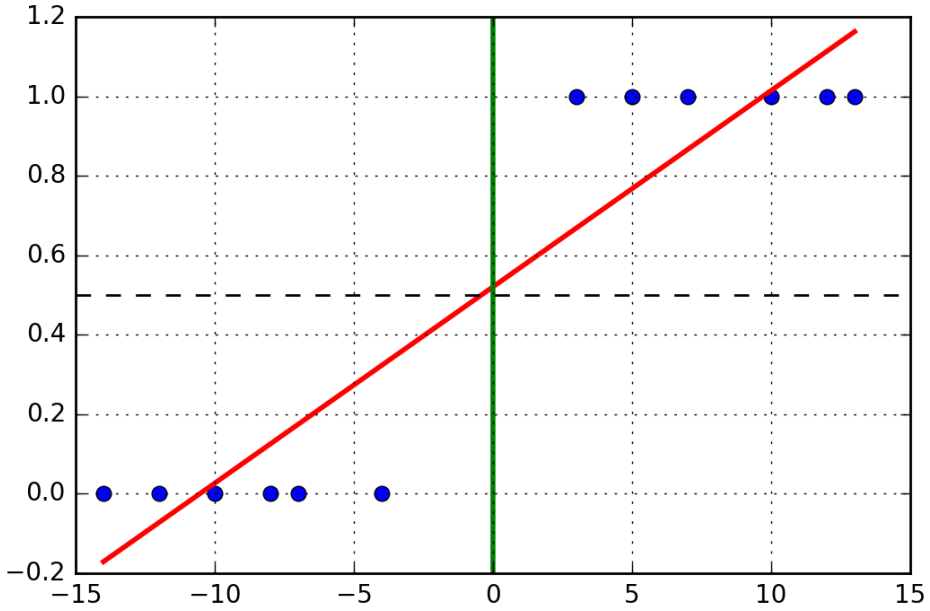
Regression for Classification

We *could* use linear regression to build from annotated examples by converting the class names to numbers:

- male=0 / female=1
- democrat=0 / republican=1
- spam=1 / non-spam=0
- cancer=1 / benign=0

Zero/one works for binary classifiers. By convention, the “positive” class gets 1 and the “negative” one 0.

Class Labels from Regression Lines

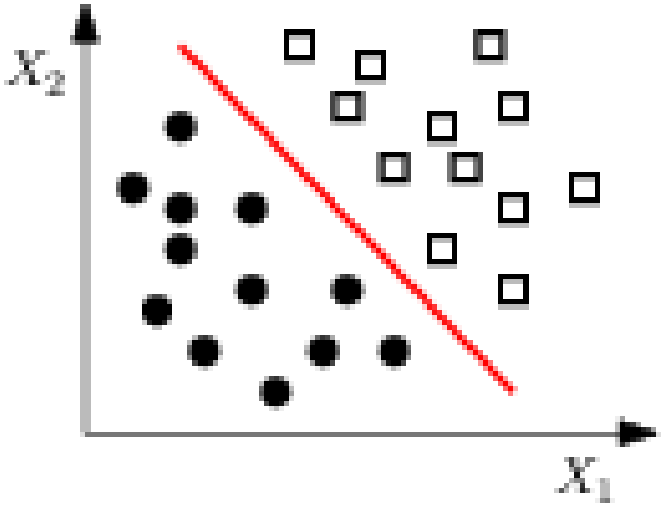


The regression line will cut through these classes.

Adding very +/- examples shifts the line and hurts the boundary

Decision Boundaries

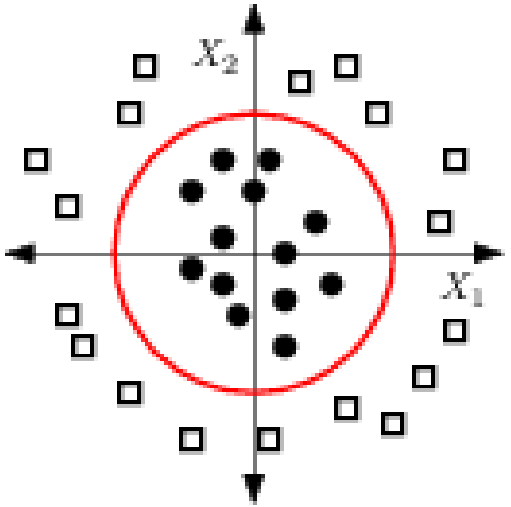
Ideally, our two classes will be well-separated in feature space, so a line can partition them.



Logistic regression is a method to find the best separating line for a given training set.

Non-Linear Decision Boundaries

Logistic regression can find non-linear boundaries if seeded with non-linear features.



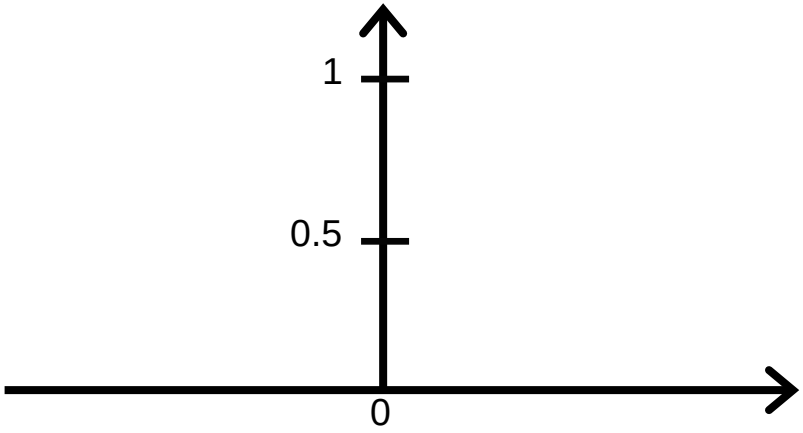
To get separating circles, we need to explicitly add features like x^2 and x1 * x2.

Logistic Regression Model

Want $0 \leq h_\theta(x) \leq 1$

$$h_\theta(x) = \quad \theta^T x$$

$$g(z) = \frac{1}{1+e^{-z}}$$



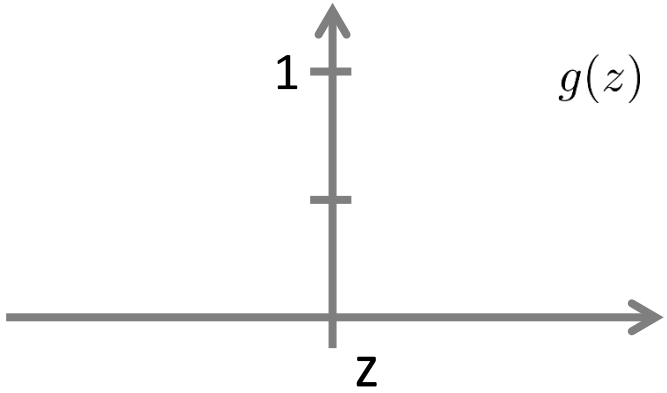
Sigmoid function

Logistic function

Logistic regression

$$h_\theta(x) = g(\theta^T x)$$

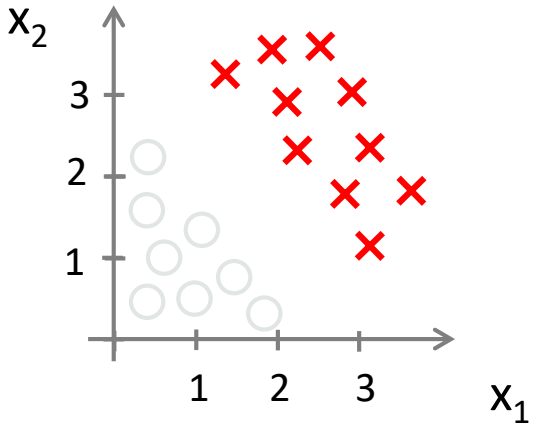
$$g(z) = \frac{1}{1+e^{-z}}$$



Suppose predict "$y = 1$" if $h_\theta(x) \geq 0.5$

predict "$y = 0$" if $h_\theta(x) < 0.5$

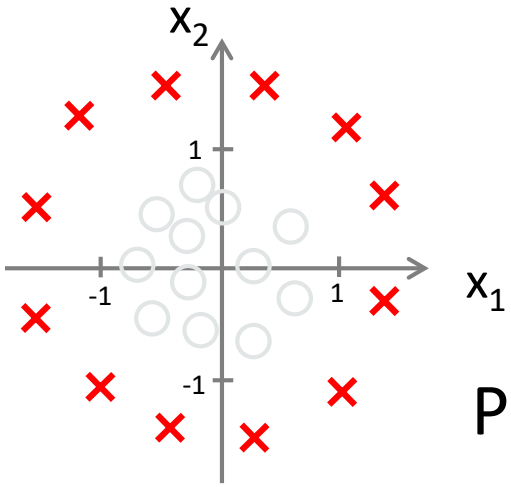
Decision Boundary



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

Non-linear decision boundaries



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$

Issues in Classification

- Balanced training class sizes
- Multi-class classification
- Hierarchical classification
- Partition functions

Balanced Training Classes

Consider the optimal separating line for grossly unbalanced class sizes, say 1 positive example vs. 1,000,000 negative examples.

The best scoring line from logistical regression will try to be very far from the big cluster instead of the midpoint between them.

Use equal numbers of pos and neg examples.

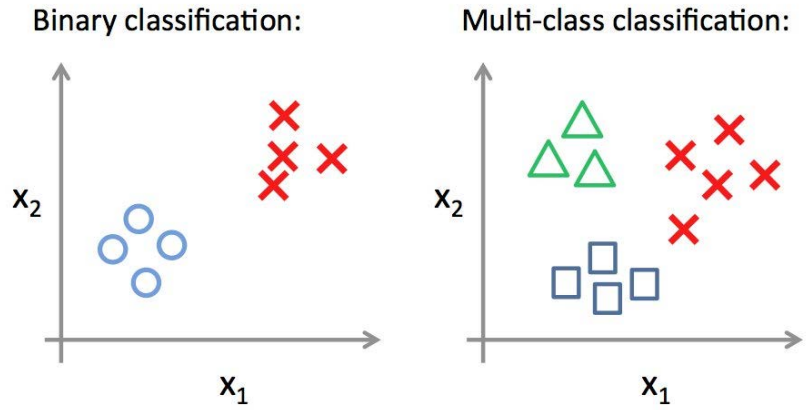
Ways to Balance Classes

- Work harder to find members of the minority class.
- Discard elements from the bigger class.
- Weigh the minority class more heavily, but beware of overfitting.
- Replicate members of the smaller class, ideally with random perturbation.

Multi-class Classification

Classification tasks are not always binary.

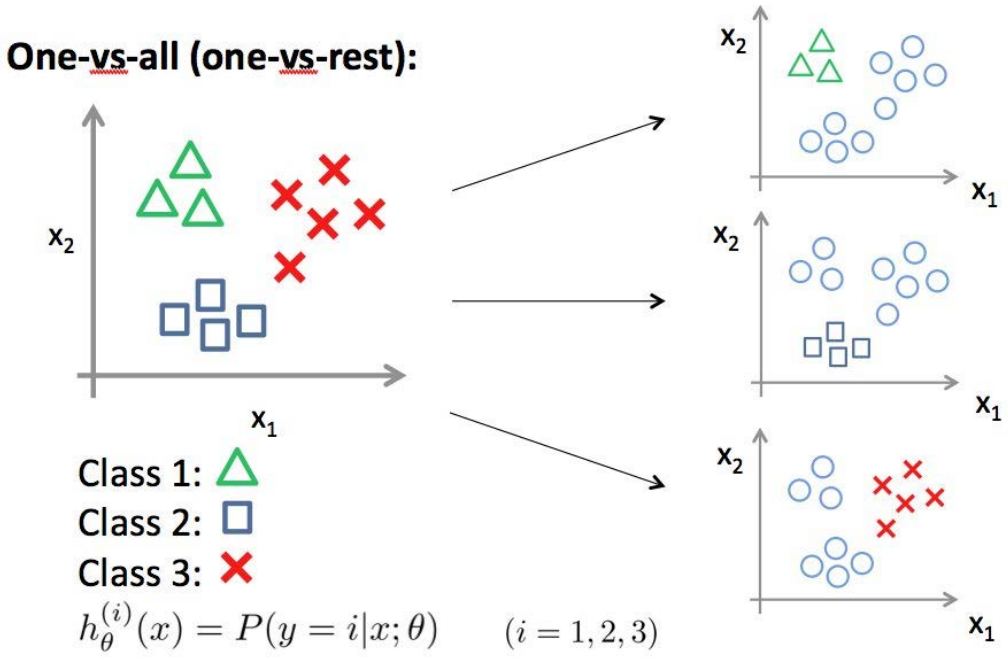
Is a given movie a comedy, drama, action, documentary, etc.?



One Versus All Classifiers

We can build multi-class classifiers by building multiple independent binary classifiers.

Select the class of highest probability as the predicted label.

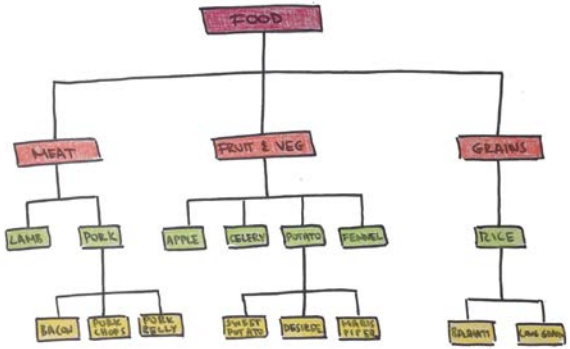


Hierarchical Classification

Grouping classes by similarity and building a taxonomy reduces the effective number of classes.

Imagenet has 27 categories (e.g. animal, appliance, food, furniture) on top of 21,841 subcategories.

Classify from top-down in tree.

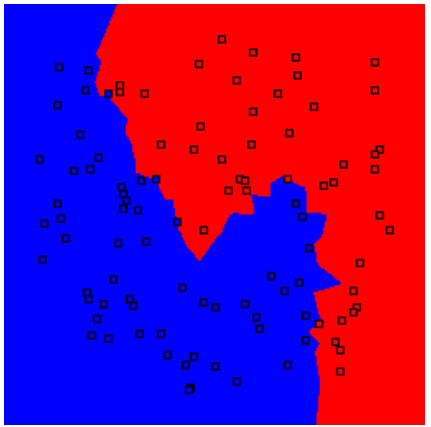


Nearest Neighbor Classification

Identify which training example is most similar to the target, and take the class label from it.

The key issue here is devising the right distance function between rows/points.

Advantages: simplicity, interpretability, and non-linearity.



K-Nearest Neighbors

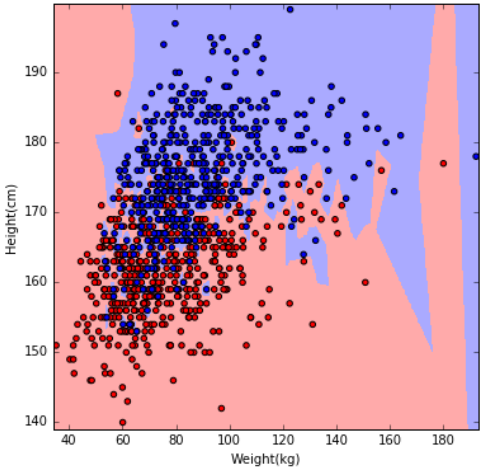
NN classification produce non-linear classifiers, because each training point changes the separating boundary.

More robust classification or interpolation follow from voting over the *k* closest neighbors for *k>1*.

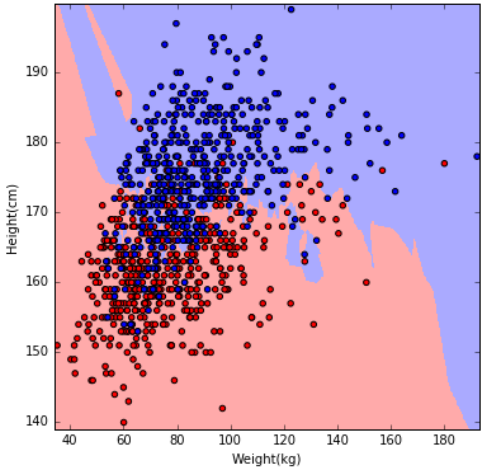
Gender Classification by Height/Weight

Smoother boundaries follow from larger k:

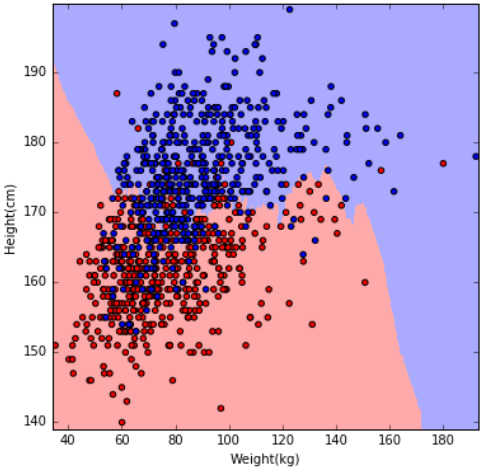
k=1



k=3



k=10



Unsupervised Learning

Supervised / Unsupervised Learning

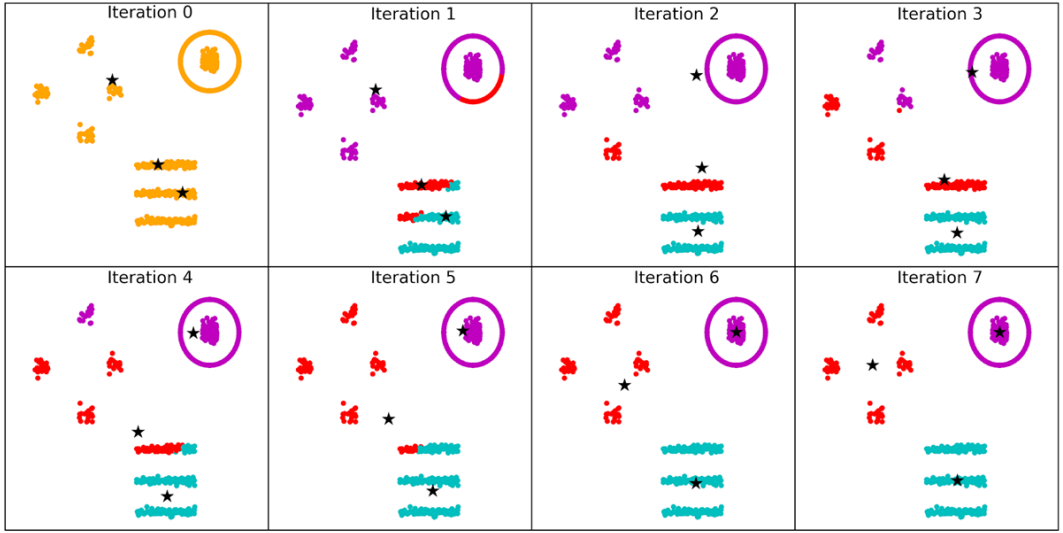
The methods discussed so far assume class labels or target variables in the training data.

Unsupervised methods try to find structure in the data, by providing labels (clusters) without a trusted standard.

Semi-supervised methods amplify small amounts of labeled data into more.

K-Means Clustering Example

It can get stuck in local optima, but generally does pretty well.

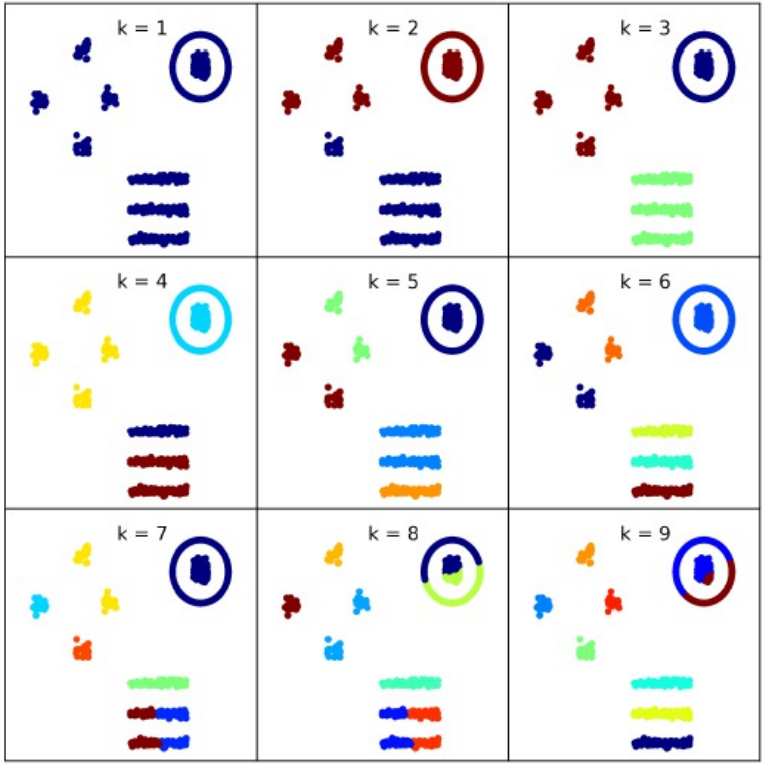


Limitations of K-means

K-means wants round clusters, so it has trouble with:

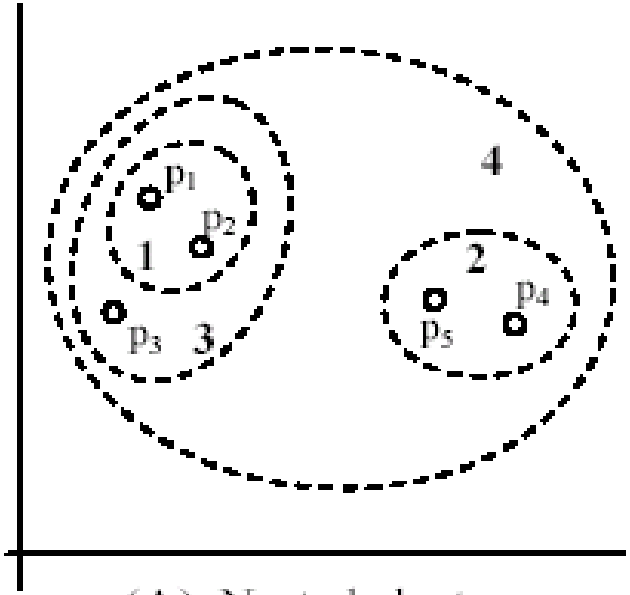
- nested clusters, and
- long thin clusters.

Repeated runs help avoid local optima.

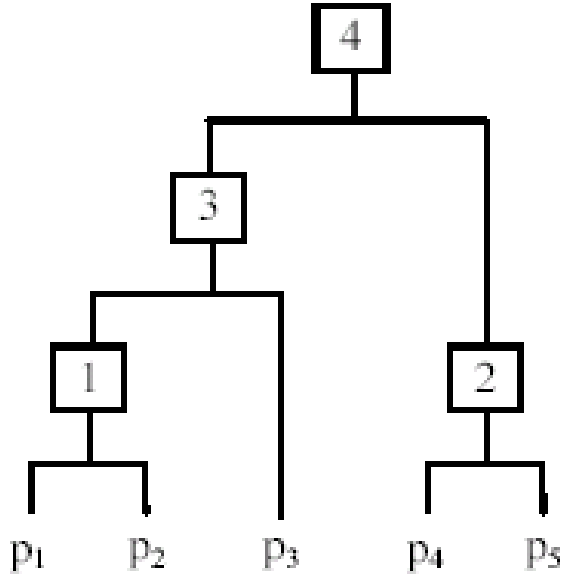


Agglomerative Clustering

These bottom-up methods repeatedly merge the two nearest clusters.



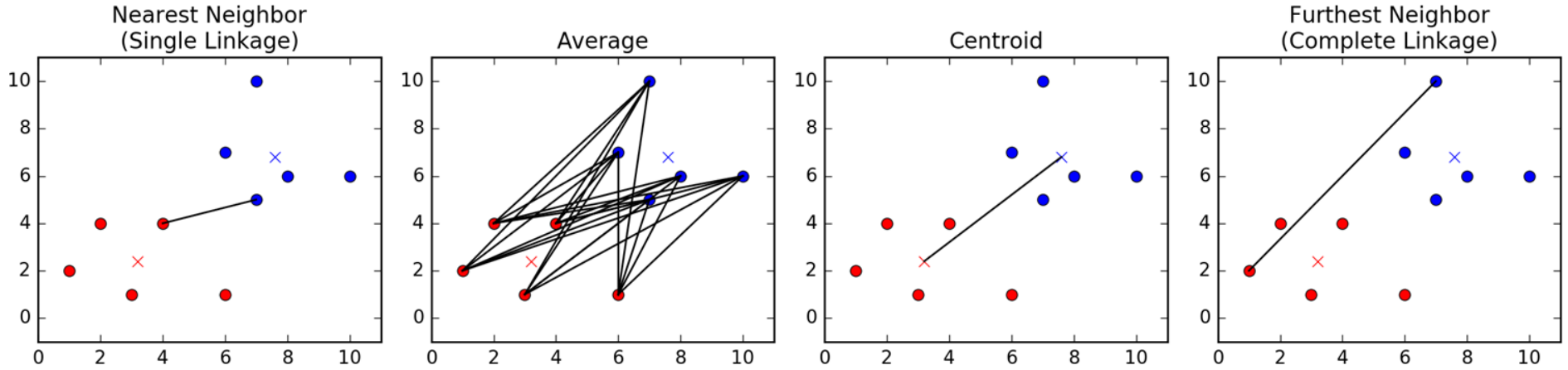
(A). Nested clusters



(B) Dendrogram

What Does Closest Cluster Mean?

The pointwise distance metric is not enough to define distance between clusters:



Which Clustering Algorithm To Use?

There are an enormous number of possible clustering algorithms, but much more important decisions are:

- using the right distance function
- properly normalizing your variables
- appropriately visualizing the final clusters to know whether they are good.

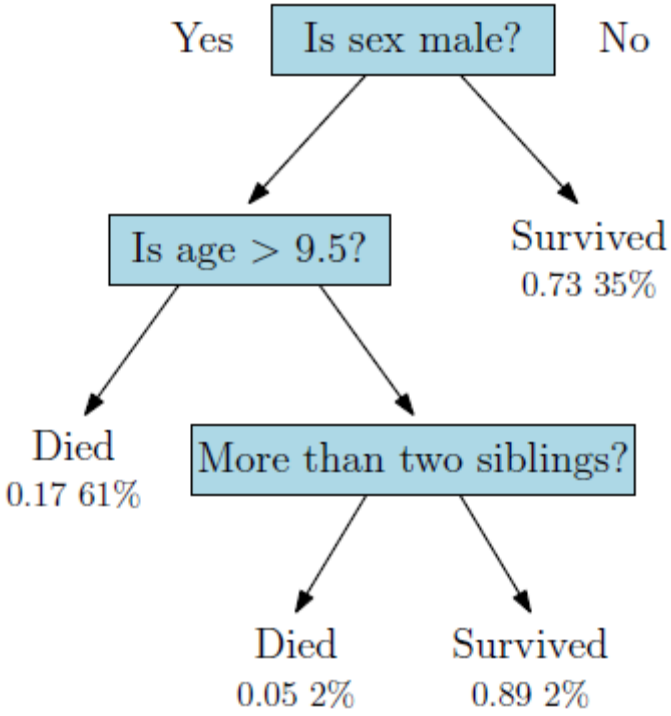
Other Machine Learning Models

Decision Tree Classifiers

Each row/instance travels a unique root-to-leaf path to classification.

The tree partitions the training examples into groups of relatively uniform composition, where the decision becomes easy.

Reducing each training example to its own leaf node implies over training.



Information-Theoretic Entropy

Entropy measures the amount of class confusion:

$$H(S) = -\sum_{i=1}^{m} f_i \log_2 f_i$$

- Complete disorder means $f_i = 1/m$, so $H(S)=\log(m)$.
- Perfect order means $f_1=1$ and $f_2=0$, so $H(S)=0$.

Split Criteria

The value of a potential split *S* is how much it reduces the entropy of the system:

- Information gain

$$IG_p(S) = H(S) - \sum_{j=1}^{2} \frac{|S_i|}{|S|} H(S_i)$$

Stopping Criteria

Building a tree until each leaf is pure likely ends with singleton elements, and is overfit.

- Better is to stop when the information gain < ϵ, instead of zero.
- An alternate strategy is to build the full tree, then prune low value nodes.

Decision tree construction is hacking, not science.

Advantages of Decision Trees

- Non-linearity
- Support for categorical variables (hair=red)
- Interpretability -- people can read the tree
- Robustness -- we can construct ensembles of different trees and vote (CART)

Biggest disadvantage is lack of elegance/*Math*.

Ensembles of Decision Trees

We can construct hundreds of decision trees by randomly selecting the feature to split on.

- Voting among multiple classifiers increases robustness and let us score our trust level.
- Bagging picks randomly selected subsets of items to train each tree on.

Boosting

Boost weak (but >0.5 accuracy) classifiers in a strong classifier.

To set the weights of the classifier, we will adjust the weights of the training examples.

Easy training examples will be properly classified by most classifiers: we reward classifiers more for getting the hard cases right.

Boosting: Pro and Con

Boosting provides a way to take advantage of weak classifiers (small correlation features) in an effective way.

Boosting tries to fit every example, thus it will overfit noisy data. “Hard cases make bad law.”

Gradient boosted decision trees are the most common method in Kaggle competitions.

Bayesian Classifiers

To classify a vector $X = (x_1, \ldots x_n)$ into one of *m* classes, we can use Bayes Theorem:

$$p(C_i|X) = \frac{p(C_i)p(X|C_i)}{p(X)}$$

This reduces decisions about the class given the input to the input given the class.

Independence and Naive Bayes

But what is $P(X|C)$, where X is a complex feature vector?

If (a,b) are independent, then $P(ab)=P(a)\ P(b)$

This calculation is much simpler than factoring in correlations and interactions of multiple factors.

Complete Naive Bayes Formulation

We seek the argmax of:

$$C(X) = \max_{i=1}^{m} p(C_i)p(X|C_i) = \max_{i=1}^{m} p(C_i) \prod_{j=1}^{n} p(x_j|C_i)$$

Multiplying many probabilities is bad, so:

$$C(X) = \max_{i=1}^{m} (\log(p(C_i)) + \sum_{j=1}^{n} \log(p(x_j|C_i)))$$

Naïve Bayes Classifier Example

Day	Outlook	Temp	Humidity	Beach?
1	Sunny	High	High	Yes
2	Sunny	High	Normal	Yes
3	Sunny	Low	Normal	No
4	Sunny	Mild	High	Yes
5	Rain	Mild	Normal	No
6	Rain	High	High	No
7	Rain	Low	Normal	No
8	Cloudy	High	High	No
9	Cloudy	High	Normal	Yes
10	Cloudy	Mild	Normal	No

P(X|Class)	Probability in Class	
Outlook	Beach	No Beach
Sunny	3/4	1/6
Rain	0/4	3/6
Cloudy	1/4	2/6
Temperature	Beach	No Beach
High	3/4	2/6
Mild	1/4	2/6
Low	0/4	2/6
Humidity	Beach	No Beach
High	2/4	2/6
Normal	2/4	4/6
P(Beach Day)	4/10	6/10

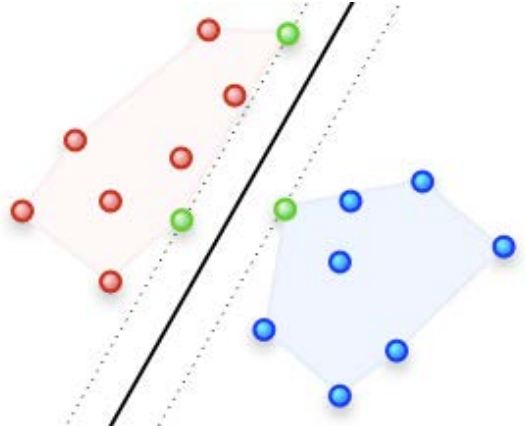
Is a Sunny-Mild-High a Beach Day?

$$
\begin{aligned}
&P(\text{Beach}|(\text{Sunny},\text{Mild},\text{High})) \\
&\quad = (P(\text{Sunny}|\text{Beach}) \times P(\text{Mild}|\text{Beach}) \times P(\text{High}|\text{Beach}) \times P(\text{Beach}) \\
&\quad = (3/4) \times (1/4) \times (2/4) \times (4/10) = 0.0375
\end{aligned}
$$

$$
\begin{aligned}
&P(\text{No Beach}|(\text{Sunny},\text{Mild},\text{High})) \\
&\quad = (P(\text{Sunny}|\text{No}) \times P(\text{Mild}|\text{No}) \times P(\text{High}|\text{No})) \times P(\text{No}) \\
&\quad = (1/6) \times (2/6) \times (2/6) \times (6/10) = 0.0111
\end{aligned}
$$

Support Vector Machines

SVMs are an important way to build non-linear classifiers.

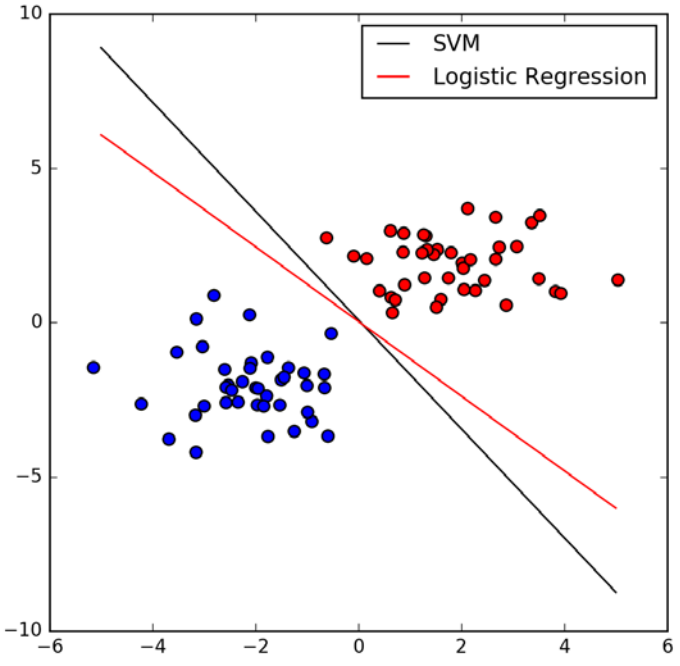


They work by seeking maximum margin linear separators between the two classes.

SVMs vs. Logistic Regression

Both methods find separating planes, but different ones.

LR values all points, but SVM only the points at the boundary.

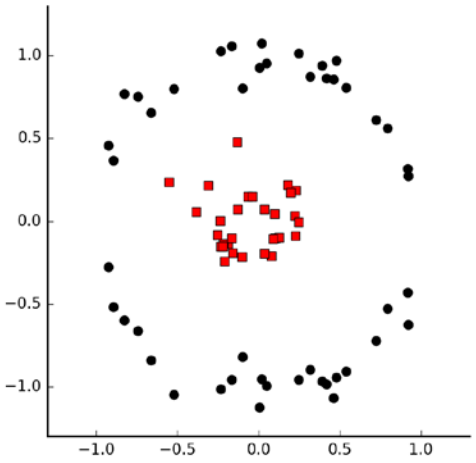


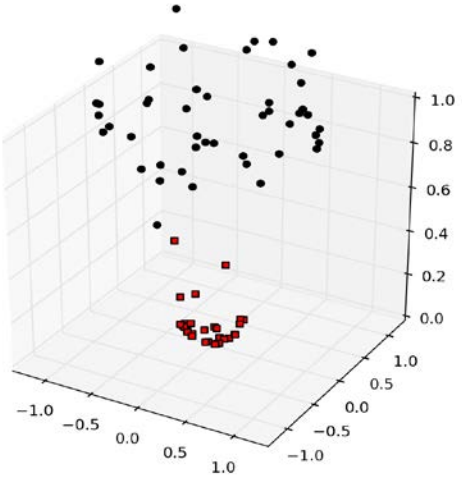
Projecting to Higher Dimensions

Adding enough dimensions makes everything linearly separable.

Here (x,y) ->(x,y,x^2+y^2) does the job.

Efficient solvers like LibSVM are available for this.





Feature Engineering

Domain-dependent data cleaning is important:

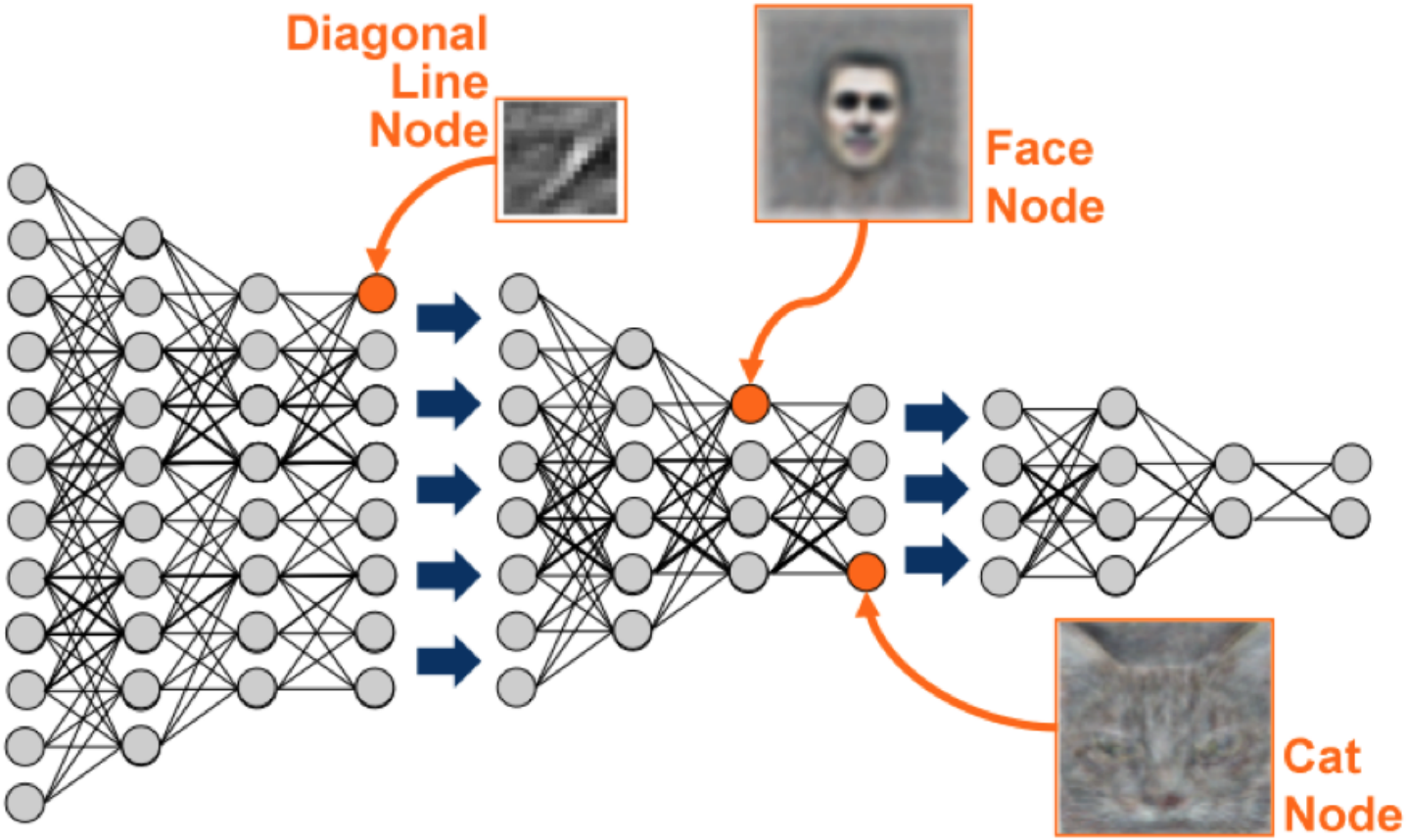
- Z-scores and normalization
- Creating bell-shaped distributions.
- Imputing missing values
- Dimension reduction, like SVD
- Explicit incorporation of non-linear combinations like products and ratios.

Basic Principles of Deep Learning

- That the weight of each edge is a distinct parameter means large networks exploits large training sets.
- The depth of the networks means they can build up hierarchical representations of features: e.g. pixels, edges, regions, objects
- Toolkits like TensorFlow make it easy to build DL models if you have the data.

Deep Learning

Lots of Data + Neural Nets + Training = Hierarchical & Associational Feature Representation



Backpropagation

NNs are trained by a stochastic gradient descent-like algorithm, with changes for each training example pushed down to lower levels.

The non-linear functions result in a non-convex optimization function, but this generally produces good results.

Dealing with Big Data

How Big Is… (2016)

- Twitter (600 million tweets/day)
- Facebook (>600 TB incoming data per day)
- Google (3.5 billion search queries/day)
- Instagram (52 million photos per day)
- Apple (130 billion app downloads)
- Netflix (125 million hours of TV shows and movies streamed daily)
- Email (205 billion message/day)

http://www.internetlivestats.com

Big Data as Bad Data

Massive data sets are the result of opportunity instead of design, with problems of:

- Unrepresentative participation (bias)
  - Instagram (too young), The New York Times (too liberal), Fox News (too conservative), or The Wall Street Journal (too wealthy).
- Spam and machine-generated content
  - e.g., bots, fake product reviews.

Big Data as Bad Data

- Power-laws mean too much redundancy
  - 1000s of Empire State Building photo but none for many buildings
- Susceptibility to temporal bias (e.g Google Flu Trends)
  - auto-complete mechanism changed the distribution of search queries

Filtering Data

An important benefit of Big Data is that you can discard much of it to make analysis cleaner.

English accounts for only 34% of all tweets on Twitter, but you can exclude the rest and leave enough for meaningful analysis.

Filtering away irrelevant or hard-to-interpret data requires application-specific knowledge.

Subsampling Data

It can pay to subsample good, relevant data:

- Cleanly separate training, testing, and evaluation data.
- Simple, robust models generally have few parameters, making Big Data is overkill.
- Spreadsheet-sized data sets are fast and easy to explore.

Data Parallelism

The easiest way to exploit parallelism partitions big data among multiple machines and trains independent models.

It is typically hard to combine the results of these runs together later (think k-means)

Grid Search

The easiest way to exploit parallelism involves independent runs on independent data.

Grid search is the quest for the right hyper-parameters for training, like deciding the right *k* for k-means clustering.

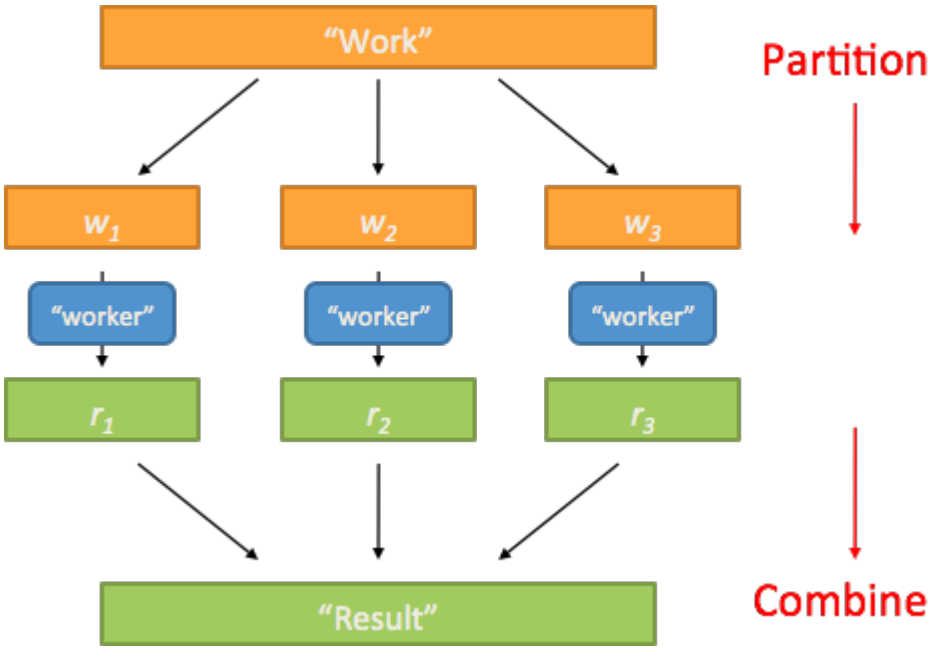
Multiple independent fits can run in parallel, where in the end we take the best one.

MapReduce / Hadoop

Google’s MapReduce paradigm for distributed computing has spread widely through the open-source implementation like Hadoop and Spark, offering:

- Simple parallel programming model
- Straightforward scaling to hundreds/thousands of machines.
- Fault tolerance through redundancy

Divide and Conquer



Ideas Behind MapReduce

- Scale “out”, not “up”: recognize limits of large shared-memory machines
- Move processing to the data: clusters have limited bandwidth
- Process data sequentially, avoid random access: seeks are expensive, disk throughput is reasonable

Components of Hadoop

- Core Hadoop has two main systems:
  - **Hadoop/MapReduce**: distributed big data processing infrastructure (abstract/paradigm, fault-tolerant, schedule, execution)
  - **HDFS (Hadoop Distributed File System)**: fault-tolerant, high-bandwidth, high availability distributed storage

Map and Reduce

- Programmers specify two functions:

map (k, v) → [(k', v')]

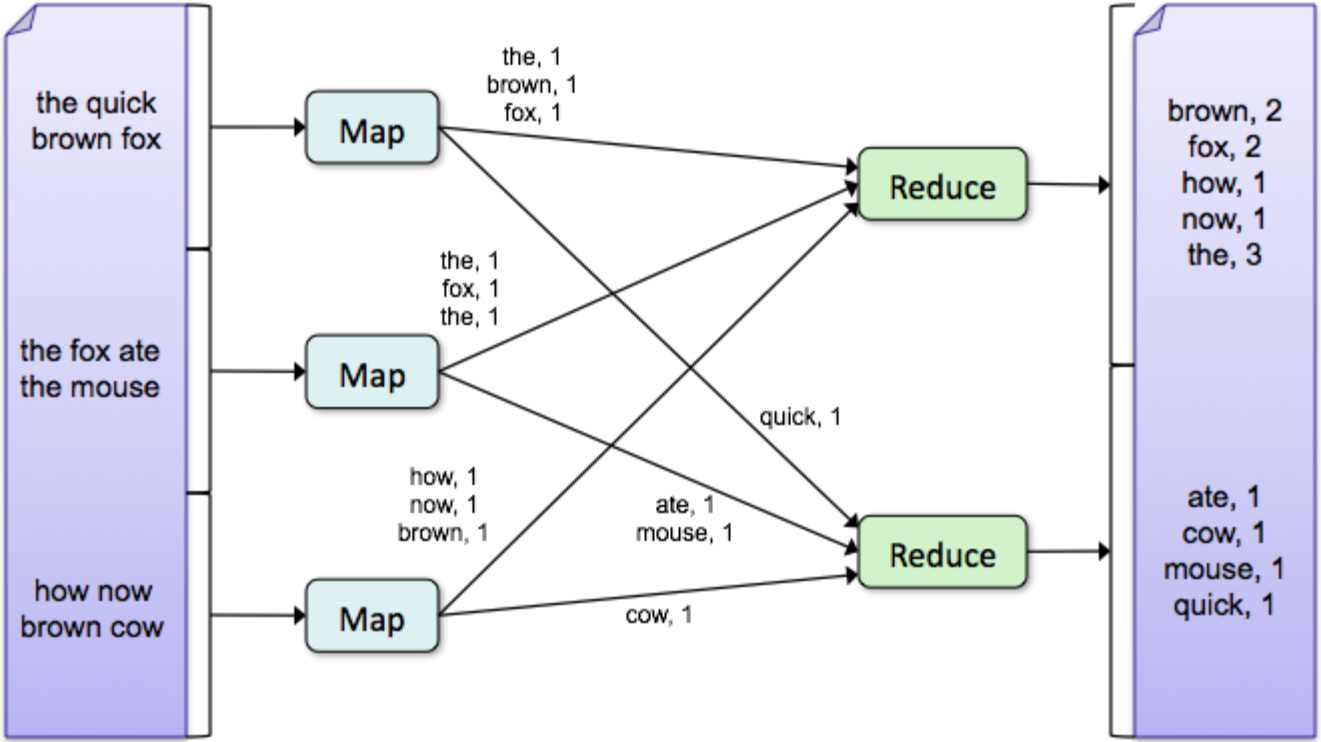
reduce (k', [v']) → [(k', v')]

– All values with the same key are sent to the same reducer

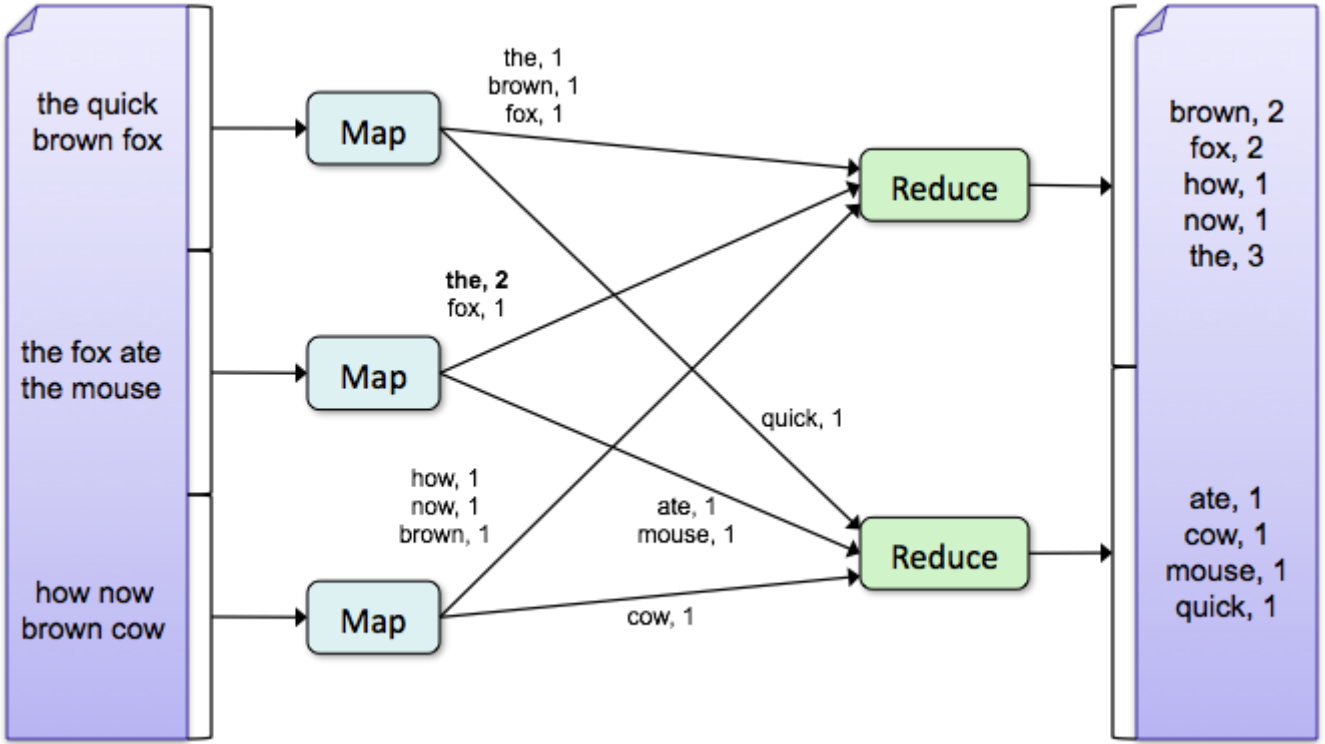
MapReduce Word Count

```
Map(String docid, String text):
    for each word w in text:
        Emit(w, 1);
Reduce(String term, Iterator<Int> values):
    int sum = 0;
    for each v in values:
        sum += v;
    Emit(term, sum);
```

Word Count Execution



Word Count with Combiner



Hadoop Distributed File System (HDFS)

Store data on the local disks of nodes in the cluster, because not enough RAM to hold all the data in memory

Disk access is slow, but disk throughput is reasonable, so linear scans through files are fine.

Replicate everything 3 times for reliability on commodity hardware.

Cloud Computing Services

Platforms like Amazon make it easy to rent large numbers of machines for short-term jobs.

There are charges on bandwidth, processors, memory, long-term storage: making it non-trivial to price exactly.

Feel Free to Experiment

Micro instances are only 1GB, single processor virtual machines.

Reasonable machines rent for 10 to 30 cents/hr.

Free Tier*

As part of AWS's Free Usage Tier, new AWS customers can get started with Amazon EC2 for free. Upon sign-up, new AWS customers receive the following EC2 services each month for one year:

- 750 hours of EC2 running Linux, RHEL, or SLES t2.micro instance usage
- 750 hours of EC2 running Microsoft Windows Server t2.micro instance usage
- 750 hours of Elastic Load Balancing plus 15 GB data processing
- 30 GB of Amazon Elastic Block Storage in any combination of General Purpose (SSD) or Magnetic, plus 2 million I/Os (with Magnetic) and 1 GB of snapshot storage
- 15 GB of bandwidth out aggregated across all AWS services
- 1 GB of Regional Data Transfer