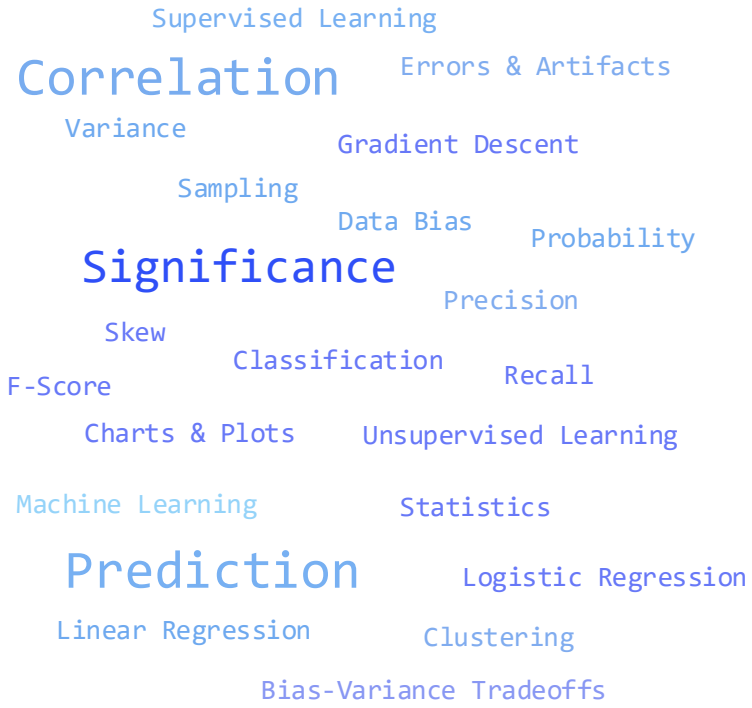


Data Science 1:
Introduction to Data Science

Intro to Machine Learning

Winter 2025

Wolfram Wingerath, Jannik Schröder

Department for Computing Science
Data Science / Information Systems

Lecture slides based on content from "The Data Science Design Manual" (Steven Skiena, 2017) and associated course materials generously made available online by the author at https://www3.cs.stonybrook.edu/~skiena/data-manual/.

Special thanks to Professor Skiena for sharing these valuable teaching resources!

Semester Schedule

CW 42	14. Oct	Lecture	1	Orga & Intro	1-26
CW 43	21. / 23. Oct	Lecture + Exercises	2	Probability, Statistics & Correlation	27-56
CW 44	28. Oct	Lecture	3	Data Munging, Cleaning & Bias	57-94 / “Invisible Women”
CW 45	04. / 06. Nov	Lecture + Exercises	4	Scores & Rankings	95-120
CW 46	11. Nov	Lecture	5	Statistical Distributions & Significance	121-154
CW 47	18. / 20. Nov	Lecture + Exercises	6	Building & Evaluating Models	201-236
CW 48	25. Nov	Guest Lecture	7	Data Visualization	155-200
CW 49	02. / 04. Dec	Lecture + Exercises	8	Intro to Machine Learning	351-390
CW 50	09. Dec	Lecture	9	Linear Algebra	237-266
CW 51	16. / 18. Dec	Lecture + Exercises	10	Linear Regression & Gradient Descent	267-288
CW 02	06. Jan	Lecture	11	Logistic Regression & Classification	289-302
CW 03	13. / 15. Jan	Lecture + Exercises	12	Nearest Neighbor Methods & Clustering	303-350
CW 04	20. Jan	Lecture	13	Data Science in the Wild	391-426
CW 05	27. / 29. Jan	Lecture + Exercises	14	Q&A / Feedback	
CW 06	03. / 04. Feb	Oral Exams (Block 1)			
CW 13	24. / 25. Mar	Oral Exams (Block 2)			

Preparation in our last session („Oral Exam Briefing“)

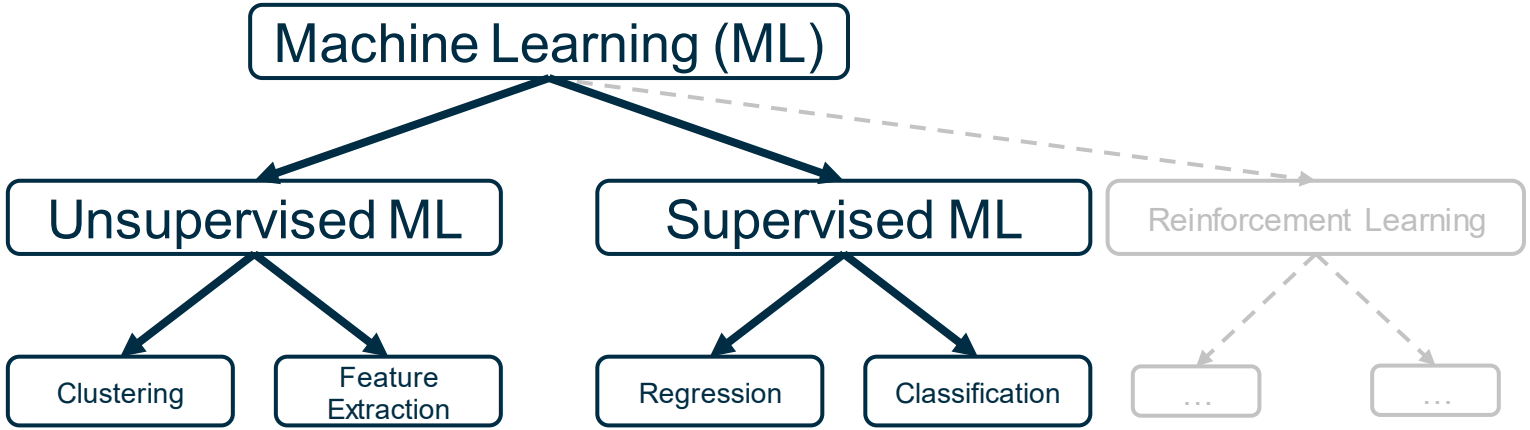
The Machine Learning (ML) Zoo

The variety of clever machine learning algorithms makes it easy to lose sight of why?

Evaluation dimensions include:

- Power and expressibility
- Interpretability
- Ease of Use
- Training speed
- Prediction speed

Machine Learning Overview



ML approaches can be classified by the way that the procedures work and how they use labeled (supervised) or unlabeled (unsupervised) data

Subjective Rankings

ML methods can be compared along different dimensions to show their pros and cons:

Method	Power of Expression	Ease of Interpretation	Ease of Use	Training Speed	Prediction Speed
Linear Regression	5	9	9	9	9
Nearest Neighbor	5	9	8	10	2
Naive Bayes	4	8	7	9	8
Decision Trees	8	8	7	7	9
Support Vector Machines	8	6	6	7	7
Boosting	9	6	6	6	6
Graphical Models	9	8	3	4	4
Deep Learning	10	3	4	3	7

Take Home Lessons

- Linear/logistic regression is pretty good, except in expressibility.
- No single method dominates all the others.
- The comparative advantages of most methods are relatively murky.

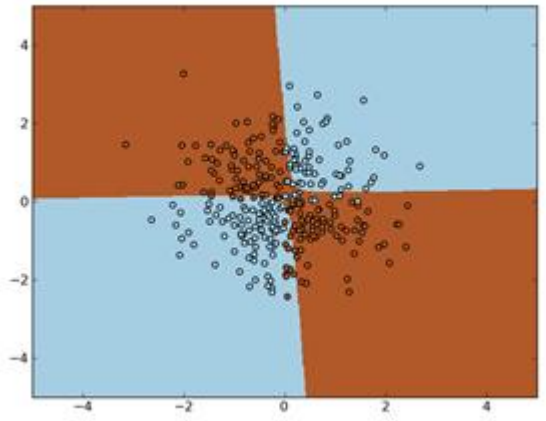
The no free lunch theorem states that no single method works best in all cases.

XOR and Linear Classifiers

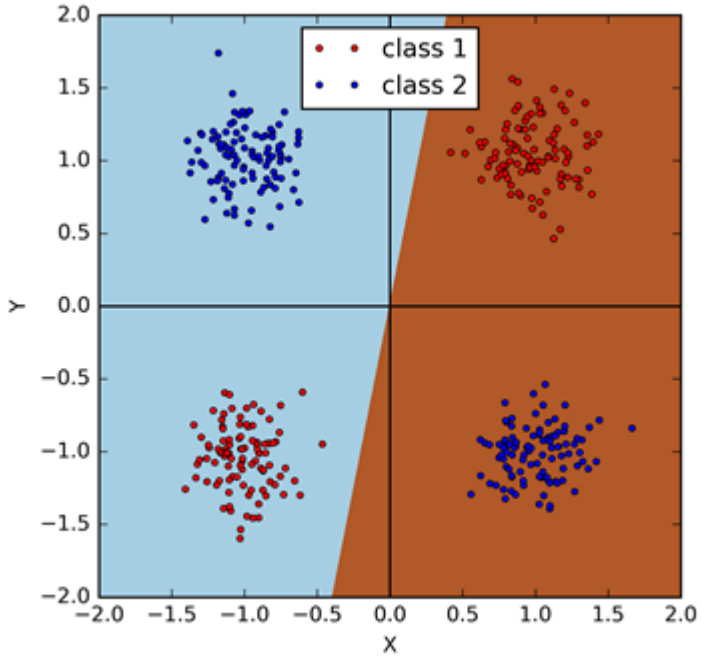
Linear classifiers cannot be used to fit simple non-linear functions like eXclusive OR.

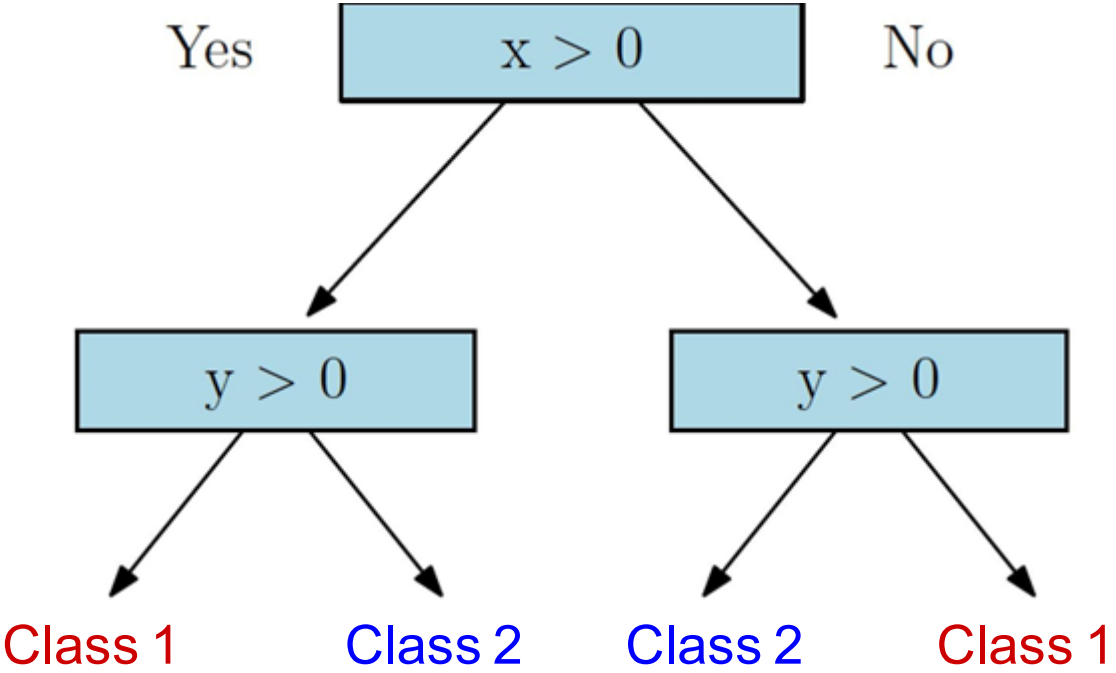
More powerful methods are needed, including:

- Decision trees
- Nearest neighbor methods
- Support vector machines



XOR and Decision Trees



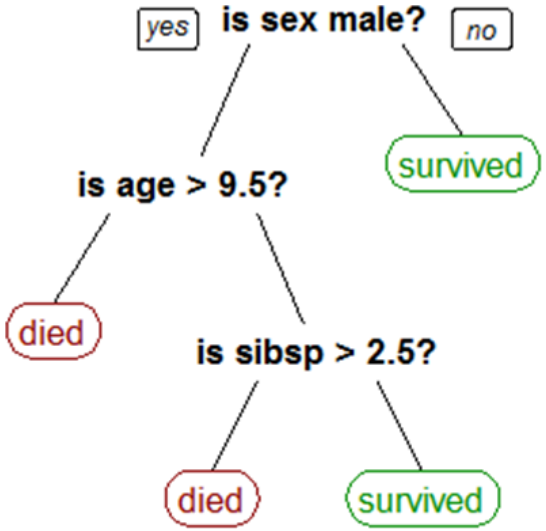


Decision Tree Classifiers

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 overtraining.



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)
- Applicability to regression within leaf subset

Biggest disadvantage is lack of elegance / *Math*

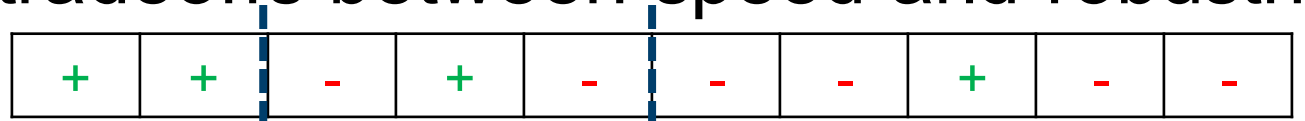
Constructing Decision Trees

Trees are constructed in a top-down manner.

Seek a split along one feature/dimension to purify the information over m classes.

- Pure splits create nodes of one single class
- Balanced splits partition the items into equal-sized groups.

Thus tradeoffs between speed and robustness.



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 $f_i = \frac{1}{m}$, 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

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

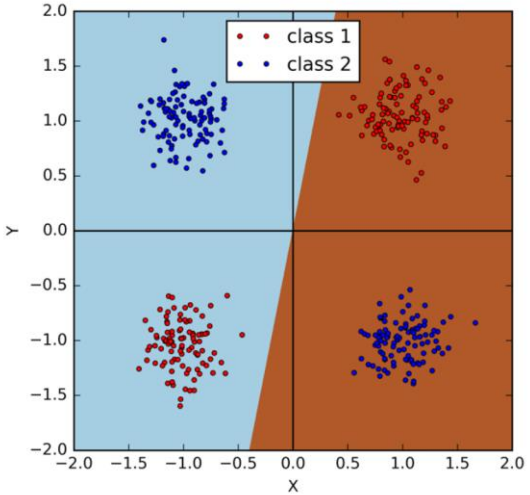
- Gini impurity

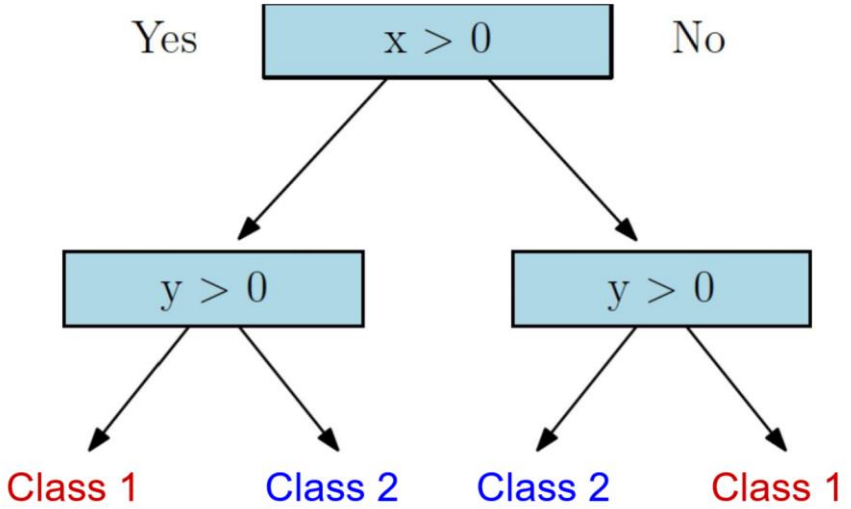
$$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i) = \sum_{i=1}^{m} (f_i - f_i^2)$$

$$= \sum_{i=1}^{m} f_i - \sum_{i=1}^{m} f_i^2 = 1 - \sum_{i=1}^{m} f_i^2$$

Look Ahead and Decision Trees

Finding the best XOR tree requires look ahead, as it cannot be found by greedy progression:





Stopping Criteria

Building a tree until each leaf is pure (f=1) likely ends with singleton elements, and is overfit.

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

Decision tree construction is hacking, not science.

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 lets us score our trust level.
- Bagging picks randomly selected subsets of items to train each tree on.

But should all trees get equal votes?

Voting Classifiers

The natural way to use multiple classifiers gives each a vote, and takes the majority label.

Such aggregation is typically done for random sets of decision trees, or even single features.

But should each classifier get the same vote?

Epicurus: *"Keep all theories that are consistent with the data"* (But not with equal votes)

Weighing Better Classifiers More

item/voter	V_1	V_2	V_3	V_4	V_5	majority	best weights
A	*		*	*	*	*	*
B	*		*	*	*	*	*
C	*	*		*	*	*	*
D	*	*					*
E		*	*				*
% correct	80%	60%	60%	60%	60%	60%	100%
best weight	1/3	1/3	1/3	0	0		

Might weigh by accuracy or linear regression.

But *better* means getting the hard cases right...

Boosting

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

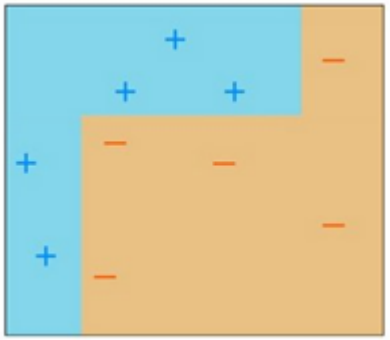
To set the weights of the classifier, we will adjust the weights of the training examples.

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

Adaboost (Adaptive Boosting)

We seek non-linear classifiers using thresholded features as classifiers (e.g x>1)

Initially all points are of equal weight.



- Samples $x_1 \ldots x_n$
- Desired outputs $y_1 \ldots y_n, y \in \{-1, 1\}$
- Initial weights $w_{1,0} \ldots w_{n,0}$ set to $\frac{1}{n}$
- Error function $E(f(x), y, i) = e^{-y_i f(x_i)}$
- Weak learners $h: x \rightarrow [-1, 1]$

Train data

x1	x2	y	D1
1	5	+	0.10
2	3	+	0.10
3	2	–	0.10
4	6	–	0.10
4	7	+	0.10
5	9	+	0.10
6	5	–	0.10
6	7	+	0.10
8	5	–	0.10
8	8	–	0.10
			1.00

Initialization

Boosting Misclassified Points

The weight of the misclassified points is boosted to make them more important in the next round.

$$w'_{i,t+1} = w_{i,t} e^{-y_i \alpha_t h_t(x_i)}$$

Normalize total weights to sum to 1.

The weight of the new classifier depends upon how accurate it is:

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

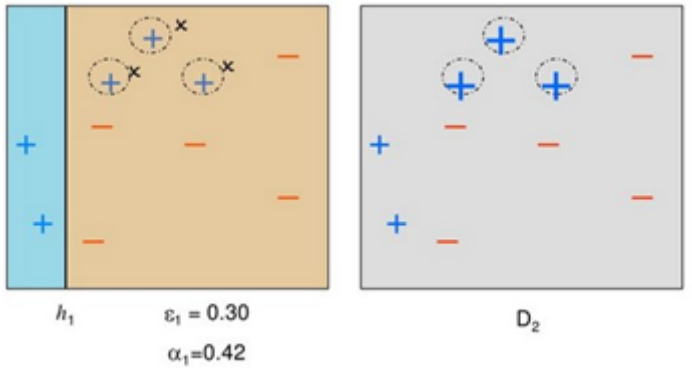
Train data

x1	x2	y
1	5	+
2	3	+
3	2	−
4	6	−
4	7	+
5	9	+
6	5	−
6	7	+
8	5	−
8	8	−

D1
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
1.00

Round 1

h1e	ε	D2
0	0.00	0.07
0	0.00	0.07
0	0.00	0.07
0	0.00	0.07
1	0.10	0.17
1	0.10	0.17
0	0.00	0.07
1	0.10	0.17
0	0.00	0.07
0	0.00	0.07
ε1	0.30	1.00
α1	0.42	↕
	Zt	0.92



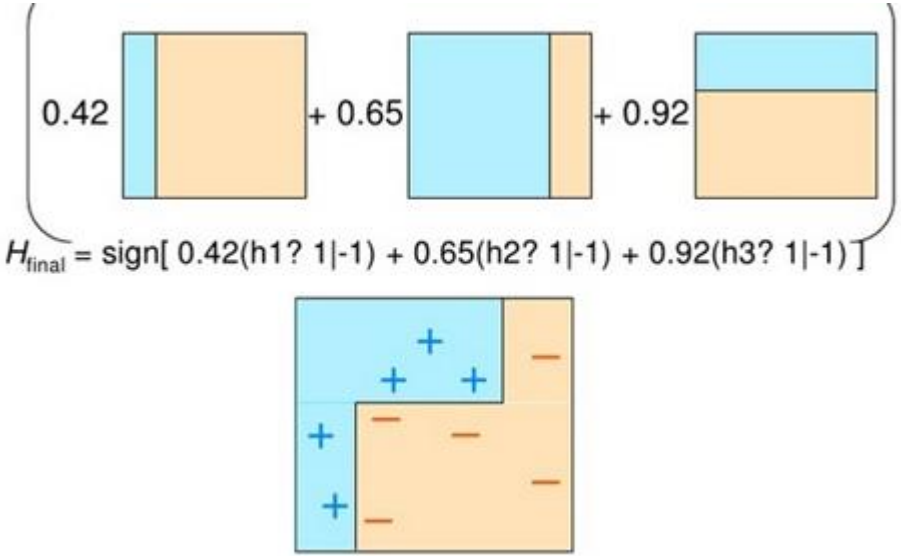
AdaBoost Algorithm

For t in $1 \ldots T$:

- Choose $f_t(x)$:
  - Find weak learner $h_t(x)$ that minimizes ϵ_t, the weighted sum error for misclassified points $\epsilon_t = \sum_i w_{i,t}$
  - Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$
- Add to ensemble:
  - $F_t(x) = F_{t-1}(x) + \alpha_t h_t(x)$
- Update weights:
  - $w_{i,t+1} = w_{i,t} e^{-y_i \alpha_t h_t(x_i)}$ for all i
  - Renormalize $w_{i,t+1}$ such that $\sum_i w_{i,t+1} = 1$

Final Classifier

This weighted ensemble correctly classifies all points.



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.”

The World of Many Weak Features

Often we have many relatively weak features to apply to a classification problem.

In text classification problems, we often have the frequency of each word in documents of positive and negative classes: e.g. the frequency of “sale’’ in spam and real email.

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.

Identifying the Most Probable Class

Argmax is the class with the highest probability:

$$C(X) = \max_{i=1}^{m} \frac{p(C_i)p(X|C_i)}{p(X)} = \max_{i=1}^{m} p(C_i)p(X|C_i)$$

$p(C_i)$ is the prior probability of class *i*.

$p(X)$ is the prob. of seeing input *X* over all classes. This is dicey, but can be ignored for classification as it is constant (i.e. the same for all classes).

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, but:

What's the probability of having two size 9 feet?

Tabulation Yields Marginal Probabilities

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

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:

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

Is a Sunny-Mild-High a Beach Day?

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

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

Dealing with Zero Counts

You may never have seen it before, but what is the probability my next word is defenestrate?

Observed counts do not accurately capture the frequency of rare events, for which there is typically a long tail.

Laplace asked: “What is the probability the sun will rise tomorrow?”

+1 Discounting

Discounting is a statistical technique to adjust counts for yet-as-unseen events.

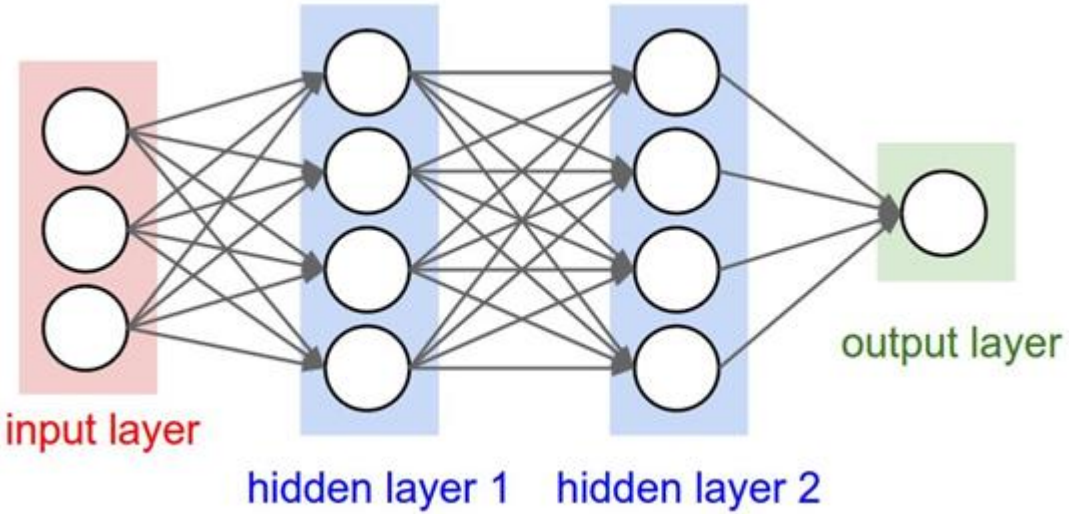
The simplest technique is add one discounting, where we add one to the frequency of all outcomes, including unseen.

Thus after seeing 5 reds and 3 greens,

$$P(newColor) = \frac{1}{(5+1)+(3+1)+(0+1)} = \frac{1}{11}$$

Neural Networks / Deep Learning

The hottest area of machine learning today involves large, deep neural network architectures.



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 & resources

Node Computations

Each node in the network typically computes a nonlinear function Phi(v) of a weighted input sum:

$$v_i = \beta + \sum_i w_i x_i$$

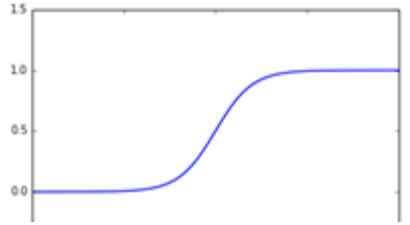
The beta term is the bias, the activation in the absence of input.

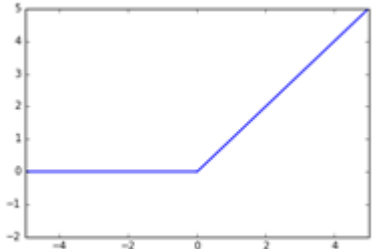
Many dot products implies matrix multiplication!

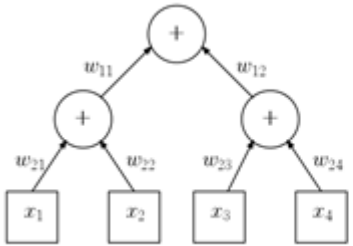
Non-Linearity

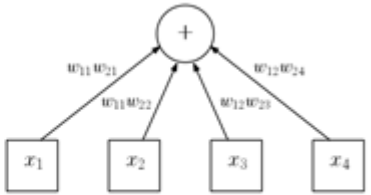
The logistic and RELU functions make good candidates for Phi.

Linear functions like addition cannot exploit depth, because hidden layers add no power.









Backpropagation

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

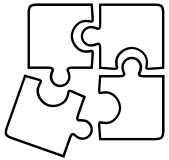
Non-linear functions result in a non-convex optimization function, but this generally produces good results.

Data Cleaning & Feature Engineering

Domain-dependent data cleaning is important:

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

Wrapup: Intro to Machine Learning



- Machine Learning approaches can roughly be categorized into **supervised** and **unsupervised** methods
- Different methods are best for different things – no approach is dominating all others
- Advanced methods are often hard to interpret
- Feature engineering is critically important