Overview of Machine Learning

Partially based on the ML lecture by Raymond J. Mooney

University of Texas at Austin

For full version see ISMU Study Materials FI:PA164 > Learning Materials > Machine learning

Machine learning

- Unsupervised learning (clustering)
- Supervised learning (classification, regression; prediction)
- Association rule learning (learning frequent patterns)
- Outlier analysis

Inductive Classification

Classification (Categorization)

- Given:
  - A description of an instance, $x \in X$, where X is the ***instance language*** or ***instance space***.
  - A fixed set of categories: $C=\{c_1, c_2,\ldots c_n\}$
- Determine:
  - The category of x: $c(x) \in C$, where $c(x)$ is a categorization function whose domain is X and whose range is C.
  - If $c(x)$ is a binary function $C=\{0,1\}$ ({true,false}, {positive, negative}) then it is called a ***concept***.

Learning for Categorization

- A training example is an instance $x \in X$, paired with its correct category $c(x)$: $<x, c(x)>$ for an unknown categorization function, c.
- Given a set of training examples, D.
- Find a hypothesized categorization function, $h(x)$, such that:

$$\forall < x, c(x) > \in D : h(x) = c(x)$$

Consistency

Sample Category Learning Problem

- Instance language: <size, color, shape>
  - size ∈ {small, medium, large}
  - color ∈ {red, blue, green}
  - shape ∈ {square, circle, triangle}
- *C* = {positive, negative}
- *D*:

Example	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negative
4	large	blue	circle	negative

Hypothesis Selection

- Many hypotheses are usually consistent with the training data.
  - red & circle
  - (small & circle) or (large & red)
  - (small & red & circle) or (large & red & circle)

Generalization

- Hypotheses must generalize to correctly classify instances not in the training data.

- Simply memorizing training examples is a consistent hypothesis that does not generalize. But …

Inductive Learning Hypothesis

- Any function that is found to approximate the target concept well on a sufficiently large set of training examples will also approximate the target function well on unobserved examples.
- Assumes that the training and test examples are drawn independently from the same underlying distribution.
- This is a fundamentally unprovable hypothesis unless additional assumptions are made about the target concept and the notion of "approximating the target function well on unobserved examples" is defined appropriately (cf. computational learning theory).

Evaluation of Classification Learning

- Training time (efficiency of training algorithm).
- Complexity of the hypotthesis that has been learned
- Testing time (efficiency of subsequent classification).

- Classification accuracy (% of instances classified correctly).
  - Measured on an independent test data.

Evaluation of Classification Learning

- Confusion matrix
- Precision and recall
- F-measures, here F1-measure only

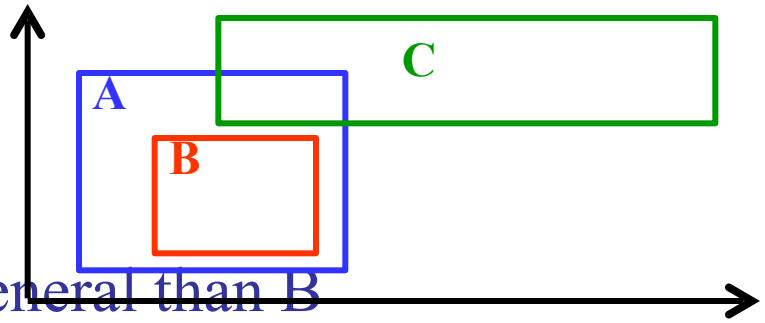
- Learning curve
- ROC curve, AUC

Using the Generality Structure

- By exploiting the structure imposed by the generality of hypotheses, an hypothesis space can be searched for consistent hypotheses without enumerating or explicitly exploring all hypotheses.
- An instance, $x \in X$, is said to ***satisfy*** an hypothesis, h, iff $h(x)=1$ (positive)
- Given two hypotheses h_1 and h_2, h_1 is ***more general than or equal to*** h_2 ($h_1 \geq h_2$) iff every instance that satisfies h_2 also satisfies h_1.
- Given two hypotheses h_1 and h_2, h_1 is **(*strictly*) *more general than*** h_2 ($h_1 > h_2$) iff $h_1 \geq h_2$ and it is not the case that $h_2 \geq h_1$.
- Generality defines a partial order on hypotheses.

Examples of Generality

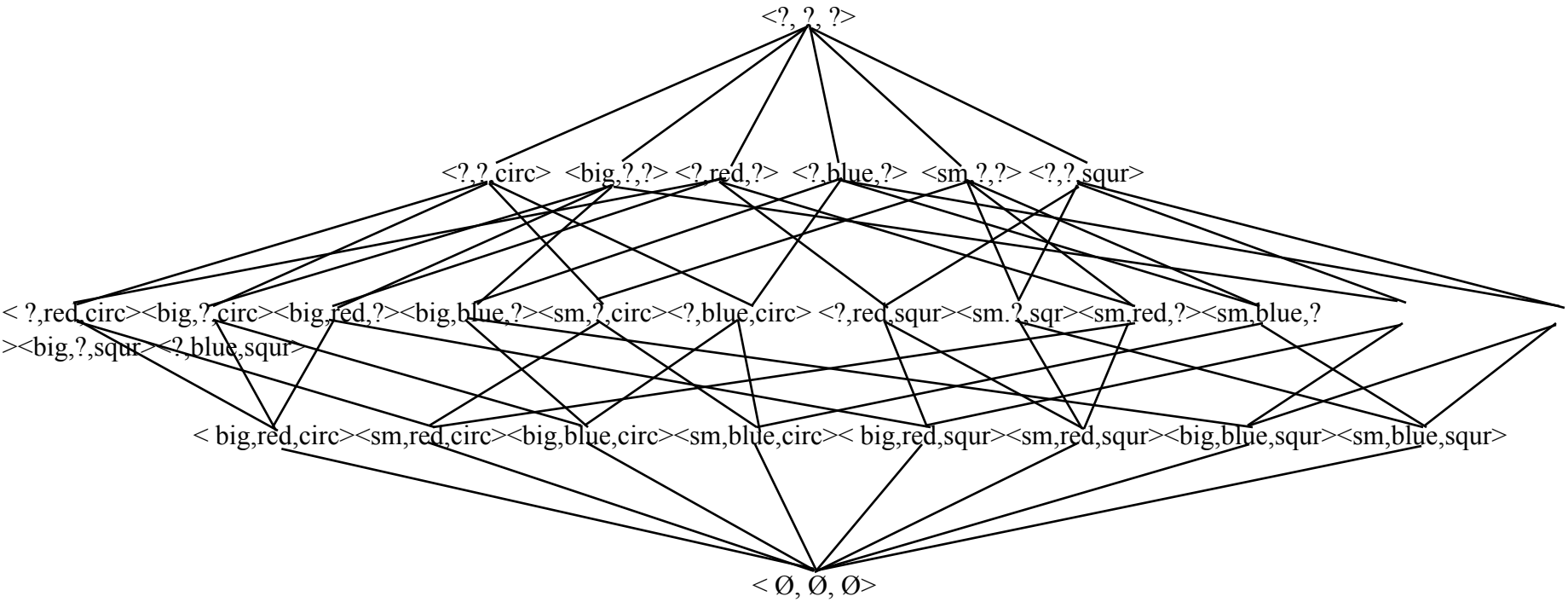
- Conjunctive feature vectors
  - <?, red, ?> is more general than <?, red, circle>
  - Neither of <?, red, ?> and <?, ?, circle> is more general than the other.
- Axis-parallel rectangles in 2-d space



  - A is more general than B
  - Neither of A and C are more general than the other.

Sample Generalization Lattice

Size: {sm, big} Color: {red, blue} Shape: {circ, squr}



Number of hypotheses = $3^3 + 1 = 28$

No Panacea

- No Free Lunch (NFL) Theorem (Wolpert, 1995)
  Law of Conservation of Generalization Performance (Schaffer, 1994)
  - One can prove that improving generalization performance on unseen data for some tasks will always decrease performance on other tasks (which require different labels on the unseen instances).
  - Averaged across all possible target functions, no learner generalizes to unseen data any better than any other learner.

- There does not exist a learning method that is uniformly better than another for all problems.

- Given any two learning methods *A* and *B* and a training set, *D*, there always exists a target function for which *A* generalizes better (or at least as well) as *B*.

Logical View of Induction

- Deduction is inferring sound specific conclusions from general rules (axioms) and specific facts.
- Induction is inferring general rules and theories from specific empirical data.
- Induction can be viewed as inverse deduction.
  - Find a hypothesis h from data D such that
    - $h \cup B \mid\!\!- D$

      where B is optional background knowledge
- ***Abduction*** is similar to induction, except it involves finding a specific hypothesis, h, that best *explains* a set of evidence, D, or inferring cause from effect. Typically, in this case B is quite large compared to induction and h is smaller and more specific to a particular event.

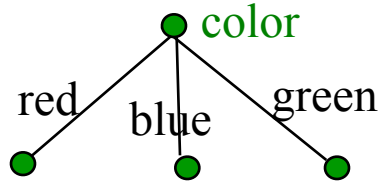
Decision Tree Learning

Top-Down Decision Tree Induction

- Recursively build a tree top-down by divide and conquer.

<big, red, circle>: +
<small, red, square>: –

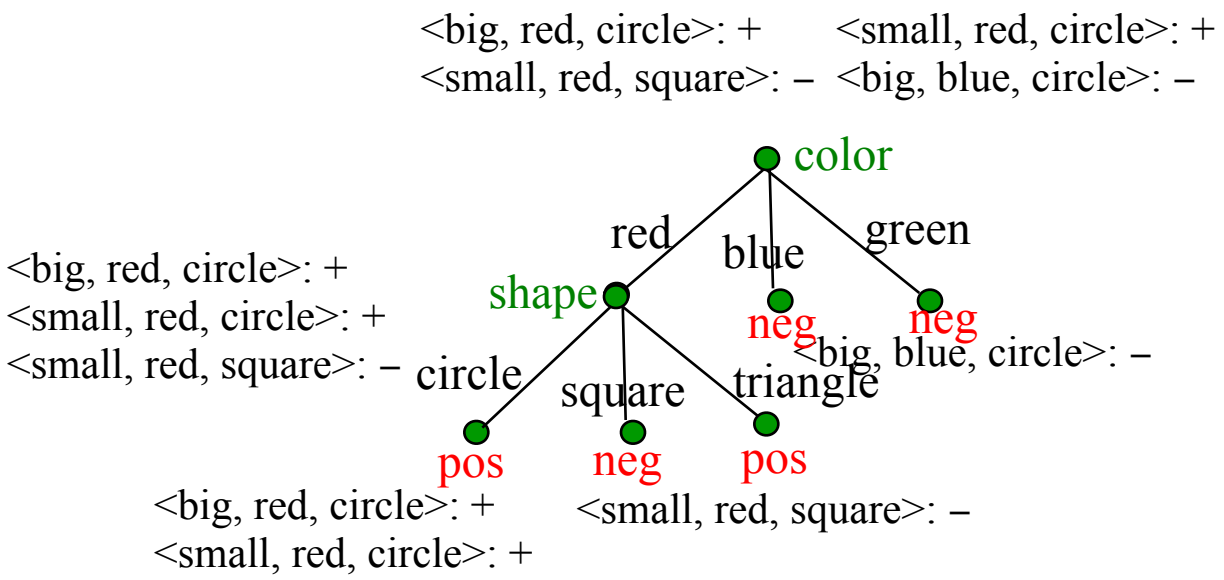
<small, red, circle>: +
<big, blue, circle>: –



<big, red, circle>: +
<small, red, circle>: +
<small, red, square>: –

Top-Down Decision Tree Induction

- Recursively build a tree top-down by divide and conquer.



Properties of Decision Tree Learning

- Continuous (real-valued) features can be handled by allowing nodes to split a real valued feature into two ranges based on a threshold (e.g. length < 3 and length ≥ 3)
- Classification trees have discrete class labels at the leaves, ***regression trees*** allow real-valued outputs at the leaves.
- Algorithms for finding consistent trees are efficient for processing large amounts of training data for data mining tasks.
- Methods developed for handling noisy training data (both class and feature noise).
- Methods developed for handling missing feature values.

Picking a Good Split Feature

- Goal is to have the resulting tree be as small as possible, per Occam's razor.
- Finding a minimal decision tree (nodes, leaves, or depth) is an NP-hard optimization problem.
- Top-down divide-and-conquer method does a greedy search for a simple tree but does not guarantee to find the smallest.
  - General lesson in ML: "Greed is good."
- Want to pick a feature that creates subsets of examples that are relatively "pure" in a single class so they are "closer" to being leaf nodes.
- There are a variety of heuristics for picking a good test, a popular one is based on information gain that originated with the ID3 system of Quinlan (1979).

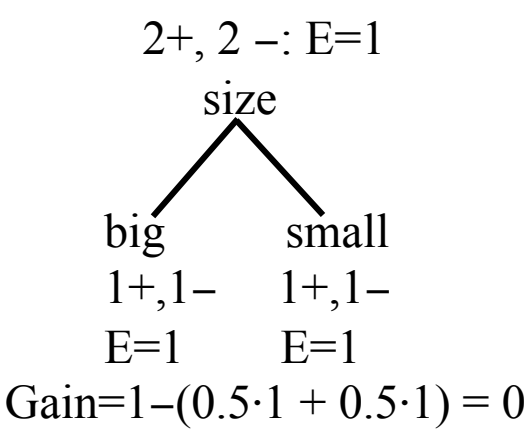
Information Gain

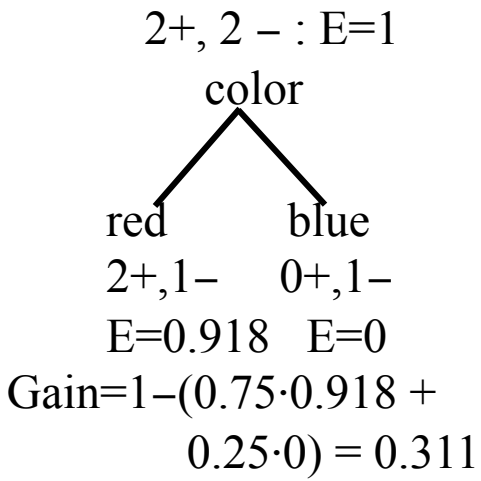
- The information gain of a feature F is the expected reduction in entropy resulting from splitting on this feature.

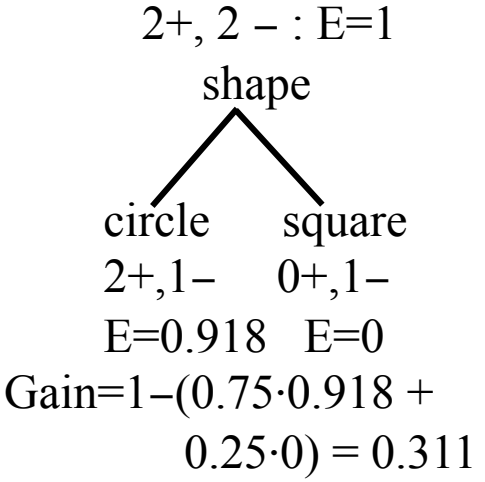
$$Gain(S,F) = Entropy(S) - \sum_{v \in Values(F)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where S_v is the subset of S having value v for feature F.

- Entropy of each resulting subset weighted by its relative size.
- Example:
  - <big, red, circle>: + <small, red, circle>: +
  - <small, red, square>: – <big, blue, circle>: –





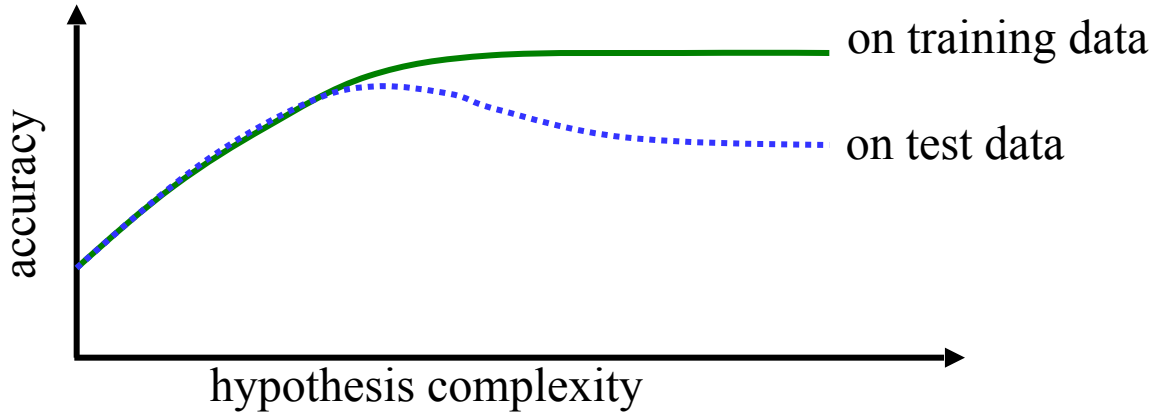


Bias in Decision-Tree Induction

- Information-gain gives a bias for trees with minimal depth.
- Implements a search (preference) bias instead of a language (restriction) bias.

Overfitting

- Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization to unseen data.
  - There may be noise in the training data that the tree is erroneously fitting.
  - The algorithm may be making poor decisions towards the leaves of the tree that are based on very little data and may not reflect reliable trends.
- A hypothesis, h, is said to overfit the training data is there exists another hypothesis which, h', such that h has less error than h' on the training data but greater error on independent test data.

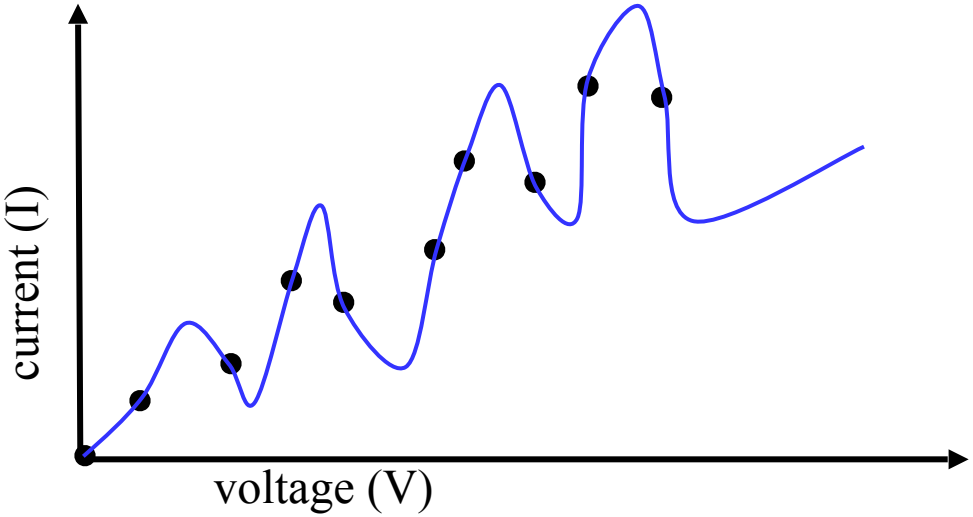


Overfitting Example

Testing Ohms Law: V = IR (I = (1/R)V)

Experimentally measure 10 points

Fit a curve to the Resulting data.

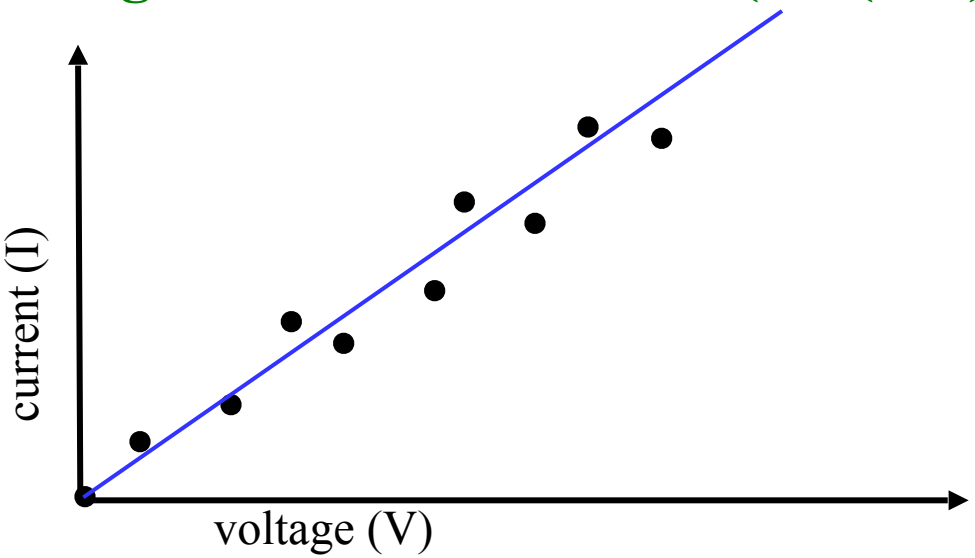


Perfect fit to training data with an 9th degree polynomial
(can fit n points exactly with an n-1 degree polynomial)

Ohm was wrong, we have found a more accurate function!

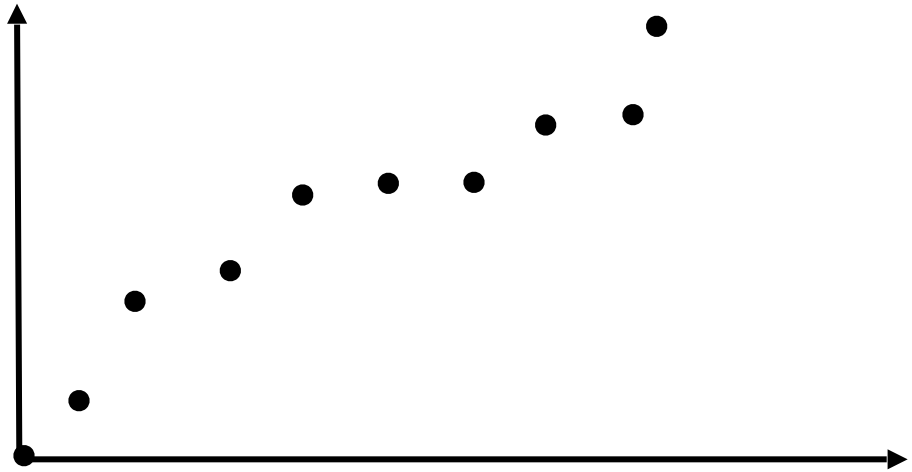
Overfitting Example

Testing Ohms Law: V = IR (I = (1/R)V)



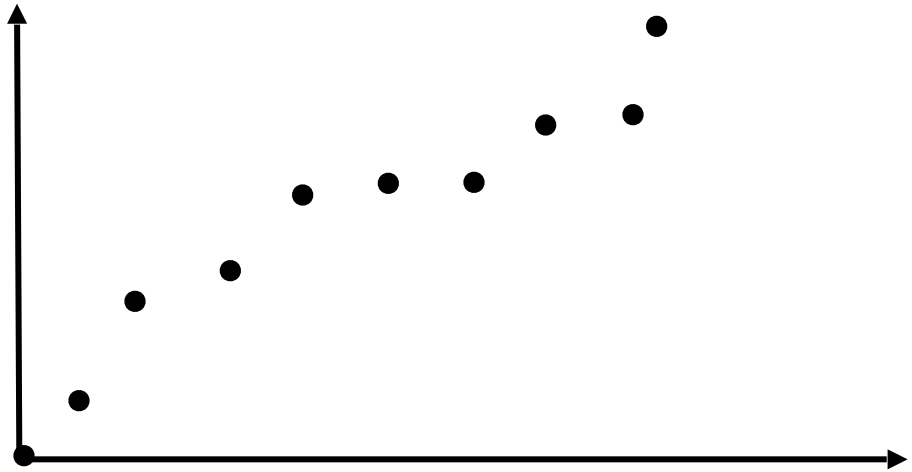
Better generalization with a linear function
that fits training data less accurately.

Bias-variance tradeoff



Another example

Bias-variance tradeoff

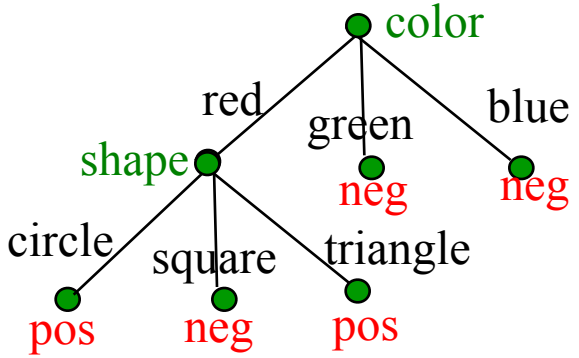


Linear function works well but
Cubic seems better

Is there any general view to the problem,
preferably with a theoretical background?

Overfitting Noise in Decision Trees

- Category or feature noise can easily cause overfitting.
  - Add noisy instance <medium, blue, circle>: pos (but really neg)



Overfitting Prevention (Pruning) Methods

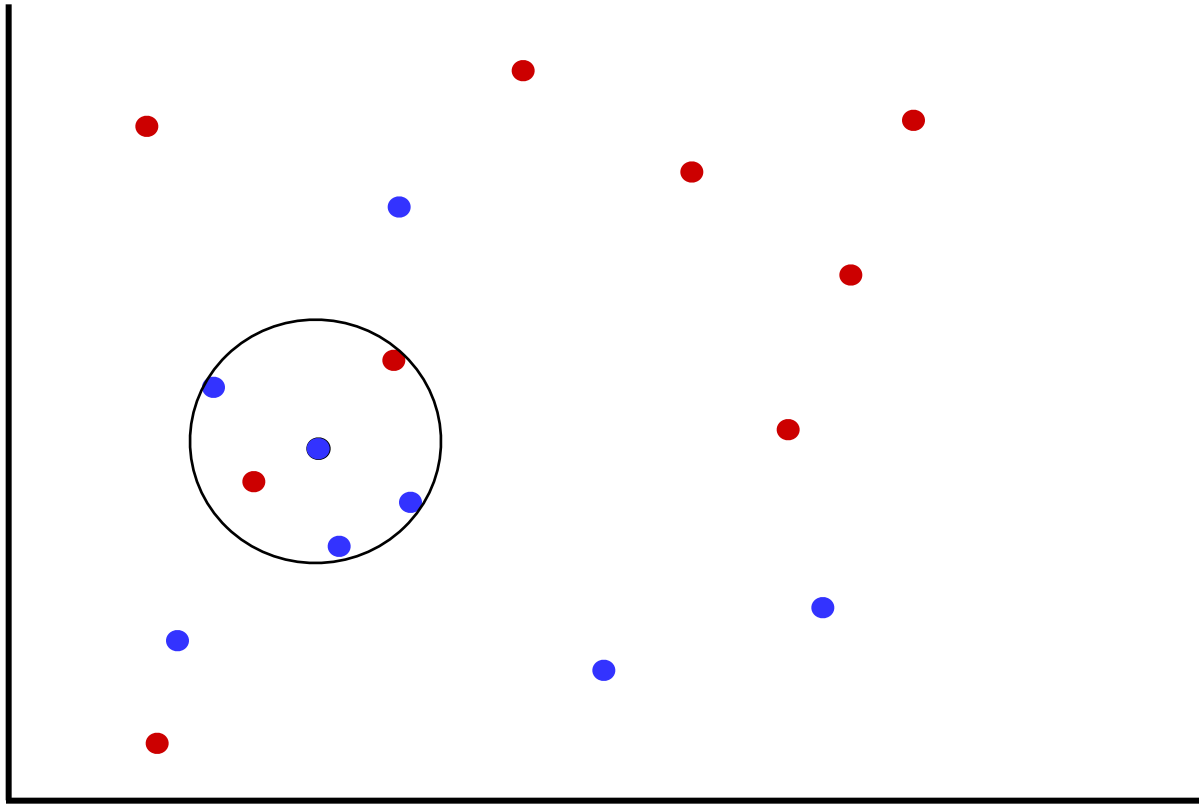
- Two basic approaches for decision trees
  - Prepruning: Stop growing tree as some point during top-down construction when there is no longer sufficient data to make reliable decisions.
  - Postpruning: Grow the full tree, then remove subtrees that do not have sufficient evidence.
- Label leaf resulting from pruning with the majority class of the remaining data, or a class probability distribution.
- Method for determining which subtrees to prune:
  - Cross-validation: Reserve some training data as a hold-out set (***validation set***, ***tuning set***) to evaluate utility of subtrees.
  - Statistical test: Use a statistical test on the training data to determine if any observed regularity can be dismisses as likely due to random chance.
  - Minimum description length (MDL): Determine if the additional complexity of the hypothesis is less complex than just explicitly remembering any exceptions resulting from pruning.

C4.5

- Based on ID3 algorithm, author Ross Quinlan
- In all (or most of) non-commercial and commercial data mining tools
- Weka Trees -> j48

Instance Based Learning

Example



Instance-Based Learning

- Unlike other learning algorithms, does not involve construction of an explicit abstract generalization but classifies new instances based on direct comparison and similarity to known training instances.
- Training can be very easy, just memorizing training instances.
- Testing can be very expensive, requiring detailed comparison to all past training instances.
- Also known as:
  - Case-based
  - Exemplar-based
  - Nearest Neighbor
  - Memory-based
  - Lazy Learning

Similarity/Distance Metrics

- Instance-based methods assume a function for determining the similarity or distance between any two instances.
- For continuous feature vectors, Euclidian distance is the generic choice:

$$d(x_i, x_j) = \sqrt{\sum_{p=1}^{n} (a_p(x_i) - a_p(x_j))^2}$$

Where $a_p(x)$ is the value of the p th feature of instance x.

- For discrete features, assume distance between two values is 0 if they are the same and 1 if they are different (e.g. Hamming distance for bit vectors).
- To compensate for difference in units across features, scale all continuous values to the interval [0,1].

Other Distance Metrics

- Mahalanobis distance (→)
  - Scale-invariant metric that normalizes for variance.
- Cosine Similarity
  - Cosine of the angle between the two vectors.
  - Used in text and other high-dimensional data.
- Pearson correlation (→)
  - Standard statistical correlation coefficient.
- Edit distance
  - Used to measure distance between unbounded length strings.

K-Nearest Neighbor

- Calculate the distance between a test point and every training instance.
- Pick the *k* closest training examples and assign the test instance to the most common category amongst these nearest neighbors.
- Voting multiple neighbors helps decrease susceptibility to noise.
- Usually use odd value for *k* to avoid ties.

Nearest Neighbor Variations

- Can be used to estimate the value of a real-valued function – regression - by taking the average function value of the *k* nearest neighbors to an input point.
- All training examples can be used to help classify a test instance by giving every training example a vote that is weighted by the inverse square of its distance from the test instance.

Feature Relevance and Weighting

- Standard distance metrics weight each feature equally when determining similarity.
  - Problematic if many features are irrelevant, since similarity along many irrelevant examples could mislead the classification.
- Features can be weighted by some measure that indicates their ability to discriminate the category of an example, such as information gain.
- Overall, instance-based methods favor global similarity over concept simplicity.

Naïve Bayes Learning

Based on Raymond J. Mooney's slides

University of Texas at Austin

Bayes Theorem

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

Simple proof from definition of conditional probability:

$$P(H \mid E) = \frac{P(H \wedge E)}{P(E)} \quad \text{(Def. cond. prob.)}$$

$$P(E \mid H) = \frac{P(H \wedge E)}{P(H)} \quad \text{(Def. cond. prob.)}$$

$$P(H \wedge E) = P(E \mid H)P(H)$$

QED: $P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$

Bayesian Categorization

- Determine category of x_k by determining for each y_i

$$P(Y = y_i \mid X = x_k) = \frac{P(Y = y_i)P(X = x_k \mid Y = y_i)}{P(X = x_k)}$$

- P(X=x_k) can be determined since categories are complete and disjoint.

$$\sum_{i=1}^{m} P(Y = y_i \mid X = x_k) = \sum_{i=1}^{m} \frac{P(Y = y_i)P(X = x_k \mid Y = y_i)}{P(X = x_k)} = 1$$

$$P(X = x_k) = \sum_{i=1}^{m} P(Y = y_i)P(X = x_k \mid Y = y_i)$$

Bayesian Categorization (cont.)

- Need to know:
  - Priors: $P(Y=y_i)$
  - Conditionals: $P(X=x_k \mid Y=y_i)$
- $P(Y=y_i)$ are easily estimated from data.
  - If n_i of the examples in D are in y_i then $P(Y=y_i) = n_i / |D|$
- Too many possible instances (e.g. 2^n for binary features) to estimate all $P(X=x_k \mid Y=y_i)$.
- Still need to make some sort of independence assumptions about the features to make learning tractable.

Naïve Bayesian Categorization

- If we assume features of an instance are independent **given the category** (***conditionally independent***).

$$P(X \mid Y) = P(X_1, X_2, \boxed{?} X_n \mid Y) = \prod_{i=1}^{n} P(X_i \mid Y)$$

- Therefore, we then only need to know P(X_i | Y) for each possible pair of a feature-value and a category.
- If Y and all X_i and binary, this requires specifying only $2n$ parameters:
  - P(X_i=true | Y=true) and P(X_i=true | Y=false) for each X_i
  - P(X_i=false | Y) = 1 – P(X_i=true | Y)

- Compared to specifying 2^n parameters without any independence assumptions.

Naïve Bayes Example

Probability	positive	negative
P(Y)	0.5	0.5
P(small | Y)	0.4	0.4
P(medium | Y)	0.1	0.2
P(large | Y)	0.5	0.4
P(red | Y)	0.9	0.3
P(blue | Y)	0.05	0.3
P(green | Y)	0.05	0.4
P(square | Y)	0.05	0.4
P(triangle | Y)	0.05	0.3
P(circle | Y)	0.9	0.3

Test Instance:
<medium ,red, circle>

Naïve Bayes Example

Probability	positive	negative
P(Y)	0.5	0.5
P(medium | Y)	0.1	0.2
P(red | Y)	0.9	0.3
P(circle | Y)	0.9	0.3

Test Instance:
<medium ,red, circle>

$$P(\text{positive} \mid X) = P(\text{positive}) * P(\text{medium} \mid \text{positive}) * P(\text{red} \mid \text{positive}) * P(\text{circle} \mid \text{positive}) / P(X)$$
$$0.5 * 0.1 * 0.9 * 0.9$$
$$= 0.0405 / P(X) = 0.0405 / 0.0495 = 0.8181$$

$$P(\text{negative} \mid X) = P(\text{negative}) * P(\text{medium} \mid \text{negative}) * P(\text{red} \mid \text{negative}) * P(\text{circle} \mid \text{negative}) / P(X)$$
$$0.5 * 0.2 * 0.3 * 0.3$$
$$= 0.009 / P(X) = 0.009 / 0.0495 = 0.1818$$

$$P(\text{positive} \mid X) + P(\text{negative} \mid X) = 0.0405 / P(X) + 0.009 / P(X) = 1$$

$$P(X) = (0.0405 + 0.009) = 0.0495$$

Estimating Probabilities

- Normally, probabilities are estimated based on observed frequencies in the training data.
- If D contains n_k examples in category y_k, and n_{ijk} of these n_k examples have the jth value for feature X_i, x_{ij}, then:

$$P(X_i = x_{ij} \mid Y = y_k) = \frac{n_{ijk}}{n_k}$$

- However, estimating such probabilities from small training sets is error-prone.
- If due only to chance, a rare feature, X_i, is always false in the training data, $\forall y_k : P(X_i = \text{true} \mid Y = y_k) = 0$.
- If X_i=true then occurs in a test example, X, the result is that $\forall y_k: P(X \mid Y = y_k) = 0$ and $\forall y_k: P(Y = y_k \mid X) = 0$

Probability Estimation Example

Ex	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negitive
4	large	blue	circle	negitive

Probability	positive	negative
P(Y)	0.5	0.5
P(small | Y)	0.5	0.5
P(medium | Y)	0.0	0.0
P(large | Y)	0.5	0.5
P(red | Y)	1.0	0.5
P(blue | Y)	0.0	0.5
P(green | Y)	0.0	0.0
P(square | Y)	0.0	0.0
P(triangle | Y)	0.0	0.5
P(circle | Y)	1.0	0.5

Test Instance X:
<medium, red, circle>

P(positive | X) = 0.5 * 0.0 * 1.0 * 1.0 / P(X) = 0

P(negative | X) = 0.5 * 0.0 * 0.5 * 0.5 / P(X) = 0

Smoothing

- To account for estimation from small samples, probability estimates are adjusted or *smoothed*.
- Laplace smoothing using an *m*-estimate assumes that each feature is given a prior probability, *p*, that is assumed to have been previously observed in a "virtual" sample of size *m*.

$$P(X_i = x_{ij} \mid Y = y_k) = \frac{n_{ijk} + mp}{n_k + m}$$

- For binary features, *p* is simply assumed to be 0.5.

Laplace Smothing Example

- Assume training set contains 10 positive examples:
  - 4: small
  - 0: medium
  - 6: large
- Estimate parameters as follows (if m=1, p=1/3)
  - P(small | positive) = (4 + 1/3) / (10 + 1) = 0.394
  - P(medium | positive) = (0 + 1/3) / (10 + 1) = 0.03
  - P(large | positive) = (6 + 1/3) / (10 + 1) = 0.576
  - P(small or medium or large | positive) = 1.0

Continuous Attributes

- If X_i is a continuous feature rather than a discrete one, need another way to calculate $P(X_i \mid Y)$.
- Assume that X_i has a Gaussian distribution whose mean and variance depends on Y.
- During training, for each combination of a continuous feature X_i and a class value for Y, y_k, estimate a mean, μ_{ik} , and standard deviation σ_{ik} based on the values of feature X_i in class y_k in the training data.
- During testing, estimate $P(X_i \mid Y=y_k)$ for a given example, using the Gaussian distribution defined by μ_{ik} and σ_{ik}.

$$P(X_i \mid Y = y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} \exp\left(\frac{-(X_i - \mu_{ik})^2}{2\sigma_{ik}^2}\right)$$

Comments on Naïve Bayes

- Tends to work well despite strong assumption of conditional independence.
- Experiments show it to be quite competitive with other classification methods on standard UCI datasets.
- Although it does not produce accurate probability estimates when its independence assumptions are violated, it may still pick the correct maximum-probability class in many cases.
  - Able to learn conjunctive concepts in any case
- Does not perform any search of the hypothesis space. Directly constructs a hypothesis from parameter estimates that are easily calculated from the training data.
  - Strong bias
- Not guarantee consistency with training data.
- Typically handles noise well since it does not even focus on completely fitting the training data.

… and many others

- Support Vector Machines (SVM) and more

- is welcome to be familiar with a ML tool like R or scikit-learn (python)

Subsymbolic learning. Neural nets

Deep Learning Revolution

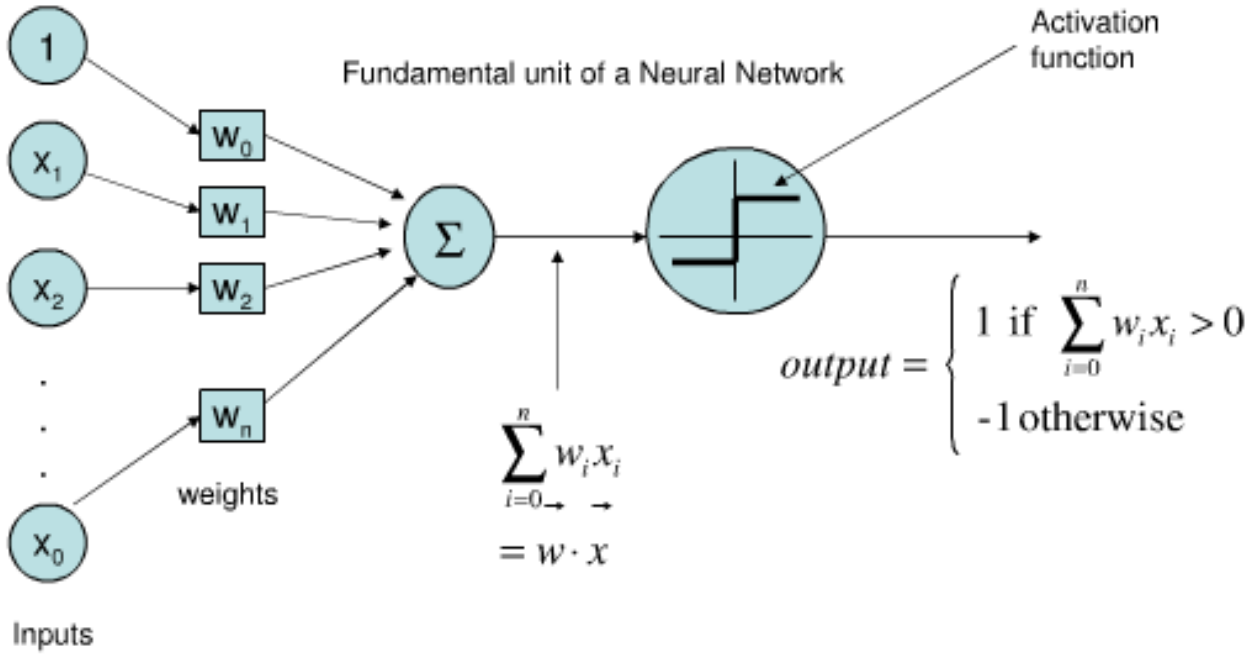
- Recent machine learning methods for training "deep" neural networks (NNs) have demonstrated remarkable progress on many challenging AI problems (e.g. speech recognition, visual object recognition, machine translation, game playing).
- However, their capabilities are prone to "hype."
- Deep learning has not "solved" AI and current methods have clear limitations.

Very Brief History of Machine Learning

- Single-layer neural networks (1957-1969)
- Symbolic AI & knowledge engineering (1970-1985)
- Multi-layer NNs and symbolic learning (1985-1995)
- Statistical (Bayesian) learning and kernel methods (1995-2010)
- Deep learning (CNNs and RNNs) (2010-?)

Single-Layer Neural Network
(Linear Threshold Unit)

- Mathematical model of an individual neuron.



Perceptron

- Rosenblatt (1957) developed an iterative, hill-climbing algorithm for learning the weights of single-layer NN to try to fit a set of training examples.
- Unable to learn or represent many classification functions (e.g. XOR), only the “linearly separable” ones are learnable.

Perceptron Learning Rule

- Update weights by:

$$w_i = w_i + \eta (t - o) x_i$$

where η is the "learning rate," t is the teacher output, and o is the network output.

- Equivalent to rules:
  - If output is correct do nothing.
  - If output is high, lower weights on active inputs
  - If output is low, increase weights on active inputs

Perceptron Learning Algorithm

- Iteratively update weights until convergence.

Initialize weights to random values
Until outputs of all training examples are correct
    For each training pair, *E*, do:
        Compute current output *o* for *E* given its inputs
        Compare current output to target value, t , for *E*
        Update weights using learning rule

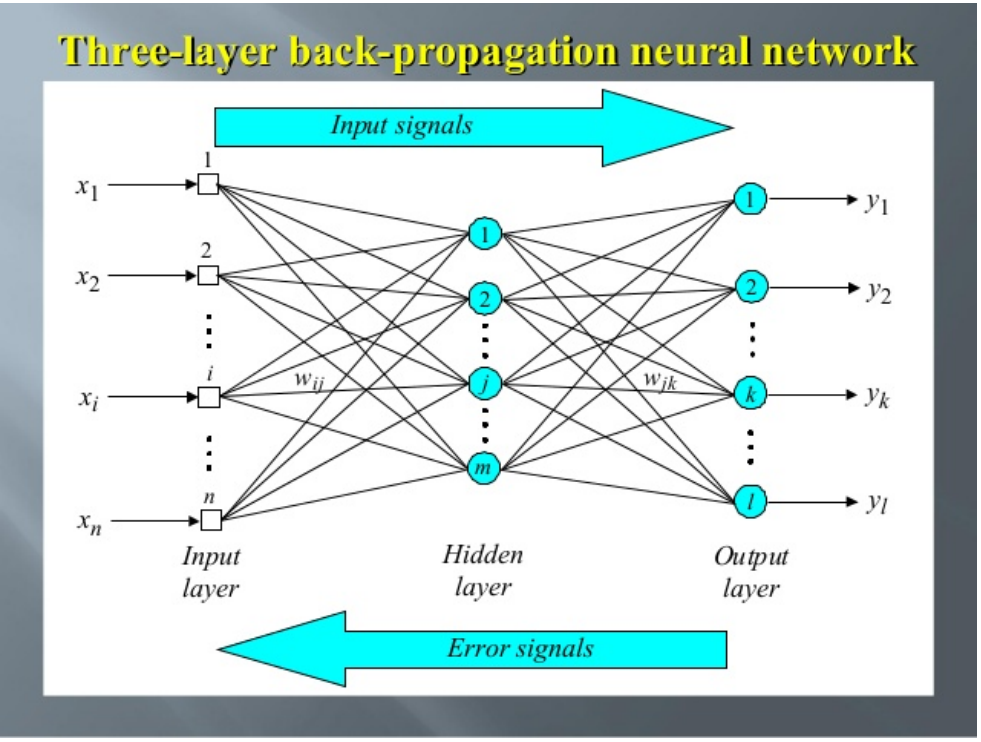
Perceptron Demise

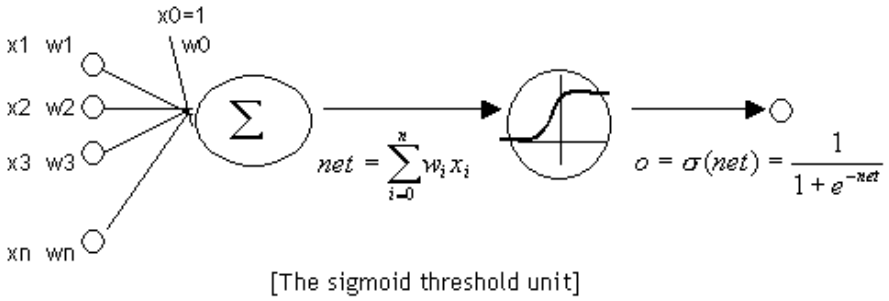
- *Perceptons* (1969) by Minsky and Papert illuminated the limitations of the perceptron.
- Work on neural-networks dissipated during the 70's and early 80's.

Neural Net Resurgence (1986)

- Interest in NNs revived in the mid 1980's due to the rise of "connectionism."
- Backpropagation algorithm popularized for training three-layer NN's.
- Generalized the iterative "hill climbing" method to approximate fitting two layers of synaptic connections, but no convergence guarantees.

3-Layer NN Backpropagation





[The sigmoid threshold unit]

Second NN Demise (1995-2010)

- Generic backpropagation did not generalize that well to training deeper networks.
- Little theoretical justification for underlying methods.
- Machine learning research moved to graphical models and kernel methods.

Deep Learning Revolution (2010…)

- Improved methods developed for training deep neural works.
- Particular successes with:
  - Convolutional neural nets (CNNs) for vision.
  - Recurrent neural nets (RNNs) for machine translation and speech recognition.
  - Deep reinforcement learning for game playing.

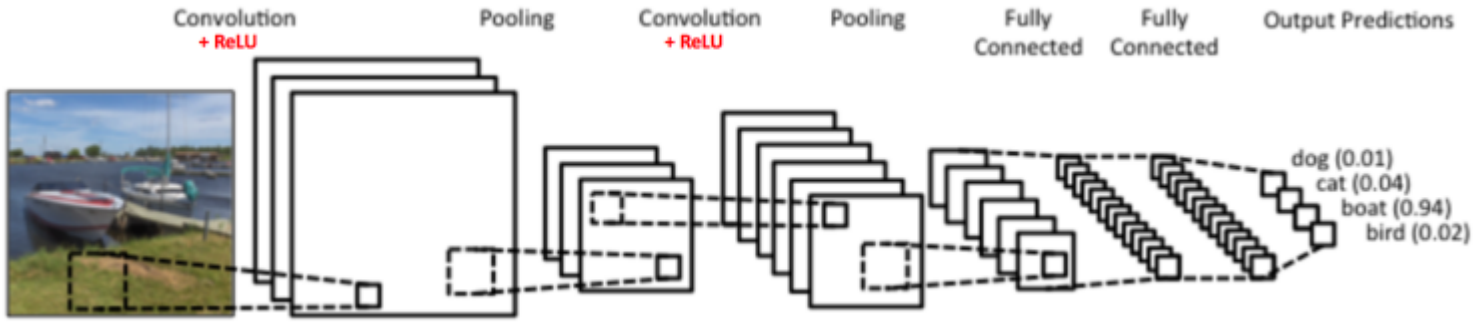
Massive Data and Specialized Hardware

- Large collections of supervised (crowdsourced) training data has been critical.
- Efficient processing of this big data using specialized hardware (Graphics Processing Units, GPUs) has been critical.

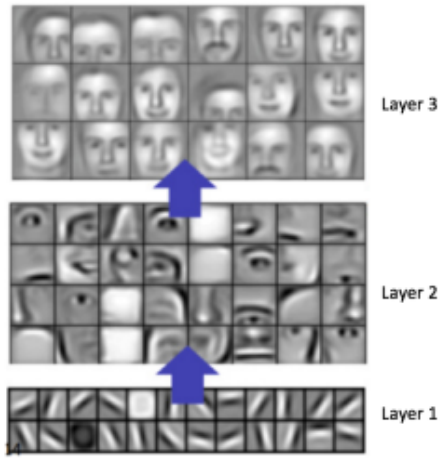
CNNs

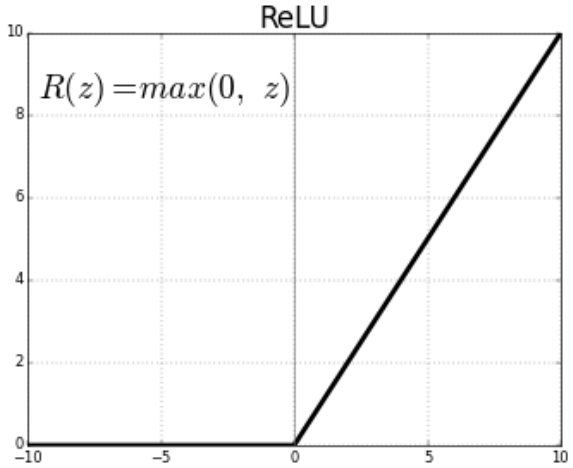
- Convolutional layers learn to extract local features from image regions (receptive fields) analogous to human vision (LeCun, et al., 1998).
- Deeper layers extract higher-level features.
- Pool activity of multiple neurons into one at the next layer using max or mean.
- Nonlinear processing with Rectified Linear Units (ReLUs)
- Decision made using final fully connected layers.

CNNs



Increasingly broader local features extracted from image regions





ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- Recognize 1,000 categories of objects in 150K test images (given 1.2M training images).

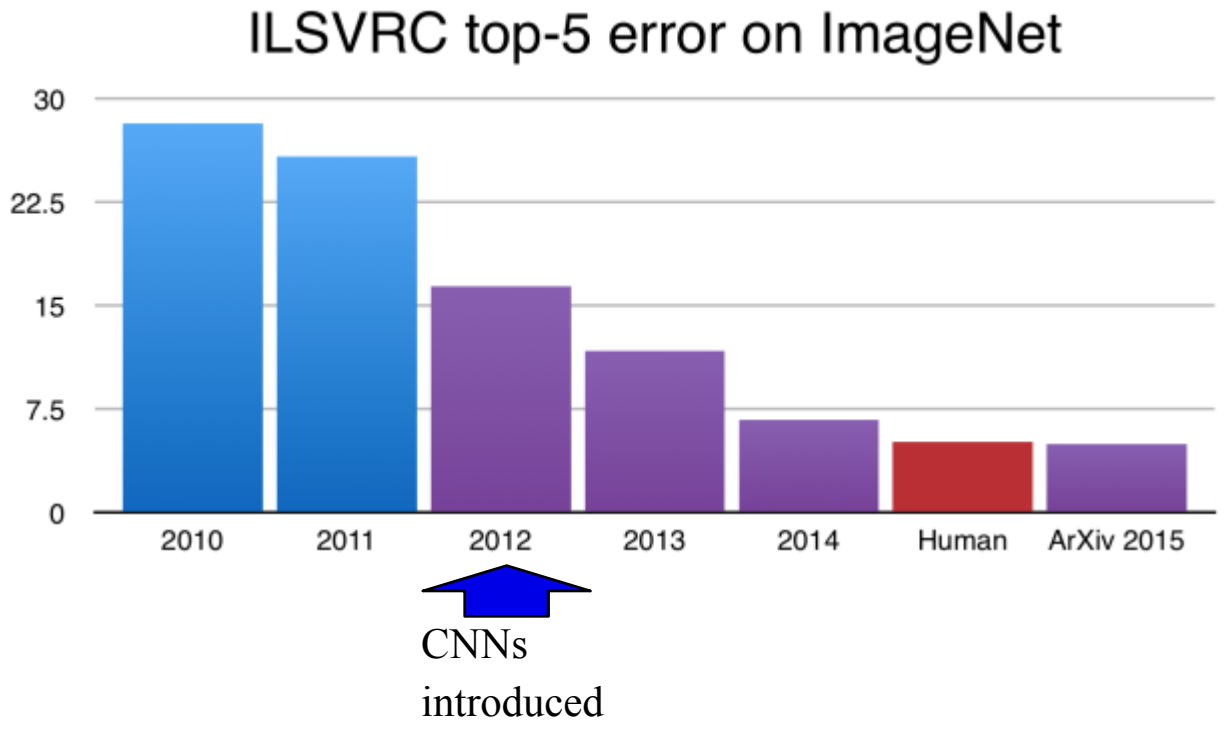
Mongoose

Canoe

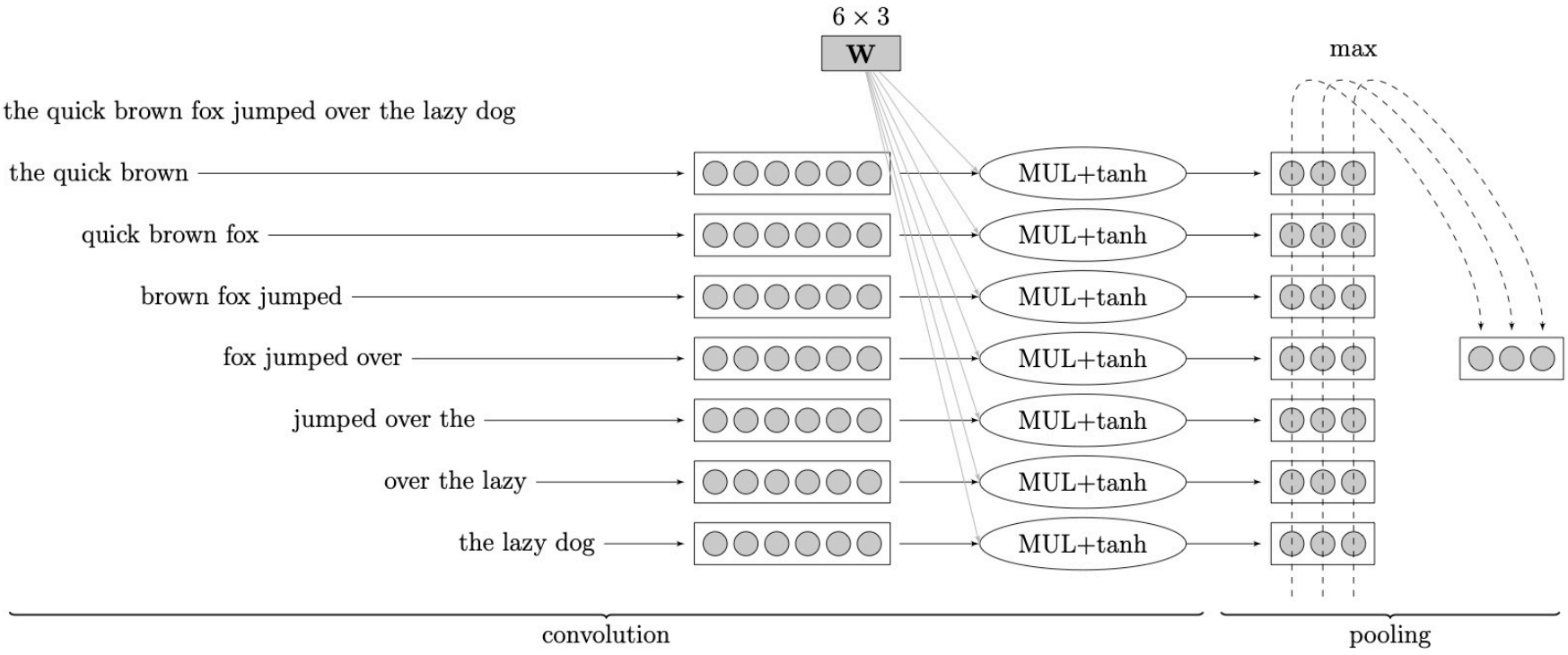
Missile

Trombone

ImageNet Performance Over Time



CNN for text


6 × 3
W
max
the quick brown fox jumped over the lazy dog
the quick brown
quick brown fox
brown fox jumped
fox jumped over
jumped over the
over the lazy
the lazy dog
MUL+tanh
MUL+tanh
MUL+tanh
MUL+tanh
MUL+tanh
MUL+tanh
MUL+tanh
convolution
pooling

Convolutional networks in NLP

Collobert, Weston et al. (2011) semantic-role labeling

Kalchbrenner et al (2014) sentiment classification

Kim (Kim, 2014) question-type classification