Lecture 10
Supervised Learning
Decision Trees and Linear Models

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Slides by Stuart Russell and Peter Norvig

Course Overview

- ✔ Introduction
  - ✔ Artificial Intelligence
  - ✔ Intelligent Agents
- ✔ Search
  - ✔ Uninformed Search
  - ✔ Heuristic Search
- ✔ Uncertain knowledge and Reasoning
  - ✔ Probability and Bayesian approach
  - ✔ Bayesian Networks
  - ✔ Hidden Markov Chains
  - ✔ Kalman Filters

- Learning
  - Supervised
    Decision Trees, Neural Networks
    Learning Bayesian Networks
  - Unsupervised
    EM Algorithm
- Reinforcement Learning
- Games and Adversarial Search
  - Minimax search and Alpha-beta pruning
  - Multiagent search
- Knowledge representation and Reasoning
  - Propositional logic
  - First order logic
  - Inference
  - Plannning

Machine Learning

What?	Parameters, network structure, hidden concepts,
What from?	inductive + unsupervised, reinforcement, supervised
What for?	prediction, diagnosis, summarization
How?	passive vs active, online vs offline
Type of outputs	regression, classification
Details	generative, discriminative

Supervised Learning

Given a training set of N example input-output pairs

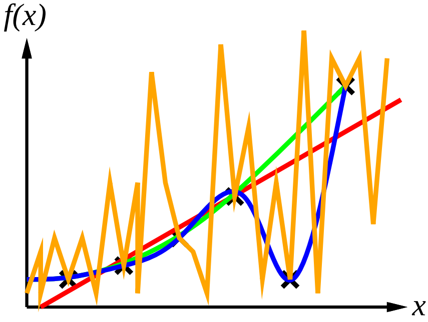
$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

where each y_1 was generated by an unknwon function $y = f(x)$,
find a hypothesis function h from an hypothesis space $\mathcal{H}$ that approximates the true function f

Measure the accuracy of the hypotheis on a test set made of new examples.
We aim a good generalization

Supervised Learning

Construct/adjust h to agree with f on training set
(h is consistent if it agrees with f on all examples)
E.g., curve fitting:



Ockham's razor: maximize a combination of consistency and simplicity if we have a probability on the hypothesis:

$$h^* = \text{argmax}_{h \in \mathcal{H}} \Pr(h \mid data) = \text{argmax}_{h\mathcal{H}} \Pr(data \mid h) \Pr(h)$$

Trade off between the expressiveness of a hypothesis space and the complexity of finding a good hypothesis within that space.

Outline

1. Decision Trees
2. k-Nearest Neighbor
3. Linear Models

Learning Decision Trees

A decision tree of a pair $(\mathbf{x}, y)$ represents a function that takes the input attribute $\mathbf{x}$ (Boolean, discrete, continuous) and outputs a simple Boolean y.

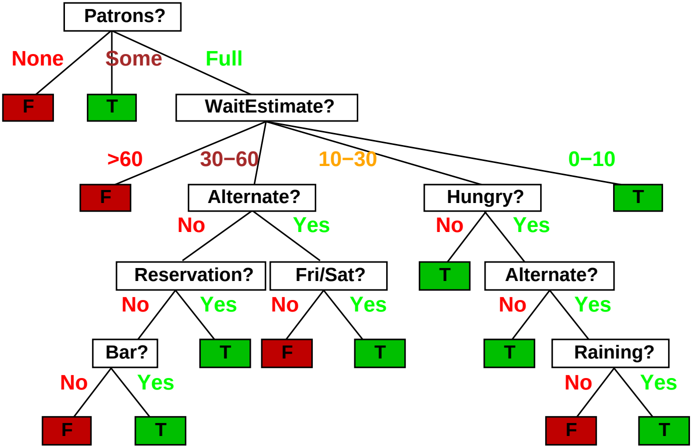
E.g., situations where I will/won't wait for a table. Training set:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Classification of examples positive (T) or negative (F)

Decision trees

One possible representation for hypotheses
E.g., here is the “true” tree for deciding whether to wait:



Example

NO.	RISK	CREDIT HISTORY	DEBT	COLLATERAL	INCOME
1.	high	bad	high	none	\$0 to \$15k
2.	high	unknown	high	none	\$15 to \$35k
3.	moderate	unknown	low	none	\$15 to \$35k
4.	high	unknown	low	none	\$0 to \$15k
5.	low	unknown	low	none	over \$35k
6.	low	unknown	low	adequate	over \$35k
7.	high	bad	low	none	\$0 to \$15k
8.	moderate	bad	low	adequate	over \$35k
9.	low	good	low	none	over \$35k
10.	low	good	high	adequate	over \$35k
11.	high	good	high	none	\$0 to \$15k
12.	moderate	good	high	none	\$15 to \$35k
13.	low	good	high	none	over \$35k
14.	high	bad	high	none	\$15 to \$35k

Table 10.1 Data from credit history of loan applications

Example

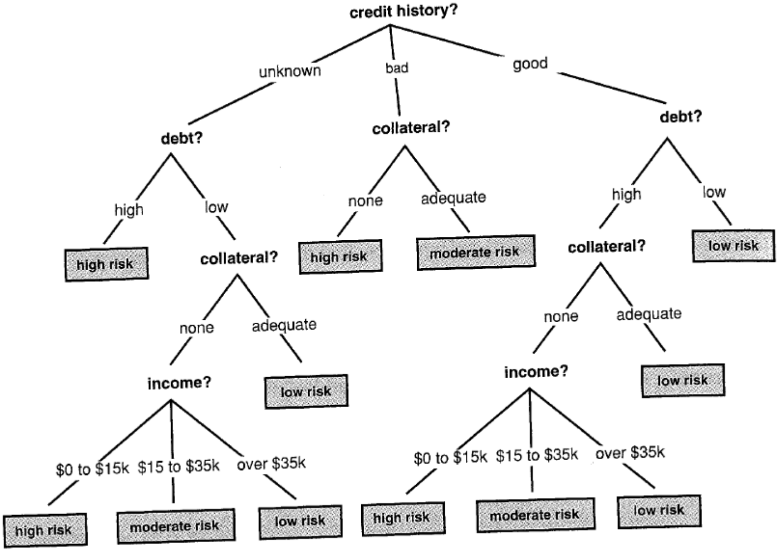
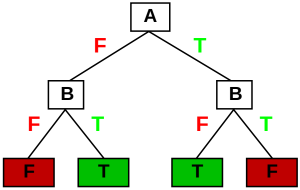


Figure 10.13 A decision tree for credit risk assessment.

Expressiveness

Decision trees can express any function of the input attributes.
E.g., for Boolean functions, truth table row $\rightarrow$ path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Trivially, there is a consistent decision tree for any training set
w/ one path to leaf for each example (unless f nondeterministic in x)
but it probably won't generalize to new examples
Prefer to find more **compact** decision trees

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

= number of Boolean functions

= number of distinct truth tables with 2^n rows = 2^{2^n} functions

E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

More expressive hypothesis space

- increases chance that target function can be expressed
- increases number of hypotheses consistent w/ training set
  - $\Longrightarrow$ may get worse predictions

There is no way to search the smallest consistent tree among 2^{2^n}.

Heuristic approach

Greedy divide-and-conquer:

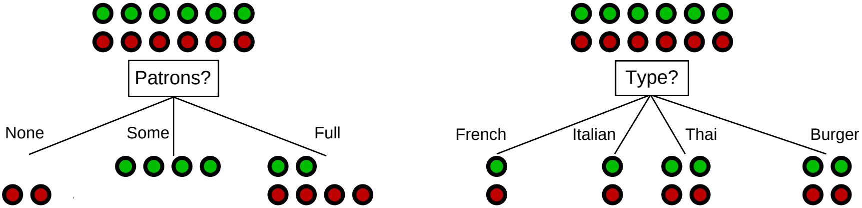
- test the most important attribute first
- divide the problem up into smaller subproblems that can be solved recursively

```
function DTL(examples, attributes, default) returns a decision tree

  if examples is empty then return default
  else if all examples have the same classification then return the classification
  else if attributes is empty then return Plurality_Value(examples)
  else
      best ← Choose-Attribute(attributes, examples)
      tree ← a new decision tree with root test best
      for each value v_i of best do
          examples_i ← {elements of examples with best = v_i}
          subtree ← DTL(examples_i, attributes − best, Mode(examples))
          add a branch to tree with label v_i and subtree subtree
      return tree
```

Choosing an attribute

Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



Patrons? is a better choice—gives **information** about the classification

Information

The more clueless I am about the answer initially, the more information is contained in the answer

0 bits to answer a query on a coin with only head
1 bit to answer query to a Boolean question with prior $\langle 0.5, 0.5\rangle$
2 bits to answer a query on a fair die with 4 faces a query on a coin with 99% probability of returing head brings less information than the query on a fair coin.

Shannon formalized this concept with the concept of entropy.
For a random variable X with values x_k and probability $\Pr(x_k)$ has entropy:

$$H(X) = -\sum_k \Pr(x_k) \log_2 \Pr(x_k)$$

- Suppose we have p positive and n negative examples is a training set, then the entropy is $H(\langle p/(p+n), n/(p+n)\rangle)$
  E.g., for 12 restaurant examples, $p = n = 6$ so we need 1 bit to classify a new example information of the table
- An attribute A splits the training set E into subsets $E_1, \ldots, E_d$, each of which (we hope) needs less information to complete the classification
- Let E_i have p_i positive and n_i negative examples
  $\rightsquigarrow$ $H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$ bits needed to classify a new example on that branch
  $\rightsquigarrow$ **expected** entropy after branching is

$$Remainder(A) = \sum_i \frac{p_i+n_i}{p+n} H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$$

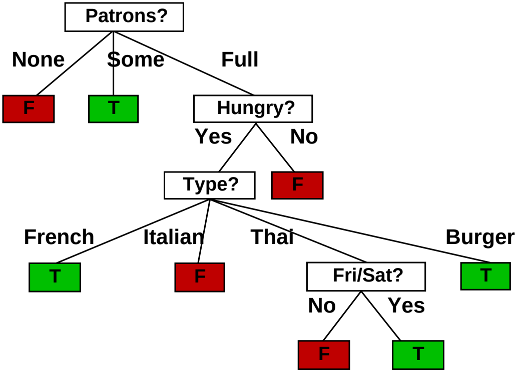
- The **information gain** from attribute A is

$$Gain(A) = H(\langle p/(p+n), n/(p+n)\rangle) - Remainder(A)$$

$\Longrightarrow$ choose the attribute that maximizes the gain

Example contd.

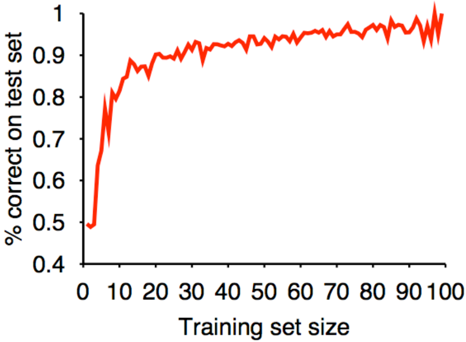
Decision tree learned from the 12 examples:



Substantially simpler than "true" tree—a more complex hypothesis isn't justified by small amount of data

Performance measurement

Learning curve = % correct on test set as a function of training set size



Restaurant data; graph averaged over 20 trials

Overfitting and Pruning

Pruning by statistical testing
under the null hyothesis expected numbers, $\hat{p}_k$ and $\hat{n}_k$:

$$\hat{p}_k = p \cdot \frac{p_k + n_k}{p + n} \quad \hat{n}_k = n \cdot \frac{p_k + n_k}{p + n}$$

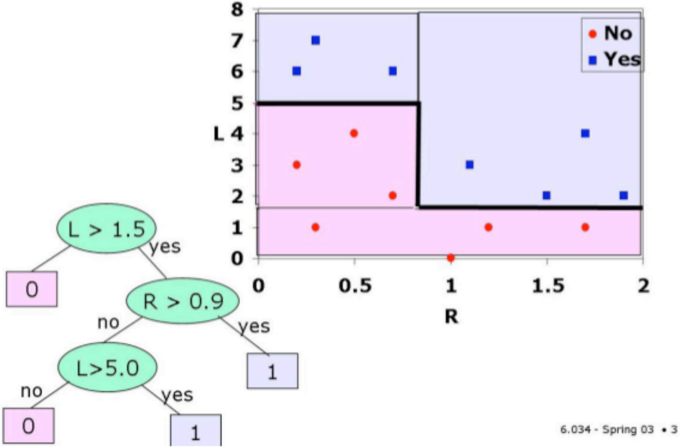
$$\Delta = \sum_{k=1}^{d} \frac{(p_k - \hat{p}_k)2}{\hat{p}_k} + \frac{(n_k - \hat{n}_k)2}{\hat{n}_k}$$

χ^2 distribution with $p + n - 1$ degrees of freedom
Early stopping misses combinations of attributes that are informative.

Further Issues

- Missing data
- Multivalued attributes
- Continuous input attributes
- Continuous-valued output attributes

Decision Trees



Decision Tree Types

- Classification tree analysis is when the predicted outcome is the class to which the data belongs. Iterative Dichotomiser 3 (ID3), C4.5, (Quinlan, 1986)
- Regression tree analysis is when the predicted outcome can be considered a real number (e.g. the price of a house, or a patient's length of stay in a hospital).
- Classification And Regression Tree (CART) analysis is used to refer to both of the above procedures, first introduced by (Breiman et al., 1984)
- CHi-squared Automatic Interaction Detector (CHAID). Performs multi-level splits when computing classification trees. (Kass, G. V. 1980).
- A Random Forest classifier uses a number of decision trees, in order to improve the classification rate.
- Boosting Trees can be used for regression-type and classification-type problems.

Used in data mining (most are included in R, see `rpart` and `party` packages, and in Weka, Waikato Environment for Knowledge Analysis)

Outline

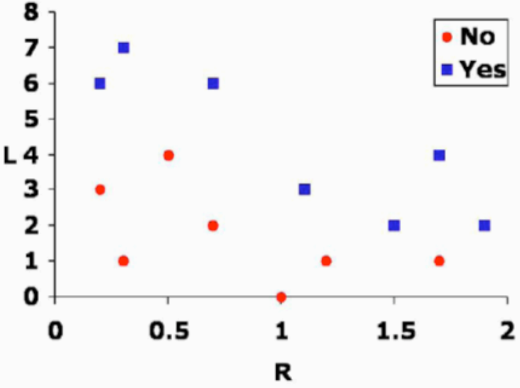
1. Decision Trees
2. k-Nearest Neighbor
3. Linear Models

Non-parametric learning

- When little data available $\rightsquigarrow$ parametric learning (restricted from the model selected)
- When massive data we can let hypothesis grow from data $\rightsquigarrow$ non parametric learning
  instance based: construct from training instances

Predicting Bankruptcy

L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1.0	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes

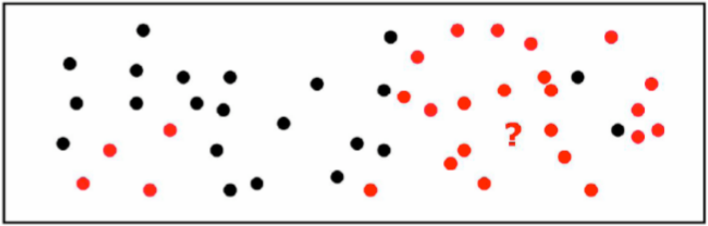


L: #late payments / year
R: expenses / income

Nearest Neighbor

Basic idea:

- Remember all your data
- When someone asks a question
  - find nearest old data point
  - return answer associated with it

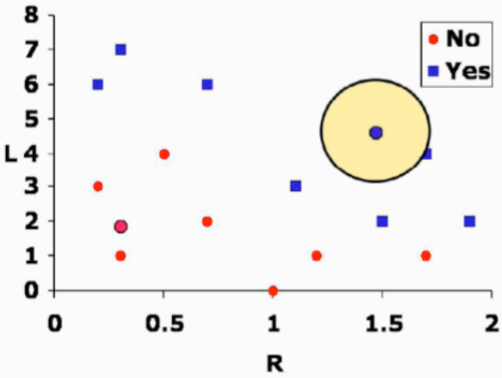


- Find k observations closest to x and average the response

$$\hat{Y} = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

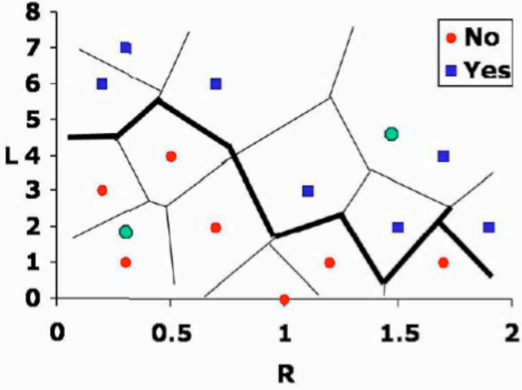
- For qualitative use majority rule
- Needed a distance measure:
  - Euclidean
  - Standardization $x' = \frac{x - \bar{x}}{\sigma_x}$ (Mahalanobis, scale invariant)
  - Hamming

Predicting Bankruptcy



$$D(x^i, x^k) = \sqrt{\sum_j (L^i - L^k)^2 + (5R^i - 5R^k)^2}$$

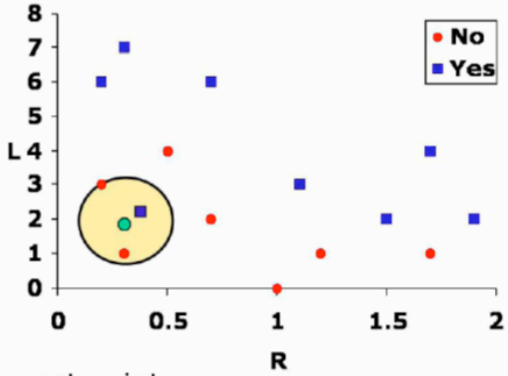
Predicting Bankruptcy



$$D(x^i, x^k) = \sqrt{\sum (L^i - L^k)^2 + (5R^i - 5R^k)^2}$$

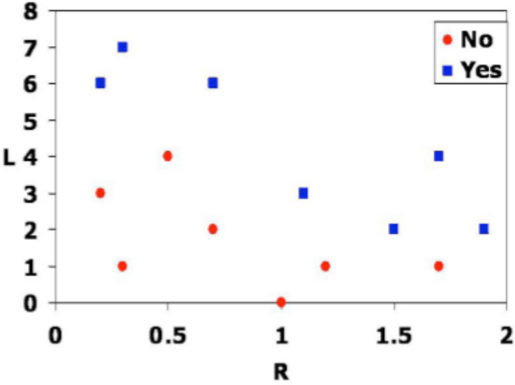
- Learning is fast
- Lookup takes about n computations
  with k-d trees can be faster
- Memory can fill up with all that data
- Problem: Course of dimensionality $b^d = \frac{k}{N}1 \implies b = \frac{k}{N}^{\frac{1}{d}}$

k-Nearest Neighbor

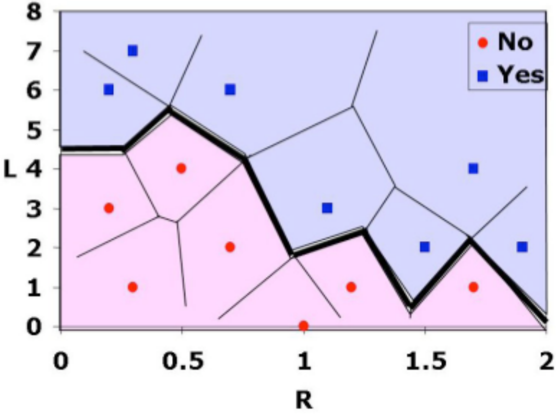


- Find the k nearest points
- Predict output according to the majority
- Choose k using cross-validation

Backruptcy Example



1-Nearest Neighbor



Outline

1. Decision Trees
2. k-Nearest Neighbor
3. Linear Models

Linear Models

Univariate case
Hypotheisis space made by linear functions

$$h_w(x) = w_1 x + w_0$$

Find w by min squared loss function:

$$\mathcal{L}(h_w) = \sum_{j=1}^{N} L_2\left(y_j, h_w(x_j)\right) = \sum_{j=1}^{N} \left(y_j - h_w(x_j)\right)^2$$

$$w^* = \operatorname{argmin} \mathcal{L}(h_w(x))$$

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial w_0} = -2(y - h_w(x)) = 0 \\ \dfrac{\partial \mathcal{L}}{\partial w_1} = -2(y - h_w(x))x = 0 \end{cases}$$

w_0, w_1 in closed form.

Multivariate case

$$h_w(x) = w_0 + w_1 x_1 + \ldots + w_n x_n = w \cdot x$$

$$w^* = \text{argmin}_w \sum_j L_2(y_j, wx_j)$$

$$w^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y \text{ in closed form}$$

- Basis functions: fixed non linear functions $\phi_j(x)$: $h_w(x) = w_0 + \sum_{j=1}^{P} \phi_j(x)$
- To avoid overfitting, regularization: $\mathrm{EmpLoss}(h) + \lambda \cdot \mathrm{Complexity}(h)$

$$\mathrm{Complexity}(h) = L_q(w) = \sum_i |w_i|^q$$

Non-Parametric Regression

Instance based methods

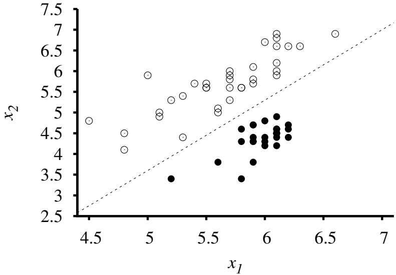
Similar idea as k-nearest neighbor:

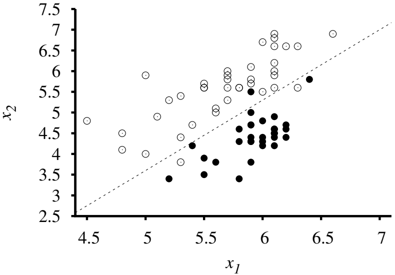
For a query point x_q solve following regression problem:

$$w^* = \text{argmin}_w \sum_j K(||x_q - x_j||)(y_j - w \cdot x_j)^2$$

where K is a kernel function (eg, radial kernel)

Linear Classification



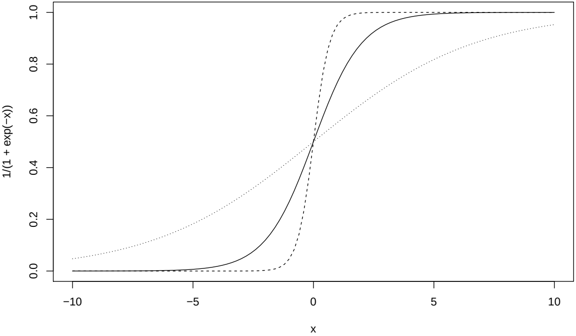


decision boundary described by $ax_1 + bx_2 = 0$

$$h_w(x) = \begin{cases} 1 & \text{if } w \cdot x \geq 0 \\ 0 & \textit{otherwise} \end{cases}$$

step function: gradient not defined

Logistic Regression



$$h_w(x) = \frac{1}{1 + \exp(-w \cdot x)}$$

$$g'(z) = g(z)(1 - g(z))$$

$$g'(w \cdot x) = g(w \cdot x)(1 - g(w \cdot x) = h_w(x)(1 - h_w(x))$$

$$\frac{\partial \mathcal{L}}{\partial w_i} = -2(y - h_w(x)) \cdot g'(w \cdot x)x_i$$

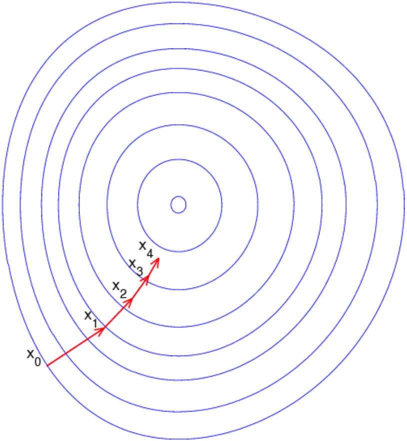
Gradient Descent

Finding local minima of derivable continuous functions

$w \leftarrow$ any initial value
repeat
  for each w_i in w **do**
    $w_i \leftarrow w_i - \alpha \frac{\partial \mathcal{L}}{\partial w_i}$
until convergence ;

Batch gradient descent: $\mathcal{L}$ is the sum of the contribution of each example. Guaranteed to converge.

Stochastic gradient descent: one example at a time in random order. Online. Not guaranteed to converge.



Gradient Descent for Step Function

In step function gradient not defined.
However, the update rule:

$$w_i \leftarrow w_i - \alpha(y - hw(x))x_i$$

ensures convergence when data are linearly separable. Otherwise unsure.