T-61.3050 Machine Learning: Basic Principles

Linear Discrimination

Kai Puolamäki

Laboratory of Computer and Information Science (CIS)
Department of Computer Science and Engineering
Helsinki University of Technology (TKK)

Autumn 2007

Machine Learning Guest Lectures on 27 November

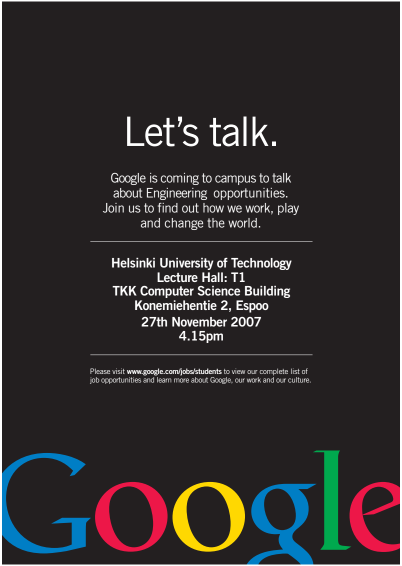
10–11 Juha Vesanto (Xtract): Data Mining in Practice

How to make succesfull analytics/data mining in an industry/corporate environment. Principles and a case study.

11–12 Hannu Helminen (Google): Machine Learning Methods in Web Search

Google is using machine learning methods in the presence of erroneous user queries and documents of low quality. Differences between a traditional information retrieval corpora and the web, and implications of these differences for improving queries and modeling the web are discussed. Inferring meaning from context and using this additional context for query expansion improves the quality of search results.

See `http://www.cis.hut.fi/Opinnot/T-61.3050/2007/guestlecture`

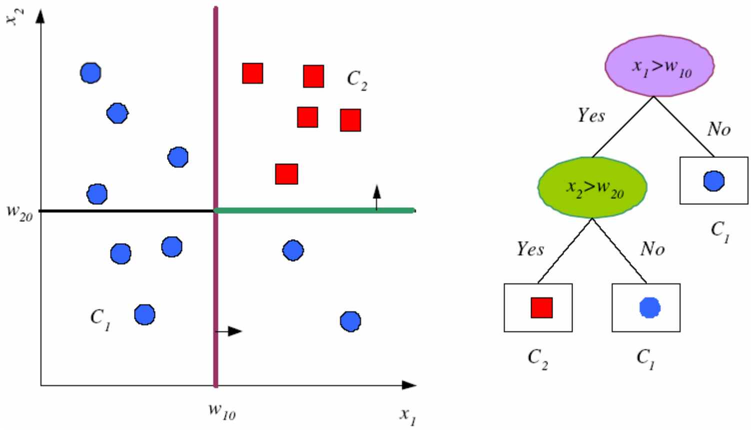


See http://www.cis.hut.fi/googletalk07/

Outline

1. Decision Trees
   - Classification Trees
   - Regression Trees
2. Linear Discrimination
   - Naive Bayes Classifier (Again)
   - Logistic Regression
   - Logistic Regression vs. Naive Bayes
3. Computing Sums and Products
   - Floating Point Numbers

Decision Trees



Lecture Notes for E Alpaydın 2004 Introduction to Machine Learning © The MIT Press (V1.1)

Decision Trees

- Each internal node tests an attribute.
- Each branch corresponds to set of attribute values.
- Each leaf node assigns a classification (classification tree) or a real number (regression tree).
- The tree is usually learned using a greedy algorithm built around $ID3$, such as $C4.5$. (The problem of finding optimal tree is generally NP-hard.)
- Advantages of trees:
  - Learning and classification is fast.
  - Trees are accurate in many domains.
  - Trees are easy to interpret as sets of decision rules.
- Often, trees should be used as a benchmark before more complicated algorithms are attempted.
- For alternative discussion, see Mitchell (1997), Ch 3.

ID3 algorithm for discrete attributes

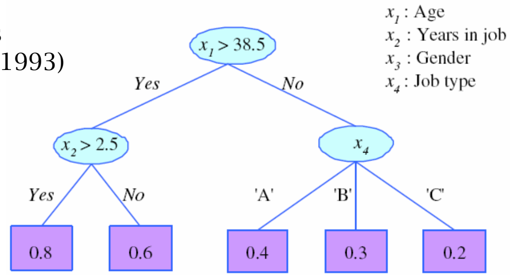
ID3($\mathcal{X}$) {Input: $\mathcal{X} = \{(r^t, \mathbf{x}^t)\}_{t=1}^N$, data set with binary attributes $r^t \in \{-1, +1\}$ and a vector of discrete variables $\mathbf{x}^t$. Output: T, classification tree.}
Create *root* node for T
If all items in $\mathcal{X}$ are positive (negative), return a single-node tree with label "+" ("-")
Let A be attribute that "best" classifies the examples
for all values v of A **do**
  Let $\mathcal{X}_v$ be subset of $\mathcal{X}$ that have value v for A
  if $\mathcal{X}_v$ is empty **then**
    Below the root of T, add a leaf node with most common label in $\mathcal{X}$
  else
    Below the root of T, add subtree ID3($\mathcal{X}_v$)
  end if
end for
return T

Variations of ID3

- Impurity measures:
  - Entropy: $-p_+ \log_2 p_+ - p_- \log_2 p_-$.
  - Gini index: $2p_+p_-$.
  - Misclassification error: $1 - \max(p_+, p_-)$.
  - All vanish for $p_+ \in \{0, 1\}$ and have a maximum at $p_+ = p_- = 1/2$.
- Continuous or ordered variables: sort x_A^t for some attribute A and find the best split $x_A \leq w$ vs. $x_A > w$.

Rule Extraction from Trees

C4.5Rules
(Quinlan, 1993)



R1: IF (age>38.5) AND (years-in-job>2.5) THEN y =0.8
R2: IF (age>38.5) AND (years-in-job≤2.5) THEN y =0.6
R3: IF (age≤38.5) AND (job-type='A') THEN y =0.4
R4: IF (age≤38.5) AND (job-type='B') THEN y =0.3
R5: IF (age≤38.5) AND (job-type='C') THEN y =0.2

Lecture Notes for E Alpaydın 2004 Introduction to Machine Learning © The MIT Press (V1.1)

Observations of ID3

- Inductive bias:
  - Preference on short trees.
  - Preference on trees with high information gain near root.
- Vanilla ID3 classifies the training data perfectly.
- Hence, in presence of noise, vanilla ID3 overfits.

Pruning

- How to avoid overfitting?
  - Prepruning: stop growing when data split is not statistically significant. For example: stop tree construction when node is smaller than a given limit, or impurity of a node is below a given limit θ_I. (*faster*)
  - Postpruning: grow the whole tree, then prune subtrees which overfit on the pruning (validation) set. (*more accurate*)

Pruning
Postpruning

- Split data into training and pruning (validation) sets.
- Do until further pruning is harmful:
    1. Evaluate impact on *pruning* set of pruning each possible node (plus those below it).
    2. Greedily remove the one that most improves the *pruning* set accuracy.
- Produces smallest version of most accurate subtree.
- Alternative: rule postpruning (commonly used, for example, $C4.5$).

Outline

1. Decision Trees
   - Classification Trees
   - Regression Trees
2. Linear Discrimination
   - Naive Bayes Classifier (Again)
   - Logistic Regression
   - Logistic Regression vs. Naive Bayes
3. Computing Sums and Products
   - Floating Point Numbers

Examples: Predicting woody cover in African savannas

- Task: woody cover (% of surface covered by trees) as a function of precipitation (MAP), soil characteristics (texture, total nitrogen total and phosphorus, and nitrogen mineralization), fire and herbivory regimes.
- Result: MAP is the most important factor.

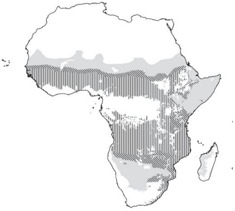


Figure 4 | The distributions of MAP-determined ('stable') and disturbance-determined ('unstable') savannas in Africa. Grey areas represent the existing distribution of savannas in Africa according to ref. 30. Vertically hatched areas show the unstable savannas (>784 mm MAP); cross-hatched areas show the transition between stable and unstable savannas (516–784 mm MAP); grey areas that are not hatched show the stable savannas (<516 mm MAP).

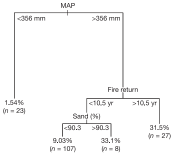


Figure 3 | Regression tree showing generalized relationships between woody cover and MAP, fire-return interval and percentage of sand. The tree is pruned to four terminal nodes and is based on 161 sites for which all data were available. No consistent herbivore effects were detected. Branches are labelled with criteria used to segregate data. Values in terminal nodes represent mean woody cover of sites grouped within the cluster. The pruned tree explained ~45.2% of the variance in woody cover, which is significantly more than a random tree ($P < 0.001$). Of this, 31% was accounted for by the first split; the second split explained an additional 10% of the variance in woody cover.

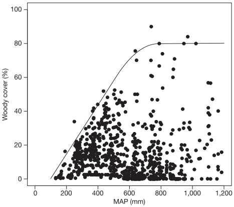


Figure 1 | Change in woody cover of African savannas as a function of MAP. Maximum tree cover is represented by using a 99th quantile piece wise linear regression. The regression analysis identifies the breakpoint (t rainfall at which maximum tree cover is attained) in the interval 650 ± 134 mm MAP (between 516 and 784 mm; see Methods). Trees are typically absent below 101 mm MAP. The equation for the line quantifyi the upper bound on tree cover between 101 and 650 mm MAP is Cover(%) = 0.14(MAP) − 14.2. Data are from 854 sites across Africa.

From Sankaran M et al. (2005) Determinants of woody cover in African savannas. Nature 438: 846–849.

Regression Trees

- Error at node m:

$$b_m(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \text{ reaches node } m \\ 0 & \text{otherwise} \end{cases}$$

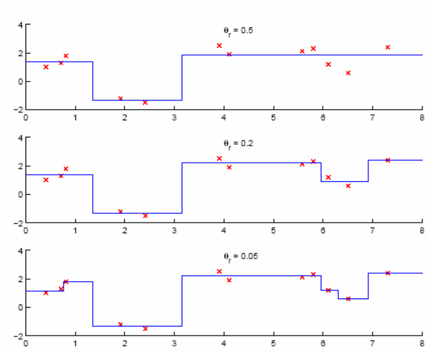
$$\mathcal{E}_m = \frac{1}{N_m} \sum_t \left(r^t - g_m\right)^2 b_m(\mathbf{x}^t) \quad , \quad g_m = \frac{\sum_t b_m(\mathbf{x}^t) r^t}{\sum_t b_m(\mathbf{x}^t)} .$$

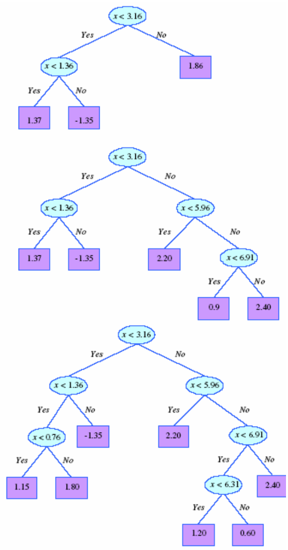
- After splitting:

$$b_{mj}(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \text{ reaches node } m \text{ and branch } j \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{E}_m = \frac{1}{N_m} \sum_j \sum_t \left(r^t - g_{mj}\right)^2 b_{mj}(\mathbf{x}^t) \quad , \quad g_{mj} = \frac{\sum_t b_{mj}(\mathbf{x}^t) r^t}{\sum_t b_{mj}(\mathbf{x}^t)} .$$

Model Selection in Trees:





Lecture Notes for E Alpaydın 2004 Introduction to Machine Learning © The MIT Press (V1.1)

Implementations

- There are many implementations, with sophisticated pruning methods.

```
> library(rpart)
> rpart(Hipparion ~ .,DD[,taxa])
n= 124

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 124 32 0 (0.74193548 0.25806452)
   2) Amphimachairodus=0 108 19 0 (0.82407407 0.17592593)
     4) Choerolophodon=0 96 13 0 (0.86458333 0.13541667)
       8) Ursus=0 76  6 0 (0.92105263 0.07894737) *
       9) Ursus=1 20  7 0 (0.65000000 0.35000000)
        18) Cervus=1 13  2 0 (0.84615385 0.15384615) *
        19) Cervus=0 7  2 1 (0.28571429 0.71428571) *
     5) Choerolophodon=1 12  6 0 (0.50000000 0.50000000) *
   3) Amphimachairodus=1 16  3 1 (0.18750000 0.81250000) *
```

Outline

Linear Discrimination

- Source material:
  - Alpaydin (2004) Ch 10, or
  - A new chapter by Mitchell (September 2005), "Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression", available as PDF at `http://www.cs.cmu.edu/~tom/NewChapters.html`

Naive Bayes Classifier

Common diagonal covariance matrix

- Idea: the means are class-specific, covariance matrix Σ is common and diagonal (Naive Bayes).
- d parameters in the covariance matrix.
- Discriminant is linear: $g_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x} + w_{i0}$, where $\mathbf{w}_i = \Sigma^{-1}\mu_i$ and $w_{i0} = -\frac{1}{2}\mu_i^T \Sigma^{-1}\mu_i + \log P(C_i)$.

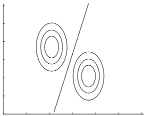
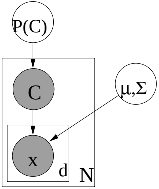
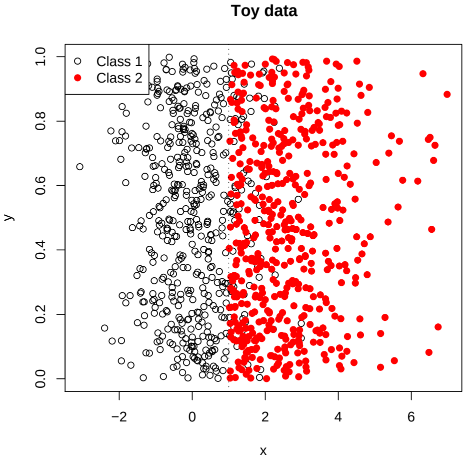


Figure 5.5: All classes have equal, diagonal covariance matrices but variances are not equal. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*



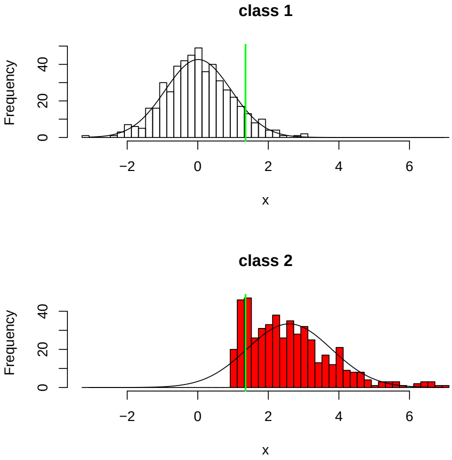
Naive Bayes Classifier

Using Naive Bayes Classifier



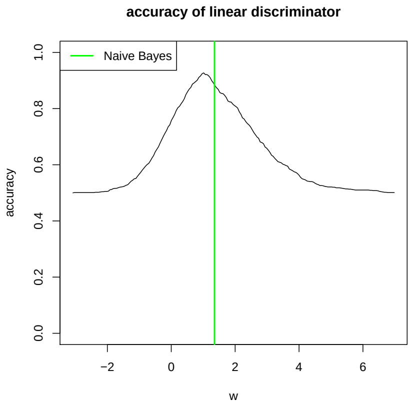
Naive Bayes Classifier

Using Naive Bayes Classifier



Naive Bayes Classifier

Using Naive Bayes Classifier



Accuracy of discriminator "class 1 if $x < w$, class 2 if $x \geq w$".

Naive Bayes Classifier

- $\mathcal{X} = \{(r^t, \mathbf{x}^t)\}_{t=1}^N$, $r^t \in \{0, 1\}$, $\mathbf{x}^t \in \mathbb{R}^d$.
- Naive Bayes assumption: $P(\mathbf{x}^t \mid r^t) = \prod_{i=1}^d P(x_i^t \mid r^t)$.
- Using Bayes rule,

$$P(r \mid \mathbf{x}) = \frac{P(r) \prod_{i=1}^d P(x_i \mid r)}{\sum_{s \in \{0,1\}} P(s) \prod_{i=1}^d P(x_i \mid s)}.$$

- Discriminant is linear:
  $g_i(\mathbf{x}) = \log P(r_i = 1 \mid \mathbf{x}) + \mathrm{const.} = \mathbf{w}_i^T \mathbf{x} + w_{i0}$, where
  $\mathbf{w}_i = \Sigma^{-1} \mu_i$ and $w_{i0} = -\frac{1}{2} \mu_i^T \Sigma^{-1} \mu_i + \log P(C_i)$.
- Observation:

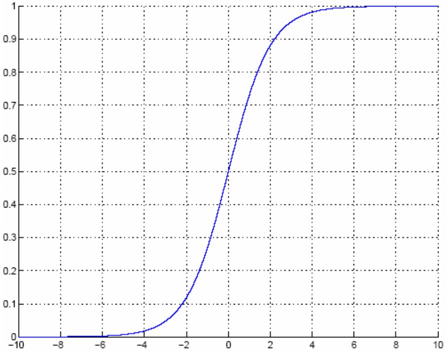
$$\log \frac{P(r = 1 \mid \mathbf{x})}{1 - P(r = 1 \mid \mathbf{x})} = \mathbf{w}^T \mathbf{x} + \mathbf{w}_0.$$

Outline

Logistic Regression

- Logit: $\mathrm{logit}(p) = \log\left(\frac{p}{1-p}\right)$.
- Sigmoid: $\mathrm{sigmoid}(t) = \mathrm{logit}^{-1}(t) = 1/(1+e^{-t})$.
- Derivative of sigmoid:
  $\mathrm{sigmoid}'(t) = \mathrm{sigmoid}(t)\,(1-\mathrm{sigmoid}(t))$.

Sigmoid (Logistic) Function



1. Calculate $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ and choose C_1 if $g(\mathbf{x}) > 0$, or
2. Calculate $y = \text{sigmoid}(\mathbf{w}^T\mathbf{x} + w_0)$ and choose C_1 if $y > 0.5$

Cost Function for Logistic Regression

- $$P(R \mid X, W) = \prod_{t=1}^{n} P(r^t \mid \mathbf{x}^t, W)$$
- $$\mathcal{L} = e^{-P(R|X,W)} = -\sum_{t=1}^{N} \left(r^t \log y^t - (1 - r^t) \log (1 - y^t) \right),$$
  where $y^t = P(r^t = 1 \mid \mathbf{x}) = \mathrm{sigmoid}(\mathbf{w}^t \mathbf{x} + w_0)$.
- Task: find $W = (\mathbf{w}, w_0)$ such that $\mathcal{L}$ is minimized.
- No EM algorithm. Use gradient ascent.

Gradient Ascent

GRADASC($\mathcal{L}(\theta)$, θ^0) {Input: $\mathcal{L}(\theta)$, cost function; θ^0, initial parameters. Output: θ, a local minimum of $\mathcal{L}$.}

$\theta \leftarrow \theta^0$ $\{\theta, \theta^0 \in \mathbb{R}^d.\}$

$t \leftarrow 1$

repeat

  for all $i \in \{1, \ldots, d\}$ **do**

    $\Delta\theta_i \leftarrow \partial\mathcal{L}(\theta)/\partial\theta_i$

  end for

  for all $i \in \{1, \ldots, d\}$ **do**

    $\theta_i \leftarrow \theta_i - \eta_t \Delta\theta_i$

  end for

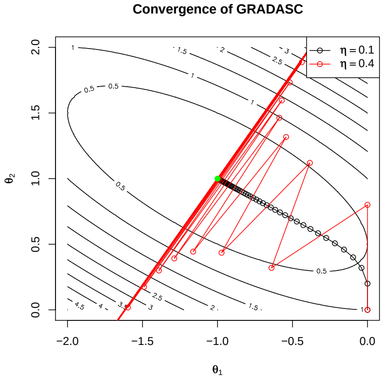
  $t \leftarrow t + 1$

until convergence

return θ

Gradient Ascent

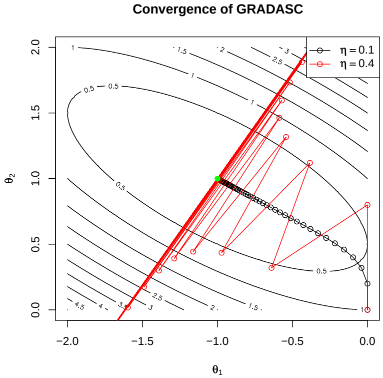
- The function GRADASC always converges if $\sum_{t=1}^{\infty} \eta_t = \infty$ and $\sum_{t=1}^{\infty} \eta_t^2 < \infty$, where $\eta_t \geq 0$ for all t, for example, $\eta_t = 1/t$.
- The function GRADASC often converges also for constant small enough $\eta_t = \eta > 0$.



Minimizing $\mathcal{L}(\theta) = (\theta_1 + \theta_2)^2 + (\theta_2 - 1)^2$, using $\theta^0 = (0, 0)^T$.

Gradient Ascent

- GRADASC is inefficient.
- Usually one should use a more sophisticated gradient ascent algorithm, such as conjugate gradient, from some numerical library (e.g., in R type `help(optim)`).



Minimizing $\mathcal{L}(\theta) = (\theta_1 + \theta_2)^2 + (\theta_2 - 1)^2$, using $\theta^0 = (0, 0)^T$.

Gradient Ascent

- Logistic regression may converge to $w \to \pm\infty$ (see right), especially when data is high dimensional and sparse. This causes problems.
- Solution: minimize regularized cost $\mathcal{L} \to \mathcal{L} + \frac{1}{2}\lambda\left(w_0^2 + \mathbf{w}^T\mathbf{w}\right)$.

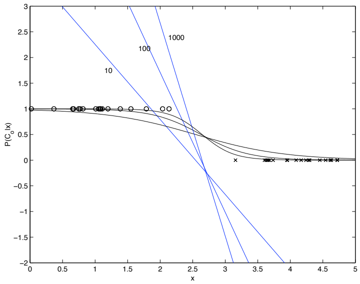


Figure 10.7: For a univariate two-class problem (shown with '$\circ$' and '$\times$'), the evolution of the line $wx + w_0$ and the sigmoid output after 10, 100, and 1,000 iterations over the sample. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

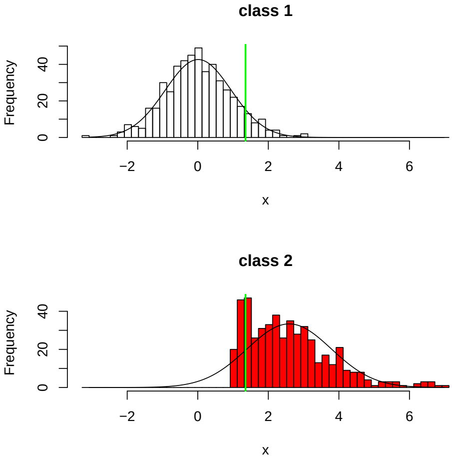
Generalized Linear Models

- Logistic regression is a special case of Generalized Linear Models (GLM)
  - logit is a link function.
- Many respectable numerical packages contain GLM implementation which includes logistic regression (e.g., in R `help(glm)`). You should probably use these in real life applications instead of programming one on your own.

Outline

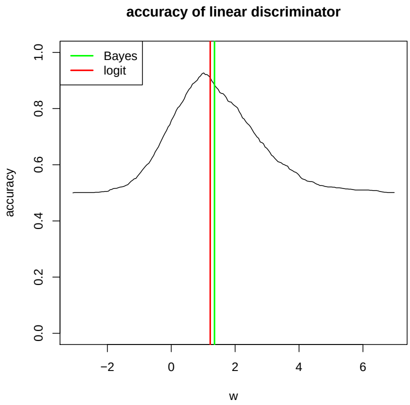
Naive Bayes Classifier

Using Naive Bayes Classifier



Naive Bayes Classifier

Using Naive Bayes Classifier



Accuracy of discriminator "class 1 if $x < w$, class 2 if $x \geq w$".

Naive Bayes vs. Logistic Regression

- Naive Bayes classifier estimates parameters of $P(\mathbf{r})$ and $P(\mathbf{x} \mid r)$ (means, covariances, etc.). (generative classifier, because we can generate the data points, given parameters)
- Logistic regression directly estimates the parameters of $P(r \mid \mathbf{x})$. (discriminative classifier, because we can directly discriminate wrt. r, given $\mathbf{x}$; no generative model for $p(\mathbf{x})$ is needed)
- If Naive Bayes assumptions hold (data from multivariate Gaussians with diagonal covariate matrix) and the number of training examples is very large, Naive Bayes and logistic regression give identical classification.

Naive Bayes vs. Logistic Regression

- The differences:
  - If data is not Gaussian etc. (that is, NB assumptions do not hold), logistic regression often gives better result (at least for large amounts of data).
  - Logistic regression needs more data. Naive Bayes needs $N = O(\log d)$ samples, while logistic regression needs $N = O(d)$. Ng & Jordan (2002) On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. In Proc NIPS 14..
- Generative classifier: more bias, less variance. There is a model for $P(\mathbf{x})$. This is good if there is little data and/or the model for $\mathbf{x}$ is correct enough.
- Discriminative classifier: less bias, more variance. There is no model for $P(\mathbf{x})$, it is estimated directly from data. This is good if the NB model for $\mathbf{x}$ is wrong and/or there is enough data.

Outline

1. Decision Trees
   - Classification Trees
   - Regression Trees
2. Linear Discrimination
   - Naive Bayes Classifier (Again)
   - Logistic Regression
   - Logistic Regression vs. Naive Bayes
3. Computing Sums and Products
   - Floating Point Numbers

Numerical Computation: Computing Sums and Products

- Sometimes it is enough to use $+$ and $*$ operators to compute sums and products. According to R: 3.14*42=131.88; 3.14+42+5=50.14.
- Sometimes it is not. According to R: 3.14e-200*42e-201*1e300=0; 1e-400*1e400=NaN; 1e-16+1-1=0.
- In probabilistic modeling it is typical to...
  - Have numbers of different orders of magnitudes, including very small numbers.
  - Do sums and products with them.
- Important numbers (examples from the R floating point implementation in Mac OS X, `help(.Machine)`):
  - Smallest positive floating point number ϵ (machine epsilon) for which $1+\epsilon \neq 1$: 2.2×10^{-16}.
  - The largest finite floating point number: 1.7×10^{308}.
  - The smallest positive floating point number: 2.2×10^{-308}.

Numerical Computation: Representing Numbers

- In many practical applications, 2.2×10^{-308} is too large for representing intermediate probabilities.
- Solution: store numbers as logs.
- Probabilities are usually always positive. (Generally, software should however be written so that to work consistently also with zero probabilities.)
- R is consistent also for zero probabilities: log(0)=-Inf; exp(-Inf)=0.
- Other software may behave differently. Read the documentation and test.

Numerical Computation: Computing Products

- Task: compute the product $y = \prod_{i=1}^{n} x_i$.
- 1e-200*1e-200*1e300=0 (wrong!).
- Solution: use logs.
- $\log y = \sum_{i=1}^{n} \log x_i$.
- log(1e-200)+log(1e-200)+log(1e300)=log(1e-100) (correct).
- Division: $\log(x/y) = \log x - \log y$. Product with negatives.

Numerical Computation: Computing Sums

- Task: compute sum $y = \sum_{i=1}^{n} x_i$.
- exp(-1000)+exp(-999)=0 (wrong!).
- Solution: scale numbers appropriately before doing the sum.
- $\log y = \log x_{MAX} + \log\left(\sum_{i=1}^{n} \exp\left(\log x_i - \log x_{MAX}\right)\right)$, where $\log x_{MAX} = \max_i \log x_i$.
- -999+log(exp(-1)+exp(0))=-998.6 (correct).
- Something like this:

```
safesum <- function(x) { xmax <- max(x) ;
xmax+log(sum(exp(x-xmax))) }
```

Numerical Computation: Example

Naive Bayes' classifier

$$P(C_i \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid C_i)P(C_i)}{\sum_{k=1}^{K} P(\mathbf{x} \mid C_k)P(C_k)} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

Store numbers as logs and denote: $a[i] = \log P(\mathbf{x} \mid C_i)$, $b[i] = \log P(C_i)$.

```
safesum <- function(x) { xmax <- max(x); xmax+log(sum(exp(x-xmax)))) }

evidence <- safesum(a+b)

posterior <- sum(c(a[i],b[i],-evidence))

exp(posterior) #P(C_i | x)
```