T-61.3050 Machine Learning: Basic Principles

Recap

Kai Puolamäki

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

Autumn 2007

Outline

1. Announcements
   - Examination
   - Course Feedback
2. Summary of the Course
   - Summary of the Course
3. Overflow
   - Optimization Algorithms
   - Computing Sums and Products
   - Validation and Cross-Validation

- To pass the course you must pass the examination and the term project.
- Grading:
  - Examination grade $E \in [0, 1]$ (0 smallest passed grade)
  - Term project grade $T \in [0, 1]$ (0 smallest passed grade)
  - Problem session grade $P \in [0, 1]$
  - Course grade $\texttt{min}(5,\texttt{floor}(4E + 2T + P))$

Examination

- Currently scheduled at 19 Dec & 2 Feb & 15 May (check the times and locations from the examination schedule!)
- You must sign in to the examination at least one week in advance using WWWTopi
- Calculator (with memory erased) is allowed
- No other extra material is allowed.
- 4–6 problems (to pass you have to get about half of the points)

Reading List

- The examination is based on the topics covered in the lectures
- See http://www.cis.hut.fi/Opinnot/T-61.3050/2007/examination

Outline

1. Announcements
   - Examination
   - Course Feedback
2. Summary of the Course
   - Summary of the Course
3. Overflow
   - Optimization Algorithms
   - Computing Sums and Products
   - Validation and Cross-Validation

Course Feedback

- Please give course feedback at http://www.cs.hut.fi/Opinnot/Palaute/kurssipalaute-en.html
- (Open until 7 January 2008)

Outline

1. Announcements
   - Examination
   - Course Feedback
2. Summary of the Course
   - Summary of the Course
3. Overflow
   - Optimization Algorithms
   - Computing Sums and Products
   - Validation and Cross-Validation

Objectives

- After this course, the student should. . .
  1. be able to apply the basic methods to real world data;
  2. understand the basic principles of the methods; and
  3. have necessary prerequisites to understand and apply new concepts and methods that build on the topics covered in the course.
- The topic is difficult (and interdisciplinary, involving at least computer science, mathematics, computational modeling and statistics)

Learning Tasks

- Supervised learning
  - classification
  - regression
- Unsupervised learning
  - clustering etc.
- Reinforcement learning [not in this course]

Concept Learning

- Task: classify a previously unseen instance into positive or negative
- Hypothesis class $\mathcal{H}$
- Learning: use positive and negative examples to prune out the hypothesis
- If none of the hypothesis in the hypothesis class is correct we might end up with no consistent hypothesis.
- Inductive bias: we must restrict the allowed hypothesis to be able to generalize (predict classes of new instances).
- The choice of a hypothesis space is called model selection.
- Underfitting: the hypothesis space is too simple.
- Overfitting: the hypothesis space is too complex.
- VC dimension can be used to measure the complexity of the hypothesis space.

Concept Learning

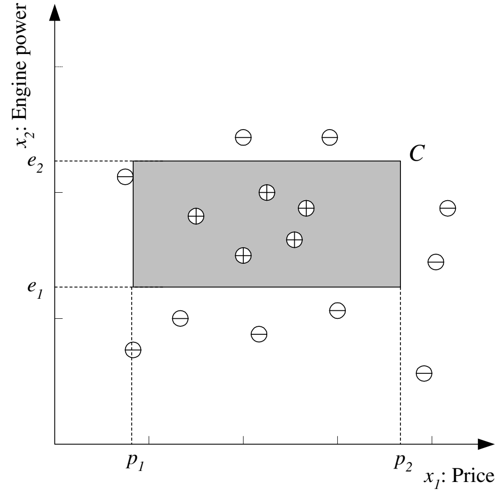


Figure 2.2 of Alpaydin (2004).

Regression with Noise

- Classification is the prediction of a 0–1 class, given attributes.
- Regression is the prediction of a real number, given
- Usually, we want to minimize a quadratic error function,

$$E(g \mid \mathcal{X}) = \frac{1}{N} \sum_{t=1}^{N} \left(r^t - g(\mathbf{x}^t)\right)^2 .$$

Dimensions of a Supervised Learner

Model

$$g(\mathbf{x} \mid \theta)$$

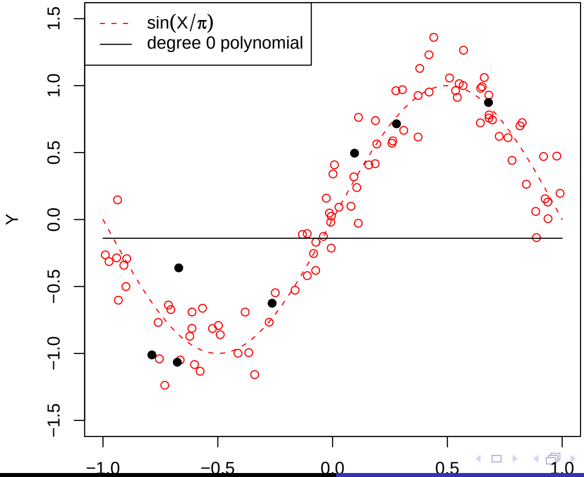
Loss Function

$$E\left(\theta \mid \mathcal{X}\right) = \frac{1}{N} \sum_{t=1}^{N} L\left(r^t, g(\mathbf{x}^t \mid \theta)\right).$$

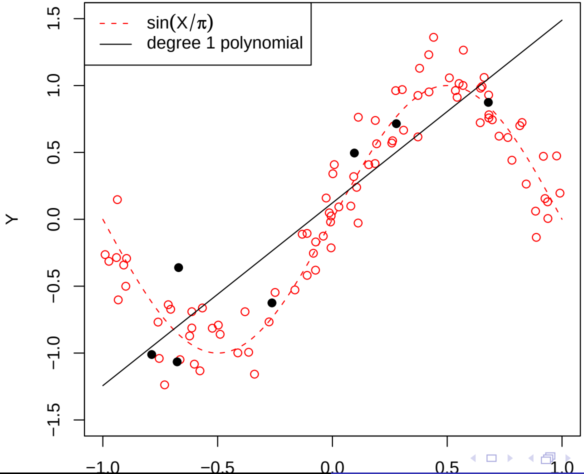
Optimization Procedure

$$\theta \leftarrow \arg \min_{\theta} E\left(\theta \mid \mathcal{X}\right).$$

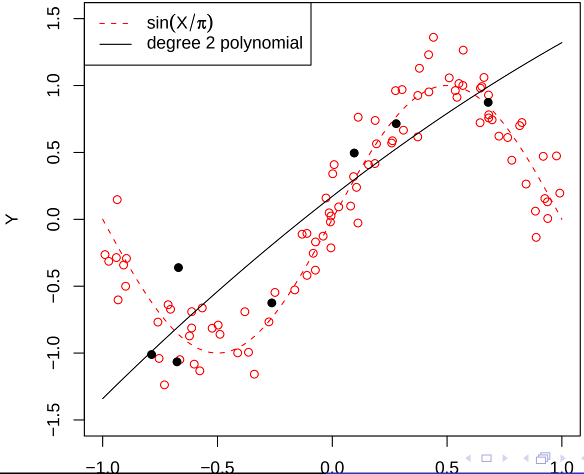
Polynomial Regressors



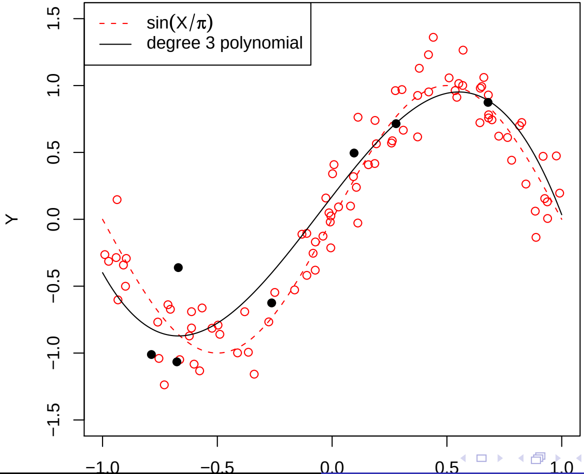
Polynomial Regressors



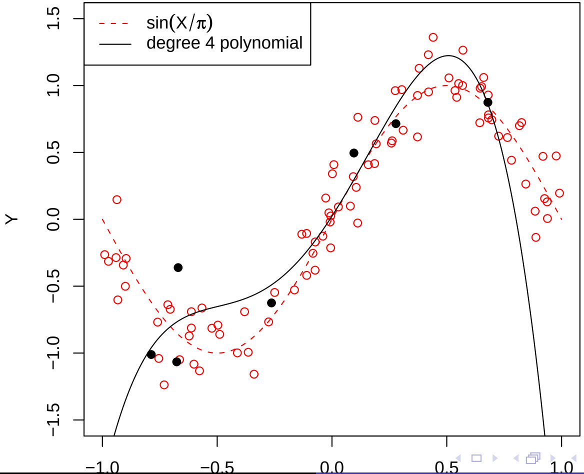
Polynomial Regressors



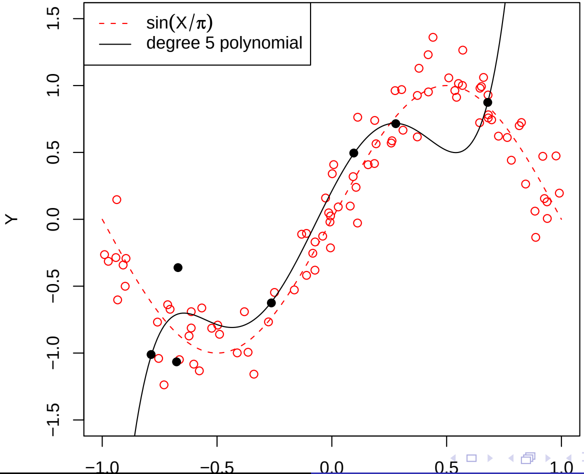
Polynomial Regressors



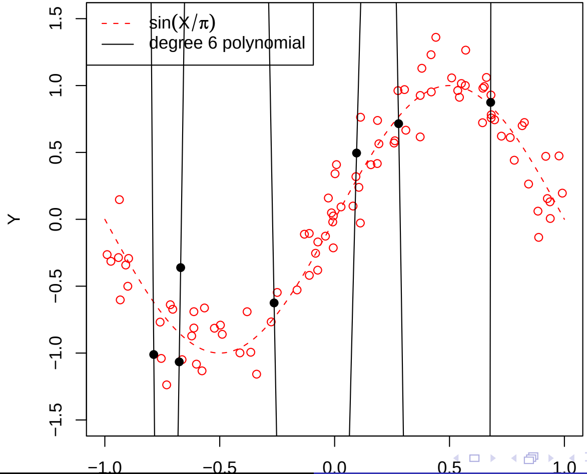
Polynomial Regressors



Polynomial Regressors

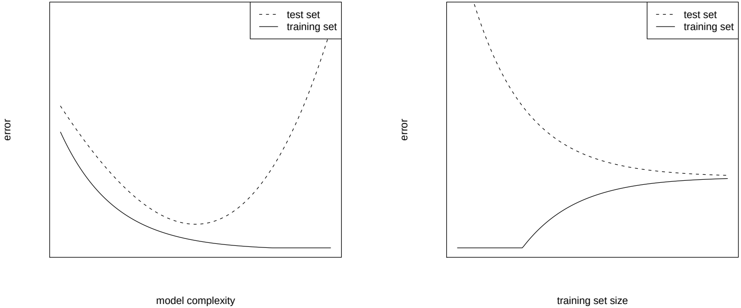


Polynomial Regressors



Model Selection and Generalization

Schematic illustration of the empirical vs. generalization error



- empirical error = error on training set
- generalization error = error on test set
- We see empirical error, but want to minimize the error on new data

K-Fold Cross-Validation

- How to use the training/validation data most efficiently?

```
CV($\mathcal{X}$,$A$,$K$){Input: $\mathcal{X}$, data $\mathcal{X} = \{(r^t, x^t)\}_{t=1}^N$; $A$, classification algorithm; $K$, number of folds. Output: $\mathcal{E}$, error measure.}
Partition $\mathcal{X}$ in random into $K$ roughly equally sized parts $\mathcal{X}_i$.
for all $i \in \{1, \ldots, K\}$ do
  Train $A$ using $\mathcal{X} \setminus \mathcal{X}_i$ as a training set.
  Let $\mathcal{E}_i$ be the error of $A$ in $\mathcal{X}_i$ (for example, the fraction of incorrectly labeled items).
end for
$\mathcal{E} \leftarrow \sum_{i=1}^K |\mathcal{X}_i| \mathcal{E}_i / |\mathcal{X}|$
return $\mathcal{E}${$\mathcal{E}$ can be used as a validation set error in model selection.}
```

Rules of Probability

- In presence of noise, we have to use probabilities.
- In principle, you can derive everything in probabilistic inference from the basic axiom, including the sum and product rules.

Rules of Probability

- $P(E, F) = P(F, E)$: probability of both E and F happening.
- $P(E) = \sum_F P(E, F)$ (sum rule, marginalization)
- $P(E, F) = P(F \mid E)P(E)$ (product rule, conditional probability)
- Consequence: $P(F \mid E) = P(E \mid F)P(F)/P(E)$ (Bayes' formula)
- We say E and F are independent if $P(E, F) = P(E)P(F)$ (for all E and F).
- We say E and F are conditionally independent given G if $P(E, F \mid G) = P(E \mid G)P(F \mid G)$, or equivalently $P(E \mid F, G) = P(E \mid G)$.

Bayes' Rule

Classification to K classes

$$P(C_i \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid C_i)P(C_i)}{P(\mathbf{x})} = \frac{P(\mathbf{x} \mid C_i)P(C_i)}{\sum_{k=1}^{K} P(\mathbf{x} \mid C_k)P(C_k)}$$

- $P(C_k) \geq 0$ and $\sum_{k=1}^{K} P(C_k) = 1$.
- Naive Bayes Classifier: choose C_k where $k = \arg\max_k P(C_k \mid \mathbf{x})$.

Classifier Using Probabilistic Model

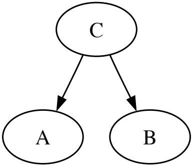
- First compute posterior class probability $P(C \mid \mathbf{x})$.
- Choose class C with the largest posterior probability.
- Another option: Choose class which minimizes risk (or maximizes utility), if the loss of misclassification is not a constant.
- A class for uncertainty: reject-option

Bayesian Networks

Bayesian network is a directed acyclic graph (DAG) that describes a joint distribution over the vertices $X_1, \ldots, X_d$ such that

$$P(X_1, \ldots, X_d) = \prod_{i=1}^{d} P(X_i \mid \mathrm{parents}(X_i)),$$

where $\mathrm{parents}(X_i)$ are the set of vertices from which there is an edge to X_i.



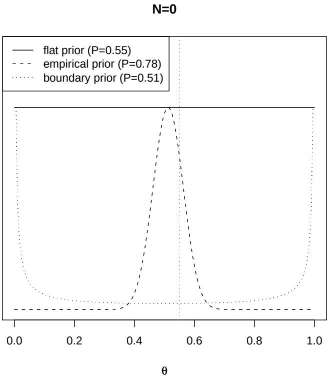
$$P(A, B, C) = P(A \mid C) P(B \mid C) P(C).$$

(A and B are conditionally independent given C.)

Estimating the Sex Ratio

- What is our degree of belief in the gender ratio, before seeing any data (prior probability density $p(\theta)$)?
- What is our degree of belief in the gender ratio, after seeing data X (posterior probability density $p(\theta \mid \mathcal{X})$)?

$$p(\theta \mid \mathcal{X}) \propto p(\theta)p(\mathcal{X} \mid \theta).$$

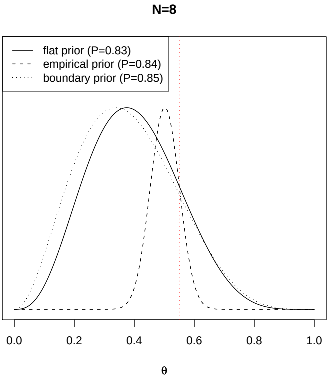


“True” $\theta = 0.55$ is shown by the red dotted line. The densities have been scaled to have a maximum of one.

Estimating the Sex Ratio

- What is our degree of belief in the gender ratio, before seeing any data (prior probability density $p(\theta)$)?
- What is our degree of belief in the gender ratio, after seeing data X (posterior probability density $p(\theta \mid \mathcal{X})$)?

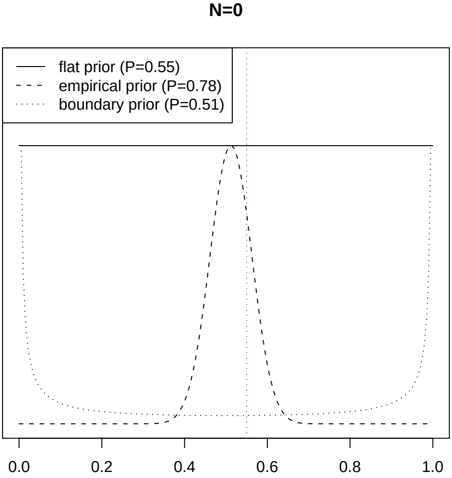
$$p(\theta \mid \mathcal{X}) \propto p(\theta)p(\mathcal{X} \mid \theta).$$



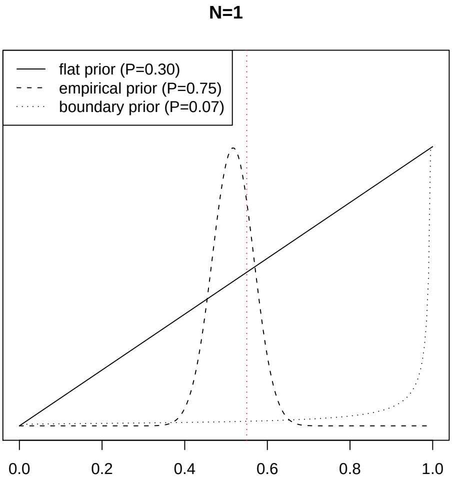
"True" $\theta = 0.55$ is shown by the red dotted line. The densities have been scaled to have a maximum of one.

Estimating the Sex Ratio

Posterior probability density

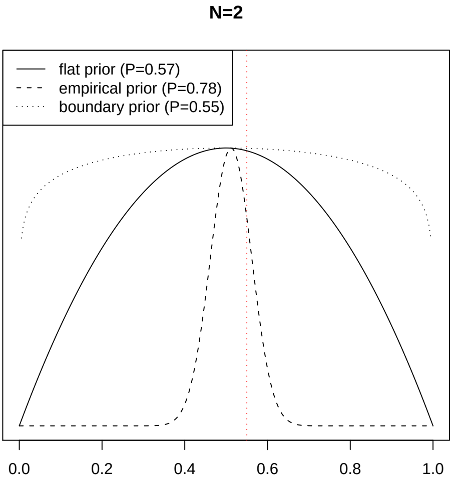


Estimating the Sex Ratio
Posterior probability density



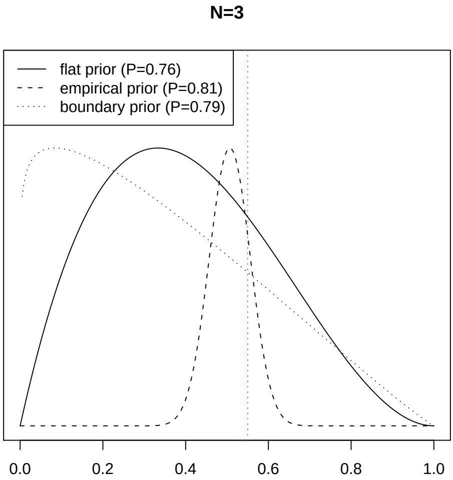
Estimating the Sex Ratio

Posterior probability density



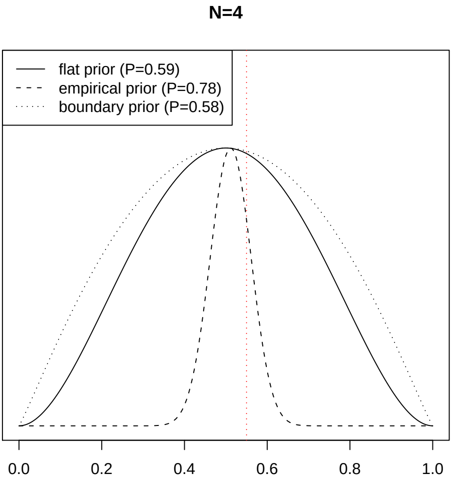
Estimating the Sex Ratio

Posterior probability density



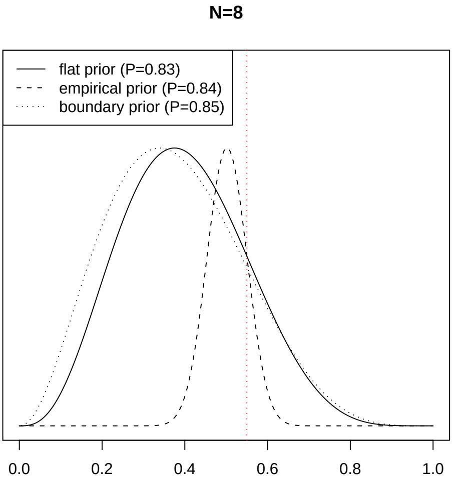
Estimating the Sex Ratio

Posterior probability density



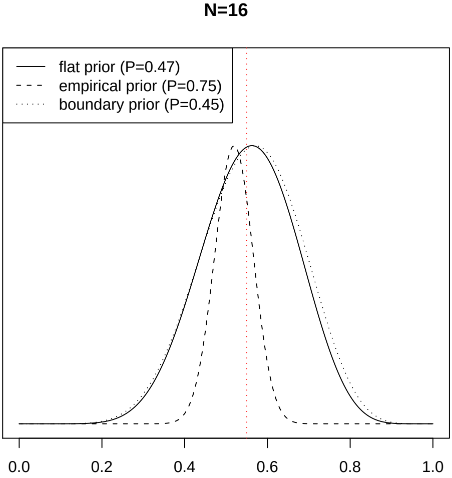
Estimating the Sex Ratio

Posterior probability density



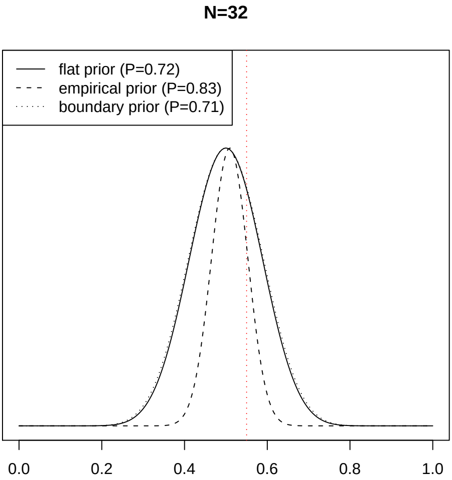
Estimating the Sex Ratio

Posterior probability density



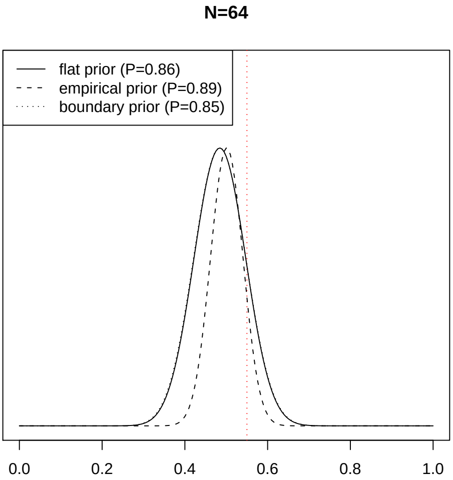
Estimating the Sex Ratio

Posterior probability density



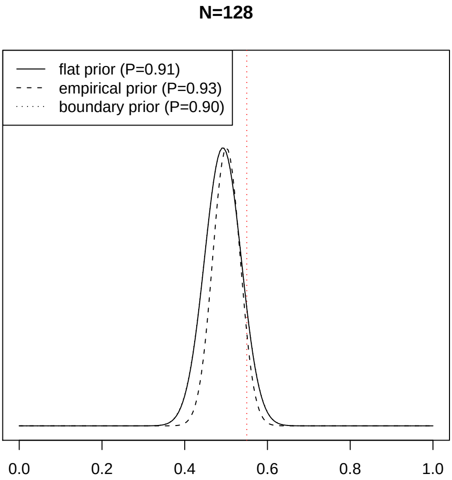
Estimating the Sex Ratio

Posterior probability density



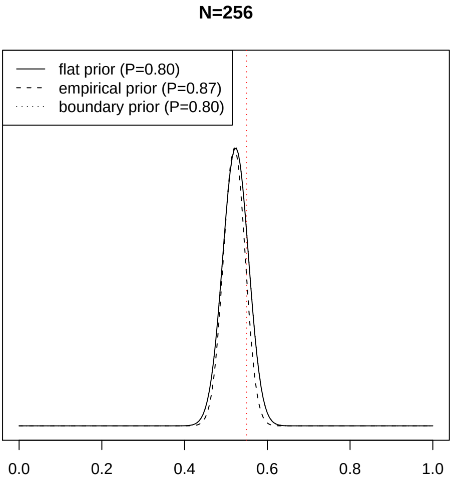
Estimating the Sex Ratio

Posterior probability density



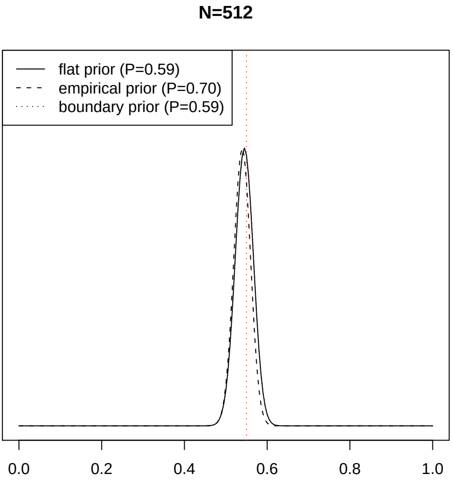
Estimating the Sex Ratio

Posterior probability density



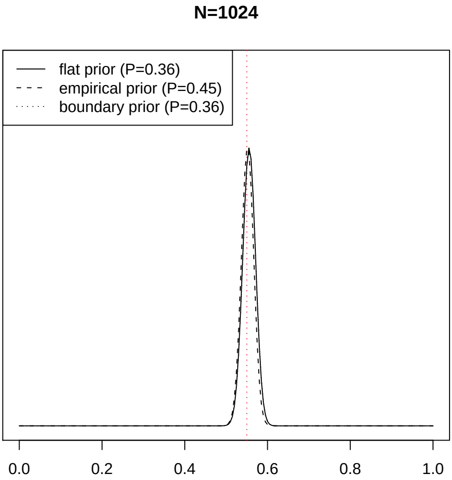
Estimating the Sex Ratio

Posterior probability density



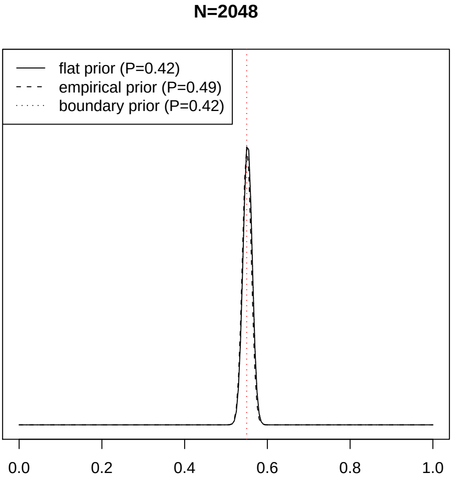
Estimating the Sex Ratio

Posterior probability density



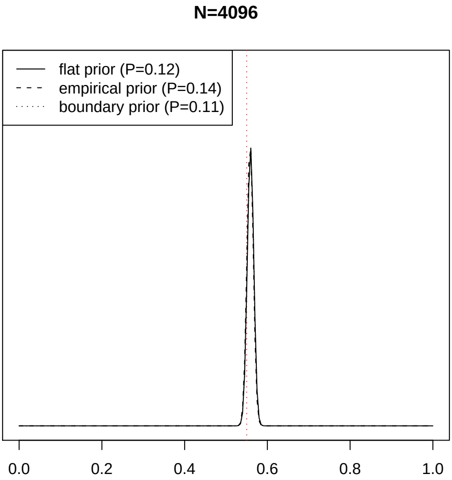
Estimating the Sex Ratio

Posterior probability density



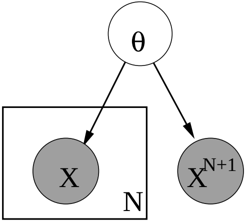
Estimating the Sex Ratio

Posterior probability density



Predictions from the Posterior Probability Density

- Task: predict probability of x^{N+1}, given N observations in $\mathcal{X}$.
- Marginalizations:
  - $p(\mathcal{X},\theta) = \int dx^{N+1} p(x^{N+1},\mathcal{X},\theta) = p(\mathcal{X} \mid \theta) p(\theta)$.
  - $p(\mathcal{X}) = \int d\theta p(\mathcal{X},\theta) = \int d\theta p(\mathcal{X} \mid \theta) p(\theta)$.
  - $p(x^{N+1},\mathcal{X}) = \int d\theta p(x^{N+1},\mathcal{X},\theta) = \int d\theta p(x^{N+1} \mid \theta) p(\mathcal{X} \mid \theta) p(\theta)$.
- Posterior: $p(\theta \mid \mathcal{X}) = p(\mathcal{X},\theta)/p(\mathcal{X})$.
- Predictor for new data point: $p(x^{N+1} \mid \mathcal{X}) = p(x^{N+1},\mathcal{X})/p(\mathcal{X}) = \int d\theta p(x^{N+1} \mid \theta) p(\mathcal{X},\theta)/p(\mathcal{X}) = \int d\theta p(x^{N+1} \mid \theta) p(\theta \mid \mathcal{X})$.



Joint distribution ($\mathcal{X} = \{x^t\}_{t=1}^N$): $p(x^{N+1},\mathcal{X},\theta) = p(x^{N+1} \mid \theta) p(\mathcal{X} \mid \theta) p(\theta)$.

Point Estimators

- The posterior $p(\theta \mid \mathcal{X})$ represents our best knowledge.
- Predictor for new data point:
  $p(x^{N+1} \mid \mathcal{X}) = \int d\theta p(x^{N+1} \mid \theta) p(\theta \mid \mathcal{X})$.
- The calculation of the integral may be infeasible.
- Estimate θ by $\hat{\theta}$ (or posterior by $p(\theta \mid \mathcal{X}) \approx \delta(\theta - \hat{\theta})$) and use the predictor

$$p(x^{N+1} \mid \mathcal{X}) \approx p(x^{N+1} \mid \hat{\theta}).$$

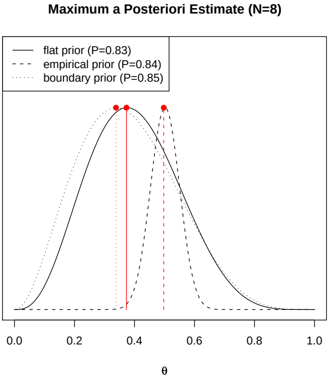
Estimators from the Posterior

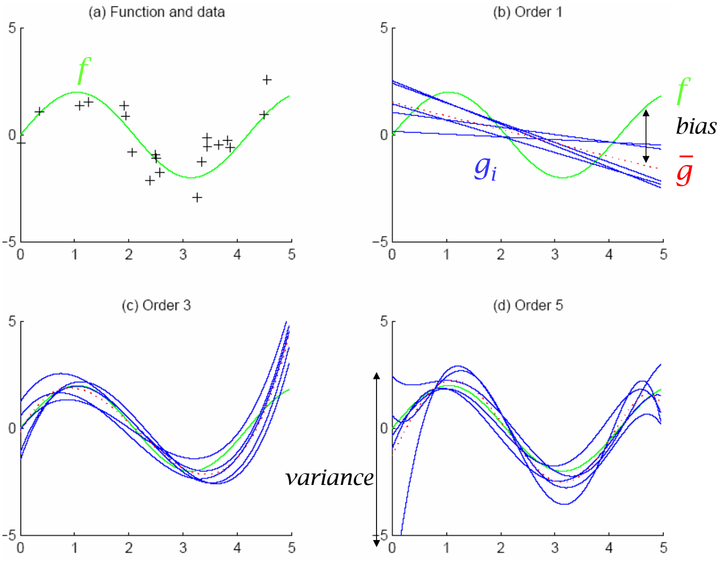
Definition (Maximum Likelihood Estimate)

$$\hat{\theta}_{ML} = \arg\max_{\theta} \log p(\mathcal{X} \mid \theta).$$

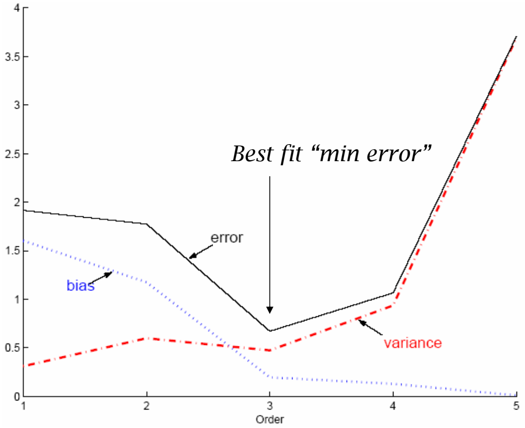
Definition (Maximum a Posteriori Estimate)

$$\hat{\theta}_{MAP} = \arg\max_{\theta} \log p(\theta \mid \mathcal{X}).$$



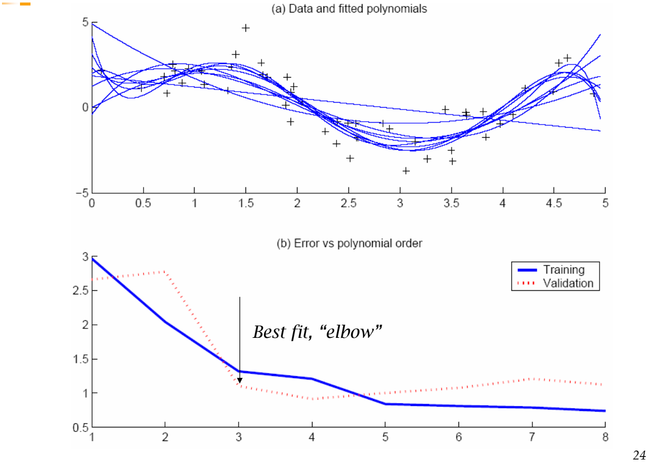


Lecture Notes for E Alpaydın 2004 Introduction to Machine Learning © The MIT Press (V1.1)



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

Polynomial Regression



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

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 $w_{i0} = -\frac{1}{2}\mu_i^T \Sigma^{-1}\mu_i + \log P(C_i)$.

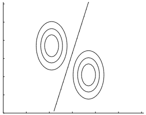
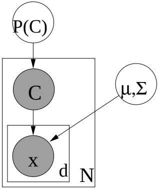
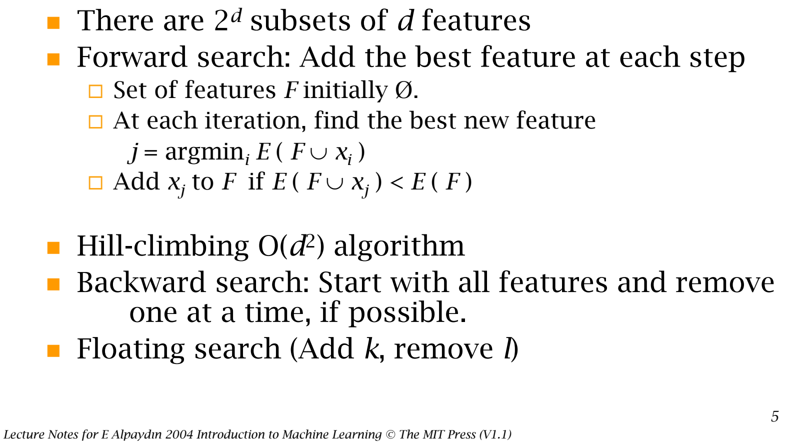


Figure 5.5: All classes have equal, diagonal covariance matrices but variances are not equal. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*



- Cross-validation: most robust if there is enough data.
- Structural risk minimization (SRM): used, for example, in support vector machines (SVM).
- Bayesian model selection: use prior and Bayes' formula.
- Minimum description length (MDL): can be viewed as MAP estimate.
- Regularization: add penalty term for complex models (can be obtained, for example, from prior).
- Latter four methods do not strictly require validation set (at least if implicit modeling assumptions are satisfied, such as that in Bayesian model selection the data is from the model family; it is always a good idea to use a test set) and latter three are related.
- There is no single best way for small amounts of data (your prior assumptions matter).

Subset Selection



- There are 2^d subsets of d features
- Forward search: Add the best feature at each step
  - □ Set of features F initially Ø.
  - □ At each iteration, find the best new feature $j = \text{argmin}_i \, E(F \cup x_i)$
  - □ Add x_j to F if $E(F \cup x_j) < E(F)$
- Hill-climbing $O(d^2)$ algorithm
- Backward search: Start with all features and remove one at a time, if possible.
- Floating search (Add k, remove l)

5

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

Principal Component Analysis (PCA)

- Observation: covariance matrix of $\{\mathbf{z}^t\}_{t=1}^N$ is a diagonal matrix D whose diagonal elements are the variances.

$$\begin{aligned} S_{\mathbf{z}} &= \sum_t \mathbf{z}\mathbf{z}^T/N = \sum_t C^T\mathbf{y}\mathbf{y}^T C/N \\ &= C^T\left(\sum_t \mathbf{y}\mathbf{y}^T/N\right)C = C^T S C = D, \end{aligned}$$

where the diagonal elements of D are the variances $D_{ii} = \sigma_{\mathbf{z}i}^2$.

- Eigenvalues $\lambda_i \Leftrightarrow$ variances σ_i^2.

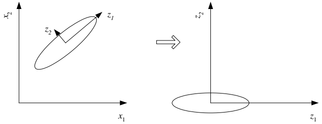
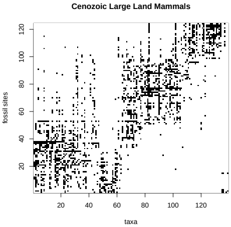
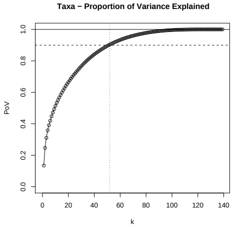


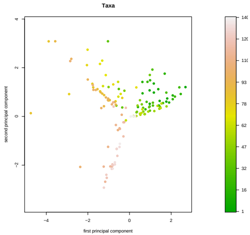
Figure 6.1: Principal components analysis centers the sample and then rotates the axes to line up with the directions of highest variance. If the variance on z_2 is too small, it can be ignored and we have dimensionality reduction from two to one. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Example: Fossils

- Large European land mammals: 124 fossil find sites (dated 23–2 million years old), 139 taxa
- Reconstruction of site vectors given PCA taxon representation for different k: $\hat{\mathbf{y}} = W\hat{\mathbf{z}} = WW^T\mathbf{y}$, or $\hat{\mathbf{x}} = WW^T(\mathbf{x} - \mathbf{m}) + \mathbf{m}$.

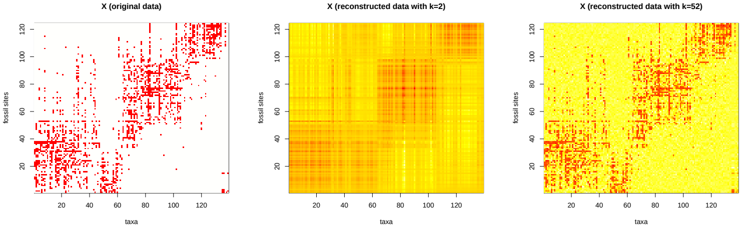






Example: Fossils

- Large European land mammals: 124 fossil find sites (dated 23–2 million years old), 139 taxa
- Reconstruction of site vectors given PCA taxon representation for different k: $\hat{\mathbf{y}} = W\hat{\mathbf{z}} = WW^T\mathbf{y}$, or $\hat{\mathbf{x}} = WW^T(\mathbf{x} - \mathbf{m}) + \mathbf{m}$.



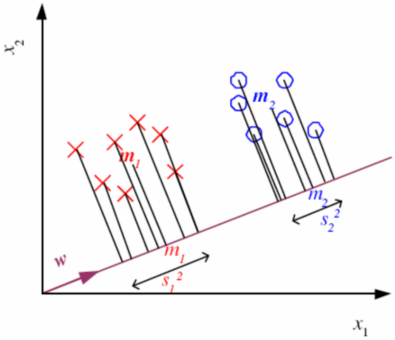
Linear Discriminant Analysis (LDA)

- PCA is unsupervised method (class information is not usually used).
- Linear Discriminant Analysis (LDA) is supervised method for dimensionality reduction in classification problems.
- As PCA, LDA can be accomplished with standard matrix algebra (eigenvalue decompositions etc.). This makes it relatively simple and useful.
- PCA is a good general purpose dimensionality reduction method, LDA is a good alternative if we want to optimize the separability of classes in a specific classification task, and are happy with dimensionality of less than the number of classes ($k < K$).

Linear Discriminant Analysis (LDA)

- Find a low-dimensional space such that when $\boldsymbol{x}$ is projected, classes are well-separated.
- Find $\boldsymbol{w}$ that maximizes

$$J(\boldsymbol{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$



$$m_1 = \frac{\sum_t \boldsymbol{w}^T \boldsymbol{x}^t r^t}{\sum_t r^t} \qquad s_1^2 = \sum_t \left(\boldsymbol{w}^T \boldsymbol{x}^t - m_1\right)^2 r^t$$

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

k-means Clustering

Lloyd's algorithm

```
LLOYDS($\mathcal{X}$,$k$) {Input: $\mathcal{X}$, data set; $k$, number of clusters. Output: $\{\mathbf{m}_i\}_{i=1}^k$, cluster prototypes.}
Initialize $\mathbf{m}_i$, $i = 1, \ldots, k$, appropriately for example, in random.
repeat
   for all $t \in \{1, \ldots, N\}$ do {E step}
      $$b_i^t \leftarrow \begin{cases} 1 & , \quad i = \arg\min_i \left|\left|\mathbf{x}^t - \mathbf{m}_i\right|\right| \\ 0 & , \quad \text{otherwise} \end{cases}$$
   end for
   for all $i \in \{1, \ldots, k\}$ do {M step}
      $$\mathbf{m}_i \leftarrow \frac{\sum_t b_i^t \mathbf{x}^t}{\sum_t b_i^t}$$
   end for
until the error $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X})$ does not change
return $\{\mathbf{m}_i\}_{i=1}^k$
```

k-means Clustering

Lloyd's algorithm

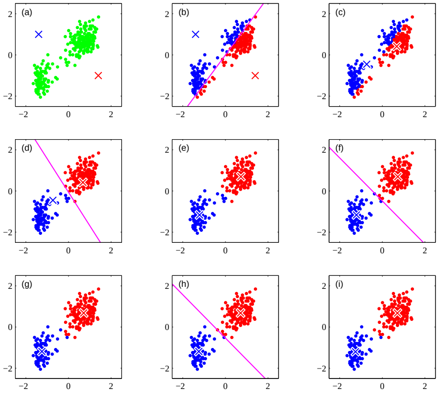
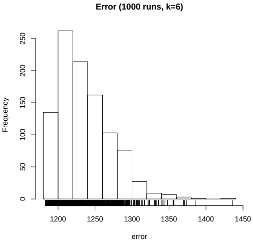


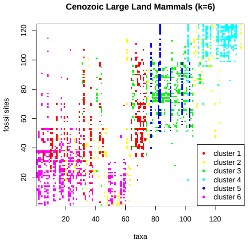
Figure 9.1 of Bishop (2006)

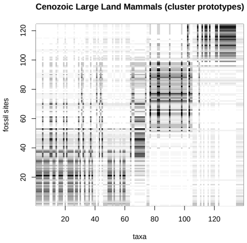
k-means Clustering

Lloyd's algorithm

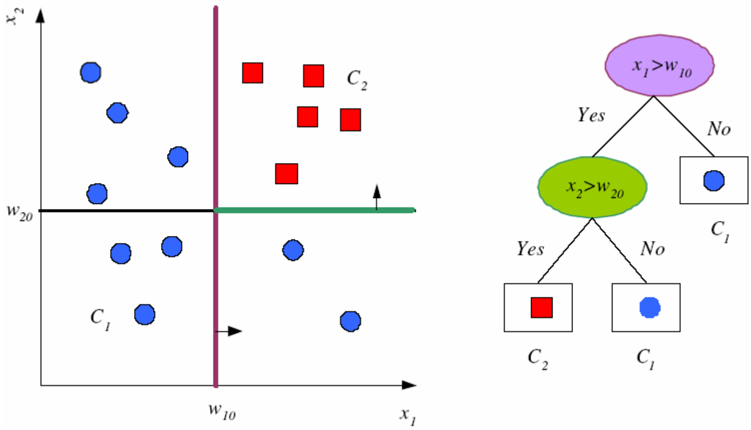
- Example: cluster taxa into $k = 6$ clusters 1000 times with Lloyd's algorithm.
- The error $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^{k} \mid \mathcal{X})$ is different for different runs!
- You should try several random initializations, and choose the solution with smallest error.
- For a cool initialization see Arthur D, Vassilivitskii S (2006) k-means++: The Advantages of Careful Seeding.







Decision Trees

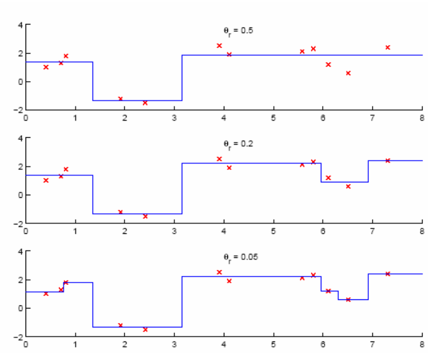


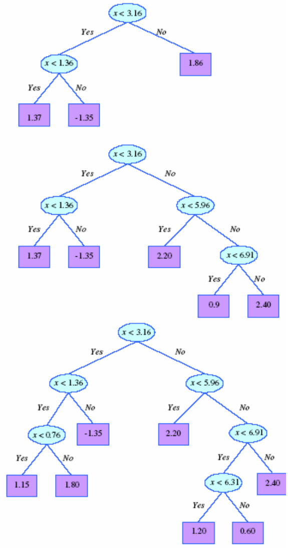
Lecture Notes for E Alpaydın 2004 Introduction to Machine Learning © The MIT Press (V1.1)

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** $T$
```

Model Selection in Trees:





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

Cost Function for Logistic Regression

- $$P(R \mid X, W) = \prod_{t=1}^{n} P(r^t \mid \mathbf{x}^t, W)$$
- $$\mathcal{L} = -\log P(R \mid X, W) = -\sum_{t=1}^{N} \left(r^t \log y^t - (1 - r^t) \log (1 - y^t) \right),$$

  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 etc. algorithm. Use gradient ascent.

Gradient Ascent

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

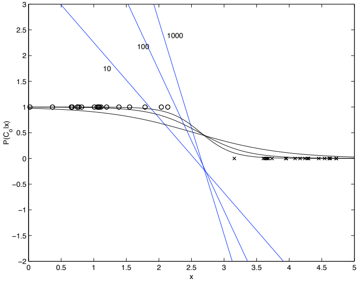


Figure 10.7: For a univariate two-class problem (shown with '$\circ$' and '$\times$'), the evolution of the line $wx + w_0$ and the sigmoid output after 10, 100, and 1,000 iterations over the sample. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

The End

(Some overflow slides follow.)

Outline

1. Announcements
   - Examination
   - Course Feedback
2. Summary of the Course
   - Summary of the Course
3. Overflow
   - Optimization Algorithms
   - Computing Sums and Products
   - Validation and Cross-Validation

Algorithms in Machine Learning

- Many (most?) machine learning algorithms problems can be stated as optimization problem: "Find parameters θ such that the cost $\mathcal{L}(\theta)$ is minimized."
- Earlier in the course:
  - Some optimization problems can be solved in polynomial time (e.g., PCA)
  - In some optimization problems (typically they are NP-hard) one must use approximation algorithms, such as greedy search. (e.g., Lloyd's algorithm in kmeans clustering).

Algorithms in Machine Learning

- Issues to take into account:
    - What is the time and memory complexity?
    - How is data accessed (for large data sets, serial access is fastest)
    - Does the algorithm find a reasonable solution (is there approximation ratio?)
    - Could there be a better greedy optimization step?
    - Is your algorithm numerically robust? (That is, does it work consistently and give accurate results for every possible input.)

Algorithms in Machine Learning

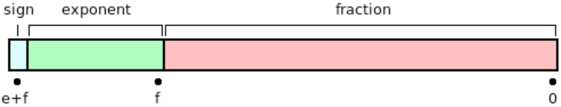
- Making numerically robust algorithms is difficult
- The first rule in numerical computation: always use robust numerical libraries when possible
- Of methods with essentially similar performance, choose the simplest/easiest to understand.

Outline

1. Announcements
   - Examination
   - Course Feedback
2. Summary of the Course
   - Summary of the Course
3. Overflow
   - Optimization Algorithms
   - Computing Sums and Products
   - Validation and Cross-Validation

IEEE Floating Point Arithmetics

- The floating point numbers are stored in three parts in binary:
  - fraction ($f = 52$ bits in double precision)
  - exponent ($e = 11$ bits in double precision)
  - sign (1 bit)
- This includes the following types of numbers:
  - normalized numbers (normal non-zero numbers)
  - zero (± 0)
  - infinities ($\pm\infty$)
  - NaN
  - denormalized numbers ($\pm$ something very small or very large)



The three fields in an IEEE 754 float. Image by Charles Esson, GFDL.

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)
```

Outline

1. Announcements
   - Examination
   - Course Feedback
2. Summary of the Course
   - Summary of the Course
3. Overflow
   - Optimization Algorithms
   - Computing Sums and Products
   - Validation and Cross-Validation

Evaluating Classification Algorithms

- Questions:
  - What is the performance of a classification algorithm on unseen data?
  - Which of the two (or more) classification algorithms is better?
- Our results are conditioned on the data set. (In fact, for all algorithms there exists data sets for which it would perform excellently or poorly, No Free Lunch Theorem, Wolpert 1995.)
- Limited amount of training/validation data makes it difficult
  - Choose the model complexity.
  - Evaluate the results.

K-Fold Cross-Validation

- How to use the training/validation data most efficiently?

```
CV($\mathcal{X}$,$A$,$K$){Input: $\mathcal{X}$, data $\mathcal{X} = \{(r^t, x^t)\}_{t=1}^N$; $A$, classification algorithm; $K$, number of folds. Output: $\mathcal{E}$, error measure.}
Partition $\mathcal{X}$ in random into $K$ roughly equally sized parts $\mathcal{X}_i$.
for all $i \in \{1, \ldots, K\}$ do
    Train $A$ using $\mathcal{X} \setminus \mathcal{X}_i$ as a training set.
    Let $\mathcal{E}_i$ be the error of $A$ in $\mathcal{X}_i$ (for example, the fraction of incorrectly labeled items).
end for
$\mathcal{E} \leftarrow \sum_{i=1}^K |\mathcal{X}_i| \mathcal{E}_i / |\mathcal{X}|$
return $\mathcal{E}${$\mathcal{E}$ can be used as a validation set error in model selection.}
```