DM825

Introduction to Machine Learning

Lecture 1
Introduction

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

1. Course Introduction
2. Introduction
3. Supervised Learning
   - Linear Regression
   - Nearest Neighbor

Outline

Machine Learning

ML is a branch of artificial intelligence and an interdisciplinary field of CS, statistics, math and engineering.
Applications in science, finance, industry:

- predict possibility for a certain disease on basis of clinical measures
- assess credit risk (default/non-default)
- identify numbers in ZIP codes
- identify risk factor for cancer based on clinical measures
- drive vehicles
- data bases in medical practice to extract knowledge
- spam filter
- costumer recommendations (eg, amazon)
- web search, fraud detection, stock trading, drug design

Automatically learn programs by generalizing from examples. As more data becomes available, more ambitious problems can be tackled.

Machine Learning vs Data Mining

- Machine learning (or predictive analytics) focuses on accuracy of prediction
  Data can be collected
- Data mining (or information re-trivial) focuses on efficiency of the algorithms since it mainly refer to big data.
  All data are given

However the terms can be used interchangeably

Aims of the course

- to convey excitement about the subject
- to learn about the state of the art methods
- to acquire skills to apply a ML algorithm, make it work and interpret the results
- to gain some bits of folk knowledge to make ML algorithms work well (developing successful machine learning applications requires a substantial amount of "black art" that is difficult to find in textbooks)

Schedule

- Schedule ($\approx$ 28 lecture hours + $\approx$ 14 exercise hours):
  - Monday, 08:15-10:00, IMADA seminarrum
  - Wednesday, 16:15-18:00, U49
  - Friday, 08.15-10:00, IMADA seminarrum
  - Last lecture: Friday, March 15, 2013

- Communication tools
  - Course Public Webpage (WWW) $\Leftrightarrow$ BlackBoard (BB)
    (link from http://www.imada.sdu.dk/~marco/DM825/)
  - Announcements in BlackBoard
  - Personal email
- Main reading material:
  - Pattern recognition and Machine Learning
    by C.M. Bishop. Springer, 2006
  - Lecture Notes by Andrew Ng, Stanford University
  - Slides

Contents

Supervised Learning: linear regression and linear models • gradient descent, Newton-Raphson (batch and sequential) • least squares method • k-nearest neighbor • curse of dimensionality • regularized least squares (aka, shrinkage or ridge regr.) • locally weighted linear regression • model selection • maximum likelihood approach • Bayesian approach

linear models for classification • logistic regression • multinomial (logistic) regression • generalized linear models • decision theory

neural networks • perceptron algorithm • multi-layer perceptrons

generative algorithms • Gaussian discriminant and linear discriminant analysis

kernels and support vector machines

probabilistic graphical models: naive Bayes • discrete • linear Gaussian • mixed variables • conditional independence • Markov random fields • Inference: exact, chains, polytree, approximate • hidden Markov models

bagging • boosting • tree based methods • learning theory

Unsupervised learning: Association rules • cluster analysis • k-means • mixture models • EM algorithm • principal components

Reinforcement learning: • MDPs • Bellman equations • value iteration and policy iteration • Q-learning • policy search • POMDPs.

Data mining: frequent pattern mining

Prerequisites

- Calculus (MM501, MM502)
- Linear Algebra (MM505)
- Probability calculus (random variables, expectation, variance)
  Discrete Methods (DM527) Science Statistics (ST501)
- Programming in R

Evaluation

- 5 ECTS
- course language: Danish and English
- obligatory Assignments, pass/fail, evaluation by teacher (2 hand in)
  practical part
- 3 hour written exam, 7-grade scale, external censor
  theory part
  similar to exercises in class

Assignments

- Small projects (in groups of 2) must be passed to attend the oral exam:
- Data set and guidelines will be provided but you can propose to work on different data (eg. www.kaggle.org)
- Entail programming in R

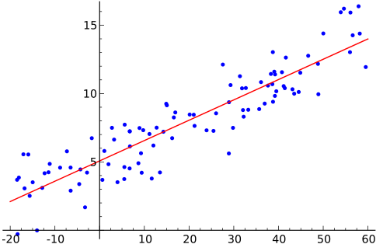
Exercises

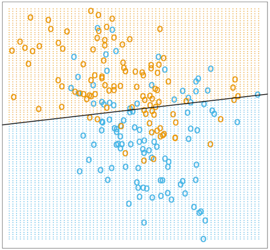
- Prepare for the exercise session revising the theory
- In class, you will work at the exercises in small groups

Outline

Supervised Learning

- inputs that influence outputs
  inputs: predictors, independent variables, features
  outputs: responses, dependent variables
- goal: predict value of outputs
- **supervised**: we provide data set with exact answers
- regression problem $\rightsquigarrow$ variable to predict is continuous/quantitative
- classification problem $\rightsquigarrow$ variable to predict is discrete/qualitative/categorical/factor





Other forms of learning

- unsupervised learning
- reinforcement learning: not one shot decision but sequence of decisions over time. (eg, elicopter fly)
  Reward function + maximize reward
- evolutionary learning: fitness, score

Learning theory: examples of analyses:

- guarantee that a learning algorithm can arrive at 99% with very large amount of data
- how much training data one needs

Notation

- $\vec{X}$ input vector, X_j the jth component
  (We use uppercase letters such as X, Y or G when referring to the generic aspects of a variable)
- $\vec{x}^i$ the ith observed value of $\vec{X}$
  (We use lowercase for observed values)
- Y, G outputs (G for for groups or quantitative outputs)
- $j = 1, \ldots, p$ for parameters and $i = 1, \ldots, m$ for observations
- $\mathbf{X} = \begin{bmatrix} x_1^1 & \ldots & x_p^1 \\ \vdots & & \\ x_1^m & \ldots & x_p^m \end{bmatrix}$ is a $m \times p$ matrix for a set of m input p-vectors
  $\vec{x}^i, i = 1, ..., m$
- $\mathbf{x}_j$ all observations on the variable X_j (column vector)

- Learning task: given the value of an input vector X, make a good prediction of the output Y, denoted by $\hat{Y}$.
- If $Y \in \mathbf{R}$ then $\hat{Y} \in \mathbb{R}$
  If $G \in \mathcal{G}$ then $\hat{G} \in \mathcal{G}$
- If $G \in \{0, 1\}$ then possible to encode as $Y \in [0, 1]$, then $\hat{G} = 0$ if $\hat{Y} < 0.5$ and $\hat{G} = 1$ if $\hat{Y} \geq 0.5$
- (x^i, y^i) or (x^i, g^i) are training data

Learning Task: Overview

Learning = Representation + Evaluation + optimization

- Representation: formal language that the computer can handle. Corresponds to choosing the set of functions that can be learned, ie. the hypothesis space of the learner. How to represent the input, that is, what features to use.
- Evaluation: an evaluation function (aka objective function or scoring function)
- Optimization. a method to search among the learners in the language for the highest-scoring one. Efficiency issues. Common for new learners to start out using off-the-shelf optimizers, which are later replaced by custom-designed ones.

Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
K-nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search
Hyperplanes	Likelihood	Branch-and-bound
Naive Bayes	Posterior probability	Continuous optimization
Logistic regression	Information gain	Unconstrained
Decision trees	K-L divergence	Gradient descent
Sets of rules	Cost/Utility	Conjugate gradient
Propositional rules	Margin	Quasi-Newton methods
Logic programs		Constrained
Neural networks		Linear programming
Graphical models		Quadratic programming
Bayesian networks		
Conditional random fields		

Outline

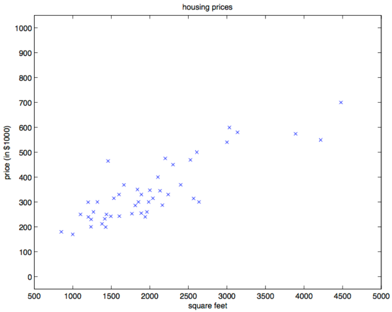
1. Course Introduction

2. Introduction

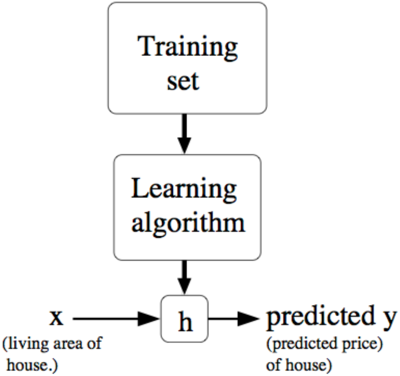
3. Supervised Learning
   - Linear Regression
   - Nearest Neighbor

Supervised Learning Problem

Living area ($feet^2$)	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮



Learning Task



Regression Problem

Living area ($feet^2$)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

Representation of hypothesis space:

$$h(x) = \theta_0 + \theta_1 x \qquad \text{linear function}$$

if we know another feature:

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = h_\theta(x)$$

for conciseness, defining $x_0 = 1$

$$h(x) = \sum_{j=0}^{2} \theta_j x_j = \vec{\theta}^T \vec{x}$$

p # of features, $\vec{\theta}$ vector of $p+1$ parameters, θ_0 si the bias

Evaluation

loss function $L(Y, h(X))$ for penalizing errors in prediction. Most common is squared error loss:

$$L(Y, h(X)) = (h(X) - Y)^2$$

this leads to minimize:

$$\min_{\vec{\theta}} L(\vec{\theta})$$

Optimization

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} \left(h_{\vec{\theta}}(\vec{x}^i) - y^i\right)^2 \qquad \text{cost function}$$

$$\min_{\vec{\theta}} J(\theta)$$

Parameter estimation

Learn by adjusting parameters to reduce error on training set
The squared error for an example with input $\vec{x}$ and true output y is
$J(\vec{\theta}) = \frac{1}{2}(h_{\vec{\theta}}(\vec{x}) - y)^2$

Find local optima for the minimization of the function $J(\vec{\theta})$ in the vector of variables $\vec{\theta}$ by gradient methods.

Gradient methods

Gradient methods are iterative approaches:

- find a descent direction with respect to the objective function J
- move $\vec{\theta}$ in that direction by a step size

The descent direction can be computed by various methods, such as gradient descent, Newton-Raphson method and others. The step size can be computed either exactly or loosely by solving a line search problem.

Example: gradient descent

1. Set iteration counter $t = 0$, and make an initial guess θ_0 for the minimum
2. Repeat:
3. Compute a descent direction $\vec{p}_t = \nabla(J(\vec{\theta}_t))$
4. Choose α_t to minimize $f(\alpha) = J(\vec{\theta}_t - \alpha\vec{p}_t)$ over $\alpha \in \mathbb{R}_+$
5. Update $\vec{\theta}_{t+1} = \vec{\theta}_t - \alpha_t\vec{p}_t$, and $t = t + 1$
6. Until $\|\nabla J(\vec{\theta}_k)\| < tolerance$

Step 4 can be solved 'loosely' by taking a fixed small enough value $\alpha > 0$

In our linear regression case the update rule of lines 3-5 for one single training example becomes:

$$\theta_j^{t+1} = \theta_j^t - \alpha \frac{\partial J(\vec{\theta})}{\partial \theta_j}$$

$$\begin{aligned}
\frac{\partial J(\vec{\theta})}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h_{\vec{\theta}}(\vec{x}) - y\right) \\
&= 2\frac{1}{2} \left(h_{\vec{\theta}}(\vec{x}) - y\right) \frac{\partial}{\partial \theta_j} \left(h_{\vec{\theta}}(\vec{x}) - y\right) \\
&= \left(h_{\vec{\theta}}(\vec{x}) - y\right) \frac{\partial}{\partial \theta_j} \left(\theta_0 + \theta_1 x_1 + \ldots + \theta_p x_p\right) \\
&= \left(h_{\vec{\theta}}(\vec{x}) - y\right) x_j
\end{aligned}$$

$$\theta_j^{t+1} = \theta_j^t - \alpha \left(h_{\vec{\theta}}(\vec{x}) - y\right) x_j$$

So far one single training example
For training set $\rightsquigarrow$ batch descent

repeat

$$\theta_j^{t+1} = \theta_j^t - \alpha \sum_{j=1}^m \left(h_{\vec{\theta}}(\vec{x}) - y\right) x_j$$

until until convergence ;

Stochastic gradient descent:

repeat

for $i = 1 \ldots m$ **do**

$$\theta_j^{t+1} = \theta_j^t - \alpha \left(h_{\vec{\theta}}(\vec{x}^i) - y^i\right) x_j^i$$

until until convergence ;

Implement them in R. Compare with the `optim` function and `grad.desc` from the package `animation`

Closed Form

The function $J(\theta)$ is a convex quadratic function. We can derive the gradient in closed form.
In matrix vectorial notation: design matrix and response vector

$$\mathbf{X} = \begin{bmatrix} \dots & (\vec{x}^1)^T & \dots \\ \dots & (\vec{x}^2)^T & \dots \\ & \vdots & \\ \dots & (\vec{x}^m)^T & \dots \end{bmatrix} \qquad \vec{y} = \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^m \end{bmatrix}$$

since $h_{\vec{\theta}}(\vec{x}^i) = (\vec{x}^i)^T\theta$ we have:

$$\mathbf{X}\theta - \vec{y} = \begin{bmatrix} (\vec{x}^1)^T\vec{\theta} \\ (\vec{x}^2)^T\vec{\theta} \\ \vdots \\ (\vec{x}^m)^T\vec{\theta} \end{bmatrix} - \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^m \end{bmatrix} = \begin{bmatrix} h_{\vec{\theta}}(\vec{x}^1) - y^1 \\ h_{\vec{\theta}}\vec{\theta}(\vec{x}^2) - y^2 \\ \vdots \\ h_{\vec{\theta}}(\vec{x}^m) - y^m \end{bmatrix}$$

for a vector $\vec{z}$ it is $\vec{z}^T\vec{z} = \sum_i z_i^2$

$$J(\vec{\theta}) = \frac{1}{2}\sum_{i=1}^{m}\left(h_{\vec{\theta}}(\vec{x}^i) - y^i\right)^2 = \frac{1}{2}(\mathbf{X}\vec{\theta} - \vec{y})^T(\mathbf{X}\theta - \vec{y})$$

to minimize J we solve $\nabla_{\vec{\theta}}J(\vec{\theta}) = 0$ with respect to $\vec{\theta}$

$$\begin{aligned}
\nabla_{\vec{\theta}}J(\vec{\theta}) &= \nabla_{\vec{\theta}}\frac{1}{2}(\mathbf{X}\theta - \vec{y})^T(\mathbf{X}\theta - \vec{y}) && \underset{f:\mathbb{R}^{m\times p}\to\mathbb{R}}{\nabla_A f(A)} \\
&= \frac{1}{2}\nabla_{\vec{\theta}}(\vec{\theta}^T\mathbf{X}^T\mathbf{X}\vec{\theta} - \mathbf{X}^T\vec{\theta}^T\vec{y} - \vec{y}^T\mathbf{X}\vec{\theta} + \vec{y}^T\vec{y}) \\
&= \frac{1}{2}\nabla_{\vec{\theta}}\mathrm{tr}(\vec{\theta}^T\mathbf{X}^T\mathbf{X}\vec{\theta} - \mathbf{X}^T\vec{\theta}^T\vec{y} - \vec{y}^T\mathbf{X}\vec{\theta} + \vec{y}^T\vec{y}) && \mathrm{tr}a = a \\
&= \frac{1}{2}\nabla_{\vec{\theta}}(\mathrm{tr}\vec{\theta}^T\mathbf{X}^T\mathbf{X}\vec{\theta} - 2\mathrm{tr}\mathbf{X}^T\vec{\theta}^T\vec{y}) && \mathrm{tr}A = A^T \\
&= \frac{1}{2}(\mathbf{X}^T\mathbf{X}\vec{\theta} + \mathbf{X}^T\mathbf{X}\vec{\theta} - 2\mathbf{X}^T\vec{y}) \\
&= \mathbf{X}^T\mathbf{X}\vec{\theta} - \mathbf{X}^T\vec{y} = 0
\end{aligned}$$

$$\mathbf{X}^T\mathbf{X}\vec{\theta} = \mathbf{X}^T\vec{y} \qquad \rightsquigarrow \qquad \vec{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\vec{y}$$

k nearest neighbor

- **Regression**

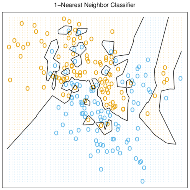
$$\hat{y}(x) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(x)} y_i$$

average of the k closest points
concept of closeness requires the definition of a metric, eg, Euclidean distance

- **Classification**

$$\hat{G} = \begin{cases} 1 & \text{if } \hat{y} > 0.5 \\ 0 & \text{if } \hat{y} \leq 0.5 \end{cases}$$

corresponds to a majority rule

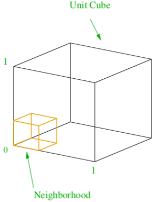


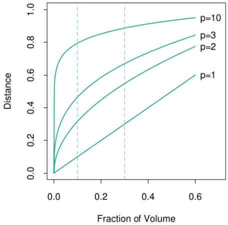
$k = 1$ predict the response of the point in the training set closest to $\vec{x}$. (Vornoi tassellation)

Remark: on training data the error increases with k, while for $k = 1$ it is zero

Curse of Dimensionality

- k-nearest neighbor in p-dim.
  points uniformly distributed in a p-dim hypercube.
  We want to capture r observations in a hypercube neighbor $\rightsquigarrow$
  corresponds to a fraction r of unit volume
  expected edge length: $e_p(r) = r^{\frac{1}{p}}$
  $p = 10$ $\quad e_{10}(0,01) = 0.63$ 1% of data must cover 63% of volume
  $\quad e_{10}(0,1) = 0.80$ 10% of data must cover 80% of volume





- Sampling density is proportional to $m^{\frac{1}{p}}$
  Thus if $m = 100$ is a dense sample in 1 dimension
  a sample with the same density in 10 dimensions needs $m = 100^{10}$

Resume

- linear models for regression (how can it be used for non linear patterns in data?)
- k-nearest neighbor (for regression and classification)
- curse of dimensionality
  the directions over which important variations in the target variables arise maybe confined
  local interpolation-like techniques help us in making predictions on new values