CSC411
Machine Learning

Russ Salakhutdinov
Department of Computer Science
Department of Statistics
rsalakhu@cs.toronto.edu
http://www.cs.toronto.edu/~rsalakhu/

Lecture 1

Some slides are borrowed from Rich Zemel
Admin Details

• Liberal wrt waiving pre-requisites: but it is up to you to determine if you have the appropriate background

• Tutorials:
  – Thursdays, 8-9pm

• Do I have the appropriate background?
  – Linear algebra: vector/matrix manipulations, properties
  – Calculus: partial derivatives
  – Probability: common distributions; Bayes Rule
  – Statistics: mean/median/mode; maximum likelihood

  – Sheldon Ross: A First Course in Probability
Textbooks

• Christopher Bishop:
  – ”Pattern Recognition and Machine Learning”.

• Ethem Alpaydin:

• Other recommended texts
  – Kevin Murphy: Machine Learning: a Probabilistic Perspective
  – David Mackay: Information Theory, Inference, and Learning Algorithms
Requirements

• Do the readings!

• Assignments
  – Three assignments, worth 40%
  – Programming: take Matlab/Python code and extend it
  – Derivations: pen(cil)-and-paper

• Mid-term
  – One hour exam
  – Worth 25% of course mark

• Final
  – Focus on second half of course
  – Worth 35% of course mark
What is Machine Learning?

• Learning systems are not directly programmed to solve a problem, instead develop own program based on:
  – Examples of how they should behave
  – From trial-and-error experience trying to solve the problem

• Different than standard CS: want to implement unknown function, only have access to sample input-output pairs (training examples)

• Learning simply means incorporating information from the training examples into the system
Mining for Structure

Massive increase in both computational power and the amount of data available from web, video cameras, laboratory measurements.

Develop statistical models that can discover underlying structure, cause, or statistical correlations from data.
Example: Boltzmann Machine

\[ P(x, y) = \frac{1}{Z} \sum_h \exp \left[ x^\top W^{(1)} h + y^\top W^{(2)} h \right] \]

Input data (e.g. pixel intensities of an image, words from webpages, speech signal).

Target variables (response) (e.g. class labels, categories, phonemes).

Model parameters

Latent (hidden) variables

Markov Random Fields, Undirected Graphical Models.
Finding Structure in Data

\[ P(x) = \frac{1}{Z} \sum_h \exp [x^\top W_h] \]

Vector of word counts on a webpage

Latent variables: hidden topics

804,414 newswire stories
Matrix Factorization

Hierarchical Bayesian Model

Rating value of user $i$ for item $j$ 
Latent user feature (preference) vector 
Latent item feature vector

$$r_{ij} \mid u_i, v_j, \sigma \sim \mathcal{N}(u_i^T v_j, \sigma^2),$$

$$u_i \mid \sigma_u \sim \mathcal{N}(0, \sigma_u^2 I), \quad i = 1, \ldots, N.$$  
$$v_j \mid \sigma_v \sim \mathcal{N}(0, \sigma_v^2 I), \quad j = 1, \ldots, M.$$  

Latent variables that we infer from observed ratings.

Prediction: predict a rating $r_{ij}^*$ for user $i$ and query movie $j$.

$$P(r_{ij}^* \mid R) = \int \int P(r_{ij}^* \mid u_i, v_j) P(u_i, v_j \mid R) du_i dv_j$$

Infer latent variables and make predictions using Markov chain Monte Carlo.
Finding Structure in Data

Collaborative Filtering/
Matrix Factorization/
Product Recommendation

Netflix dataset:
480,189 users
17,770 movies
Over 100 million ratings.

Learned ``genre''

Fahrenheit 9/11
Bowling for Columbine
The People vs. Larry Flynt
Canadian Bacon
La Dolce Vita

Independence Day
The Day After Tomorrow
Con Air
Men in Black II
Men in Black

Friday the 13th
The Texas Chainsaw Massacre
Children of the Corn
Child's Play
The Return of Michael Myers

• Part of the winning solution in the Netflix contest (1 million dollar prize).
Impact of Machine Learning

- Speech Recognition
- Computer Vision
- Recommender Systems
- Language Understanding
- Drug Discovery and Medical Image Analysis
Multimodal Data

mosque, tower, building, cathedral, dome, castle

kitchen, stove, oven, refrigerator, microwave

ski, skiing, skiers, skiers, snowmobile

bowl, cup, soup, cups, coffee

beach

snow
Example: Understanding Images

TAGS:
strangers, coworkers, conventioneers, attendants, patrons

Nearest Neighbor Sentence:
people taking pictures of a crazy person

Model Samples
• a group of people in a crowded area.
• a group of people are walking and talking.
• a group of people, standing around and talking.
• a group of people that are in the outside.
Caption Generation

- A car is parked in the middle of nowhere.
- A wooden table and chairs arranged in a room.
- There is a cat sitting on a shelf.
- A ferry boat on a marina.
- A little boy with a group of people on the street.
Speech Recognition

According to Microsoft’s speech group:

Using DL
Merck Molecular Activity Challenge

- To develop new medicines, it is important to identify molecules that are highly active toward their intended targets.

- Deep Learning technique: Predict biological activities of different molecules, given numerical descriptors generated from their chemical structures.

Dahl et.al., 2014
Netflix uses:

- Restricted Boltzmann machines
- Probabilistic Matrix Factorization

• From their blog:

To put these algorithms to use, we had to work to overcome some limitations, for instance that they were built to handle 100 million ratings, instead of the more than 5 billion that we have, and that they were not built to adapt as members added more ratings. But once we overcame those challenges, we put the two algorithms into production, where they are still used as part of our recommendation engine.
Types of Learning

Consider observing a series of input vectors:

\[ x_1, x_2, x_3, x_4, \ldots \]

• **Supervised Learning**: We are also given target outputs (labels, responses): \( y_1, y_2, \ldots \), and the goal is to predict correct output given a new input.

• **Unsupervised Learning**: The goal is to build a statistical model of \( x \), which can be used for making predictions, decisions.

• **Reinforcement Learning**: the model (agent) produces a set of actions: \( a_1, a_2, \ldots \) that affect the state of the world, and received rewards \( r_1, r_2, \ldots \). The goal is to learn actions that maximize the reward (we will not cover this topic in this course).

• **Semi-supervised Learning**: We are given only a limited amount of labels, but lots of unlabeled data.
Supervised Learning

**Classification**: target outputs $y_i$ are discrete class labels. The goal is to correctly classify new inputs.

**Regression**: target outputs $y_i$ are continuous. The goal is to predict the output given new inputs.
Handwritten Digit Classification
Unsupervised Learning

The goal is to construct statistical model that finds useful representation of data:

- Clustering
- Dimensionality reduction
- Modeling the data density
- Finding hidden causes (useful explanation) of the data

Unsupervised Learning can be used for:

- Structure discovery
- Anomaly detection / Outlier detection
- Data compression, Data visualization
- Used to aid classification/regression tasks
DNA Microarray Data

Expression matrix of 6830 genes (rows) and 64 samples (columns) for the human tumor data.

The display is a heat map ranging from bright green (under expressed) to bright red (over expressed).

Questions we may ask:
• Which samples are similar to other samples in terms of their expression levels across genes.
• Which genes are similar to each other in terms of their expression levels across samples.
Why use learning?

• It is very hard to write programs that solve problems like recognizing a handwritten digit
  – What distinguishes a 2 from a 7?
  – How does our brain do it?
• Instead of writing a program by hand, we collect examples that specify the correct output for a given input
• A machine learning algorithm then takes these examples and produces a program that does the job
  – The program produced by the learning algorithm may look very different from a typical hand-written program. It may contain millions of numbers.
  – If we do it right, the program works for new cases as well as the ones we trained it on.
Two classic examples of tasks that are best solved by using a learning algorithm.
Learning algorithms are useful in other tasks

• Recognizing patterns:
  – Facial identities, expressions
  – Handwritten or spoken words

• Digital images and videos:
  – Locating, tracking, and identifying objects
  – Driving a car

• Recognizing anomalies:
  – Unusual sequences of credit card transactions

• Spam filtering, fraud detection:
  – The enemy adapts so we must adapt too

• Recommendation systems:
  – Noisy data, commercial pay-off (Amazon, Netflix).

• Information retrieval:
  – Find documents or images with similar content
Can you pick out the tufas?

Josh Tenenbaum
Machine Learning & Data Mining

- **Data-mining**: Typically using very simple machine learning techniques on very large databases because computers are too slow to do anything more interesting with ten billion examples.

- Previously used in a negative sense – misguided statistical procedure of looking for all kinds of relationships in the data until finally find one.

- Now lines are blurred: many ML problems involve tons of data.

- But problems with AI flavor (e.g., recognition, robot navigation) still domain of ML.
Machine Learning & Statistics

• ML uses statistical theory to build models – core task is inference from a sample

• A lot of ML is rediscovery of things statisticians already knew; often disguised by differences in terminology

• But the emphasis is very different:
  – Good piece of statistics: Clever proof that relatively simple estimation procedure is asymptotically unbiased.
  – Good piece of ML: Demo that a complicated algorithm produces impressive results on a specific task.

• Can view ML as applying computational techniques to statistical problems. But go beyond typical statistics problems, with different aims (speed vs. accuracy).
Cultural gap (Tibshirani)

Machine Learning-------------------------------Statistics

• network, graphs
• weights
• learning
• generalization
• supervised learning

• unsupervised learning.

• large grant: $1,000,000
• conference location: Snowbird, French Alps

• model
• parameters
• fitting
• test set performance
• regression/classification

• density estimation, clustering
• large grant: $50,000
• conference location: Las Vegas in August
Initial Case Study

- What grade will I get in this course?

- Data: entry survey and marks from previous years

- Process the data
  - Split into training set; test set
  - Determine representation of input features; output

- Choose form of model: linear regression

- Decide how to evaluate the system’s performance: objective function

- Set model parameters to optimize performance

- Evaluate on test set: generalization
Outline

• Linear regression problem
  – continuous outputs
  – simple model

• Introduce key concepts:
  – loss functions
  – generalization
  – optimization
  – model complexity
  – regularization
Very simple example: 1-D regression

Green shows the true curve – not known

The data points are uniform in $x$ but may be displaced in $y$

$$t(x) = f(x) + \varepsilon$$

Aim: fit a curve to these points

Key questions:
- How do we parametrize the model?
- What loss (objective) function should we use to judge fit?
- How do we optimize fit to unseen test data (generalization)?
Example: Boston Housing data

- Estimate median house price in a neighborhood based on neighborhood statistics

- Look at first (of 13) attributes: per capita crime rate

- Use this to predict house prices in other neighborhoods
Represent the Data

Data described as pairs $D = ((x^{(1)}, t^{(1)}), (x^{(2)}, t^{(2)}), \ldots, (x^{(N)}, t^{(N)}))$
- $x$ is the input feature (per capita crime rate)
- $t$ is the target output (median house price)

• Here $t$ is continuous, so this is a regression problem

Model outputs $y$, an estimate of $t$

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

Could take first 300 examples as training set, remaining 206 as test set
- Use the training examples to construct hypothesis, or function approximator, that maps $x$ to predicted $y$
- Evaluate hypothesis on test set
Noise

A simple model typically does not exactly fit the data – lack of fit can be considered noise

Sources of noise

– Imprecision in data attributes (input noise)

– Errors in data targets (mis-labeling)

– Additional attributes not taken into account by data attributes, affect target values (latent variables)

– Model may be too simple to account for data targets
Least-squares Regression

- Standard loss/cost/objective function measures the squared error in $y$ [the prediction of $t(x)$] from $x$.

\[ J(w) = \sum_{n=1}^{N} \left[ t^{(n)} - (w_0 + w \cdot x^{(n)}) \right]^2 \]

- The loss for the red hypothesis is the sum of the squared vertical errors.

from Bishop
Optimizing the Objective

- One straightforward method: initialize $w$ randomly, repeatedly update based on gradient descent in $J$

$$w \leftarrow w - \lambda \frac{\partial J}{\partial w}$$

- Here $\lambda$ is the learning rate

- For a single training case, this gives the LMS update rule:

$$w \leftarrow w + 2\lambda(t^{(n)} - y(x^{(n)}))x^{(n)}$$

- Note: as error approaches zero, so does update
Optimizing Across the Training Set

Two ways to generalize this for all examples in training set:

1. Stochastic/online updates – update the parameters for each training case in turn, according to its own gradients

2. Batch updates: sum or average updates across every example $i$, then change the parameter values

$$ w \leftarrow w + 2\lambda \sum_{n=1}^{N} (t^{(n)} - y(x^{(n)}))x^{(n)} $$

- Underlying assumption: sample is independent and identically distributed (i.i.d.)
When is minimizing the squared error equivalent to Maximum Likelihood Learning?

Minimizing the squared residuals is equivalent to maximizing the log probability of the correct answer under a Gaussian centered at the model's guess.

\[ y^{(n)} = y(x^{(n)}, w) \]

\[
p(t^{(n)} | y^{(n)}) = p(y^{(n)} + \text{noise} = t^{(n)} | x^{(n)}, w) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(t^{(n)} - y^{(n)})^2}{2\sigma^2}}
\]

\[-\log p(t^{(n)} | y^{(n)}) = \log \sqrt{2\pi} + \log \sigma + \frac{(t^{(n)} - y^{(n)})^2}{2\sigma^2}\]

- can be ignored if sigma is fixed
- can be ignored if sigma is same for every case
Linear Least Squares

• Given a vector of d-dimensional inputs $\mathbf{x} = (x_1, x_2, ..., x_d)^T$, we want to predict the target (response) using the linear model:

$$y(x, \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + ... + w_d x_d = w_0 + \sum_{j=1}^{d} w_j x_j.$$

• The term $w_0$ is the intercept, or often called bias term. It will be convenient to include the constant variable 1 in $\mathbf{x}$ and write:

$$y(x, \mathbf{w}) = \mathbf{x}^T \mathbf{w}.$$

• Observe a training set consisting of N observations $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)^T$, together with corresponding target values $\mathbf{t} = (t_1, t_2, ..., t_N)^T$.

• Note that $\mathbf{X}$ is an $N \times (d + 1)$ matrix.
Linear Least Squares

One option is to minimize the sum of the squares of the errors between the predictions \( y(x_n, w) \) for each data point \( x_n \) and the corresponding real-valued targets \( t_n \).

Loss function: sum-of-squared error function:

\[
E(w) = \frac{1}{2} \sum_{n=1}^{N} (x_n^T w - t_n)^2
\]

\[
= \frac{1}{2} (Xw - t)^T (Xw - t).
\]
Linear Least Squares

If $X^TX$ is nonsingular, then the unique solution is given by:

$$w^* = (X^TX)^{-1}X^Tt$$

- At an arbitrary input $x_0$, the prediction is $y(x_0, w) = x_0^Tw^*$.
- The entire model is characterized by $d+1$ parameters $w^*$.

Example: Polynomial Curve Fitting

Consider observing a training set consisting of N 1-dimensional observations:
\[ x = (x_1, x_2, ..., x_N)^T, \]
together with corresponding real-valued targets:
\[ t = (t_1, t_2, ..., t_N)^T. \]

- The green plot is the true function \( \sin(2\pi x) \).
- The training data was generated by taking \( x_n \) spaced uniformly between \([0 \ 1]\).
- The target set (blue circles) was obtained by first computing the corresponding values of the sin function, and then adding a small Gaussian noise.

Goal: Fit the data using a polynomial function of the form:
\[
y(x, w) = w_0 + w_1 x + w_2 x^2 + ... + w_M x^M = \sum_{j=0}^{M} w_j x^j.
\]

Note: the polynomial function is a nonlinear function of \( x \), but it is a linear function of the coefficients \( w \) \( \rightarrow \) Linear Models.
Example: Polynomial Curve Fitting

As for the least squares example: we can minimize the sum of the squares of the errors between the predictions $y(x_n, w)$ for each data point $x_n$ and the corresponding target values $t_n$.

Similar to the linear least squares: Minimizing sum-of-squared error function has a unique solution $w^*$.

The model is characterized by $M+1$ parameters $w^*$.

How do we choose $M$? → Model Selection.

Loss function: sum-of-squared error function:

$$E(w) = \frac{1}{2} \sum_{i=1}^{N} (y(x_n, w) - t_n)^2.$$
Some Fits to the Data

For $M=9$, we have fitted the training data perfectly.
Overfitting

• Consider a separate test set containing 100 new data points generated using the same procedure that was used to generate the training data.

• For M=9, the training error is zero → The polynomial contains 10 degrees of freedom corresponding to 10 parameters \( w \), and so can be fitted exactly to the 10 data points.

• However, the test error has become very large. Why?
Overfitting

As $M$ increases, the magnitude of coefficients gets larger.

For $M=9$, the coefficients have become finely tuned to the data.

Between data points, the function exhibits large oscillations.

More flexible polynomials with larger $M$ tune to the random noise on the target values.
Varying the Size of the Data

9th order polynomial

- For a given model complexity, the overfitting problem becomes less severe as the size of the dataset increases.

- However, the number of parameters is not necessarily the most appropriate measure of the model complexity.
Generalization

• The goal is achieve good **generalization** by making accurate predictions for new test data that is not known during learning.

• Choosing the values of parameters that minimize the loss function on the training data may not be the best option.

• We would like to model the true regularities in the data and ignore the noise in the data:
  - It is hard to know which regularities are real and which are accidental due to the particular training examples we happen to pick.

  • **Intuition:** We expect the model to generalize if it explains the data well given the complexity of the model.
  • If the model has as many degrees of freedom as the data, it can fit the data perfectly. But this is not very informative.
  • Some theory on how to control model complexity to optimize generalization.
A Simple Way to Penalize Complexity

One technique for controlling over-fitting phenomenon is **regularization**, which amounts to adding a penalty term to the error function.

The penalized error function is given by:

\[
\widetilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \left\{ y(x_n, \mathbf{w}) - t_n \right\}^2 + \frac{\lambda}{2} \| \mathbf{w} \|^2
\]

where \( \| \mathbf{w} \| = \mathbf{w}^T \mathbf{w} = w_1^2 + w_2^2 + \ldots + w_M^2 \) called the regularization term.

Note that we do not penalize the bias term \( w_0 \).

- The idea is to “shrink” estimated parameters towards zero (or towards the mean of some other weights).
- Shrinking to zero: penalize coefficients based on their size.
- For a penalty function which is the sum of the squares of the parameters, this is known as a “weight decay”, or “ridge regression”.

\[ \text{ln} \lambda = -18 \]
Regularization

Graph of the root-mean-squared training and test errors vs. ln\(\lambda\) for the M=9 polynomial.

How to choose \(\lambda\)?
Cross Validation

If the data is plentiful, we can divide the dataset into three subsets:

- **Training Data**: used to fitting/learning the parameters of the model.
- **Validation Data**: not used for learning but for selecting the model, or choosing the amount of regularization that works best.
- **Test Data**: used to get performance of the final model.

For many applications, the supply of data for training and testing is limited. To build good models, we may want to use as much training data as possible. If the validation set is small, we get noisy estimate of the predictive performance.

S fold cross-validation

- The data is partitioned into S groups.
- Then S-1 of the groups are used for training the model, which is evaluated on the remaining group.
- Repeat procedure for all S possible choices of the held-out group.
- Performance from the S runs are averaged.
Probabilistic Perspective

• So far we saw that polynomial curve fitting can be expressed in terms of error minimization. We now view it from probabilistic perspective.

• Suppose that our model arose from a statistical model:

\[ t = y(x, w) + \epsilon, \]

where \( \epsilon \) is a random error having Gaussian distribution with zero mean, and is independent of \( x \).

Thus we have:

\[ p(t|x, w, \beta) = \mathcal{N}(t|y(x, w), \beta^{-1}), \]

where \( \beta \) is a precision parameter, corresponding to the inverse variance.

I will use probability distribution and probability density interchangeably. It should be obvious from the context.
Maximum Likelihood

If the data are assumed to be independently and identically distributed (i.i.d assumption), the likelihood function takes form:

\[
p(t|x, w, \beta) = \prod_{i=1}^{N} \mathcal{N}(t_n | y(x_n, w), \beta^{-1}).
\]

It is often convenient to maximize the log of the likelihood function:

\[
\ln p(t|x, w, \beta) = -\frac{\beta}{2} \sum_{n=1}^{N} (y(x_n, w) - t_n)^2 + \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi). 
\]

\[= \beta E(w) - \frac{\beta}{2} \sum_{n=1}^{N} (y(x_n, w) - t_n)^2 \]

• Maximizing log-likelihood with respect to \(w\) (under the assumption of a Gaussian noise) is equivalent to minimizing the sum-of-squared error function.

• Determine \(w_{ML}\) by maximizing log-likelihood. Then maximizing w.r.t. \(\beta\):

\[
\frac{1}{\beta_{ML}} = \frac{1}{N} \sum_{n} (y(x_n, w_{ML}) - t_n)^2.
\]
Predictive Distribution

Once we determined the parameters $\mathbf{w}$ and $\beta$, we can make prediction for new values of $\mathbf{x}$:

$$p(t|\mathbf{x}, \mathbf{w}_{ML}, \beta_{ML}) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}_{ML}), \beta_{ML}^{-1}).$$