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:
  - ”Introduction to Machine Learning”, 2nd edition, 2010.

- 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

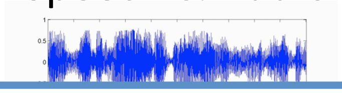
Mining for Structure

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

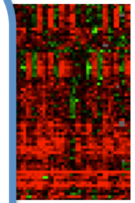
Images & Video

Text & Language

Speech & Audio



Gene Expression

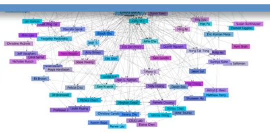


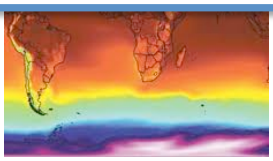
Prod

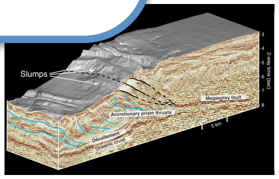
Reco

al Data

Develop statistical models that can discover underlying structure, cause, or statistical correlations from data.







Example: Boltzmann Machine

Model parameters

Latent (hidden) variables

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

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

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

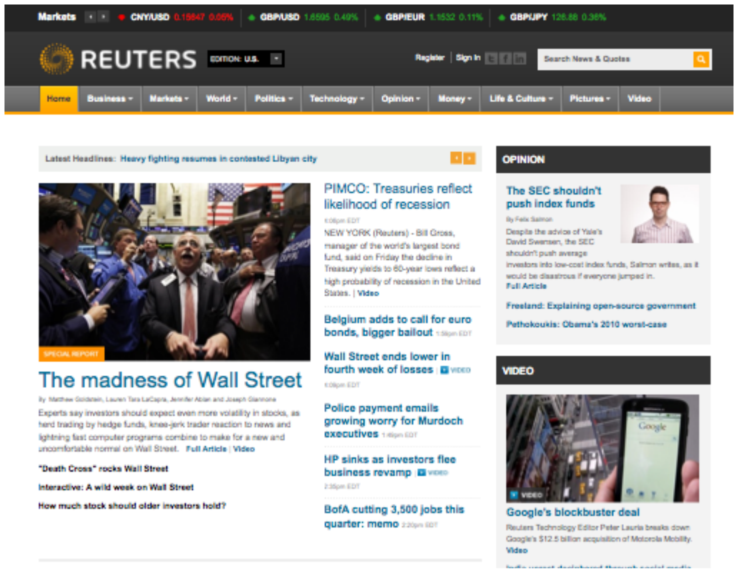
Markov Random Fields, Undirected Graphical Models.

Finding Structure in Data

$$P(\mathbf{x}) = \frac{1}{\mathcal{Z}} \sum_{\mathbf{h}} \exp\left[\mathbf{x}^\top \mathbf{W} \mathbf{h}\right]$$

Vector of word counts on a webpage

Latent variables: hidden topics



Latest Headlines: Heavy fighting resumes in contested Libyan city

The madness of Wall Street

By Matthew Goldstein, Lauren Tara LaCapra, Jennifer Ablan and Joseph Giannone

Experts say investors should expect even more volatility in stocks, as herd trading by hedge funds, knee-jerk trader reaction to news and lightning fast computer programs combine to make for a new and uncomfortable normal on Wall Street. Full Article | Video

"Death Cross" rocks Wall Street

Interactive: A wild week on Wall Street

How much stock should older investors hold?

PIMCO: Treasuries reflect likelihood of recession

4:08pm EDT

NEW YORK (Reuters) - Bill Gross, manager of the world's largest bond fund, said on Friday the decline in Treasury yields to 60-year lows reflect a high probability of recession in the United States. | Video

Belgium adds to call for euro bonds, bigger bailout 1:58pm EDT

Wall Street ends lower in fourth week of losses | VIDEO

4:08pm EDT

Police payment emails growing worry for Murdoch executives 1:49pm EDT

HP sinks as investors flee business revamp | VIDEO

2:35pm EDT

BofA cutting 3,500 jobs this quarter: memo 2:20pm EDT

The SEC shouldn't push index funds

By Felix Salmon

Despite the advice of Yale's David Swensen, the SEC shouldn't push average investors into low-cost index funds, Salmon writes, as it would be disastrous if everyone jumped in.

Full Article

Freeland: Explaining open-source government

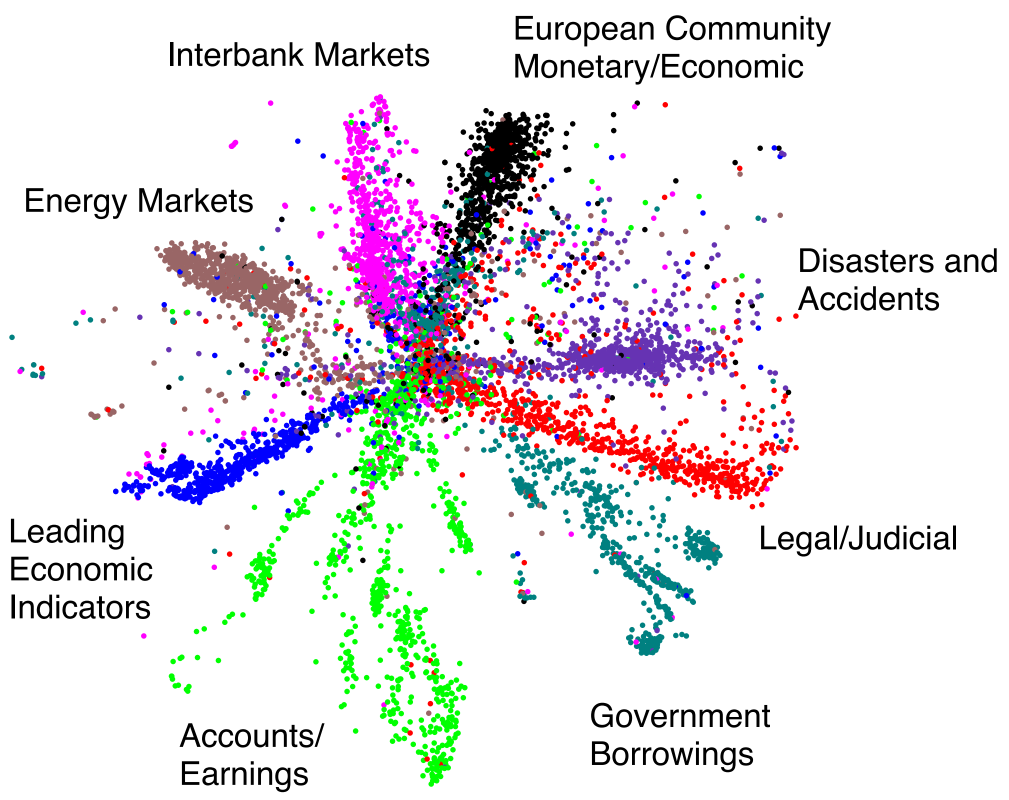
Pethokoukis: Obama's 2010 worst-case

Google's blockbuster deal

Reuters Technology Editor Peter Lauria breaks down Google's $12.5 billion acquisition of Motorola Mobility.

Video

804,414 newswire stories



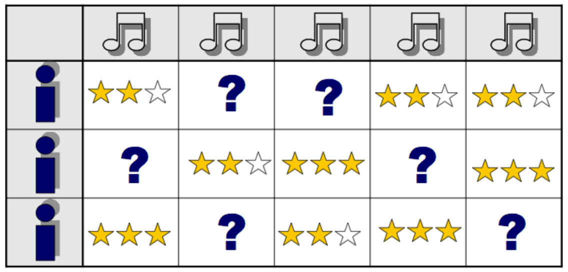
Matrix Factorization

Collaborative Filtering/
Matrix Factorization/

NETFLIX

movielens
helping you find the *right* movies

amazon



	♫	♫	♫	♫	♫
ℹ	★★☆	?	?	★★☆	★★☆
ℹ	?	★★☆	★★★	?	★★★
ℹ	★★★	?	★★☆	★★★	?

Hierarchical Bayesian Model

Rating value of user i for item j

Latent user feature (preference) vector

Latent item feature vector

$$r_{ij}|\mathbf{u}_i, \mathbf{v}_j, \sigma \sim \mathcal{N}(\mathbf{u}_i^\top \mathbf{v}_j, \sigma^2),$$

$$\left.\begin{array}{lcll} \mathbf{u}_i|\sigma_u & \sim & \mathcal{N}(\mathbf{0}, \sigma_u^2 \mathbf{I}), & i = 1, ..., N. \\ \mathbf{v}_j|\sigma_v & \sim & \mathcal{N}(\mathbf{0}, \sigma_v^2 \mathbf{I}), & j = 1, ..., M. \end{array}\right\}$$

Latent variables that we infer from observed ratings.

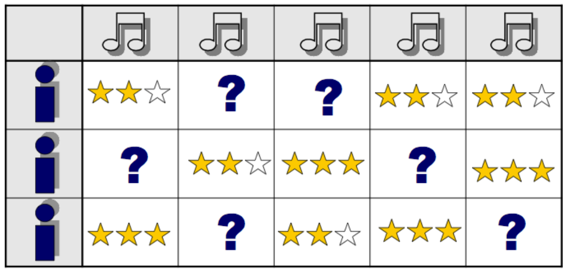
Prediction: predict a rating r^*_{ij} for user i and query movie j.

$$P(r^*_{ij}|\mathbf{R}) = \iint P(r^*_{ij}|\mathbf{u}_i, \mathbf{v}_j) \underbrace{P(\mathbf{u}_i, \mathbf{v}_j|\mathbf{R})}_{\textbf{Posterior over Latent Variables}} d\mathbf{u}_i d\mathbf{v}_j$$

Infer latent variables and make predictions using Markov chain Monte Carlo.

Finding Structure in Data

Collaborative Filtering/
Matrix Factorization/
Product Recommendation



Learned ``genre''

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

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

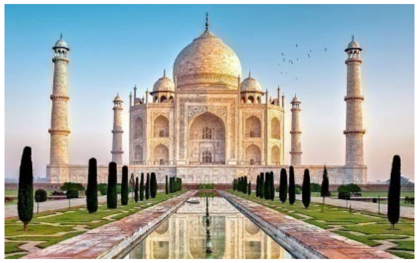
Friday the 13th
The Texas Chainsaw Massacre
Children of the Corn
Child's Play
The Return of Michael Myers

- Part of the wining 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

ski, skiing, skiers, skiiers, snowmobile

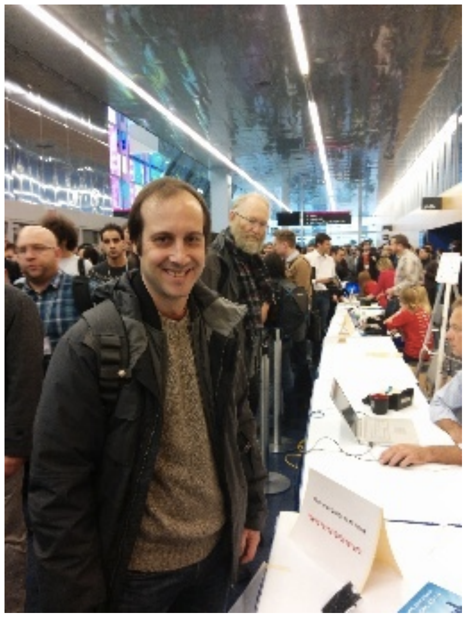
kitchen, stove, oven, refrigerator, microwave

bowl, cup, soup, cups, coffee

beach

snow

Example: Understanding Images



TAGS:

strangers, coworkers, conventioneers, attendants, patrons

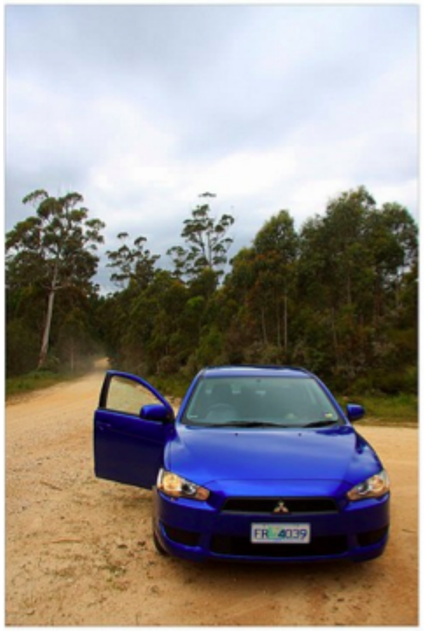
Nearest Neighbor Sentence:

people taking pictures of a crazy person

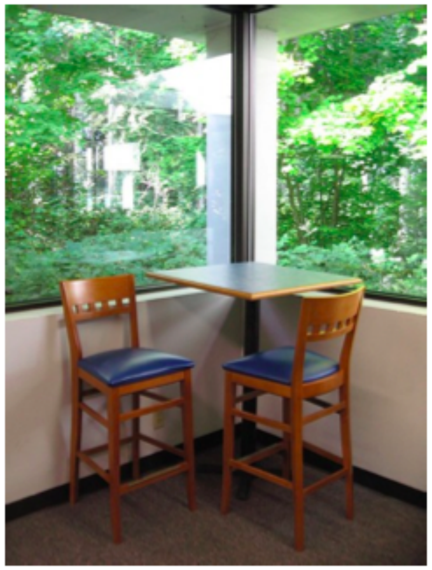
Model Samples

- 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 .

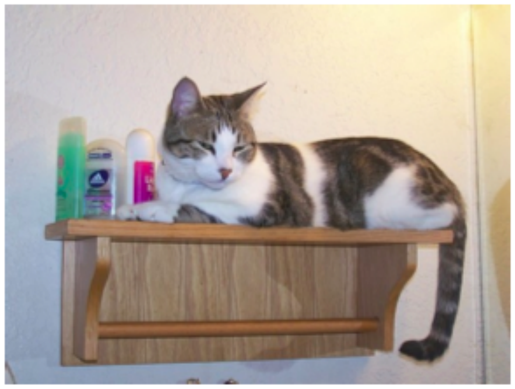
Caption Generation



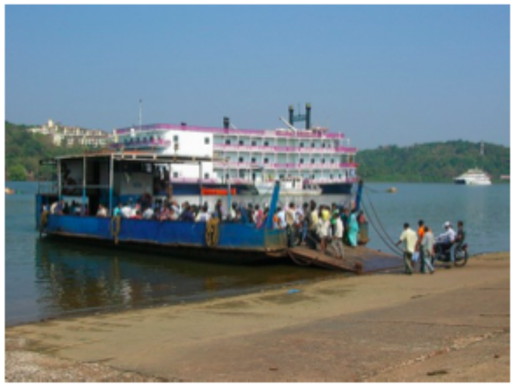
a car is parked in
the middle of nowhere .



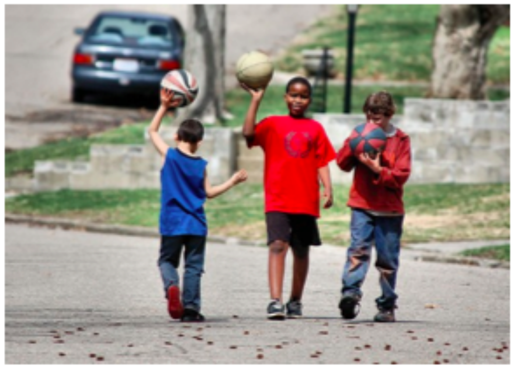
a wooden table and chairs
arranged in a room .



there is a cat sitting on a shelf .

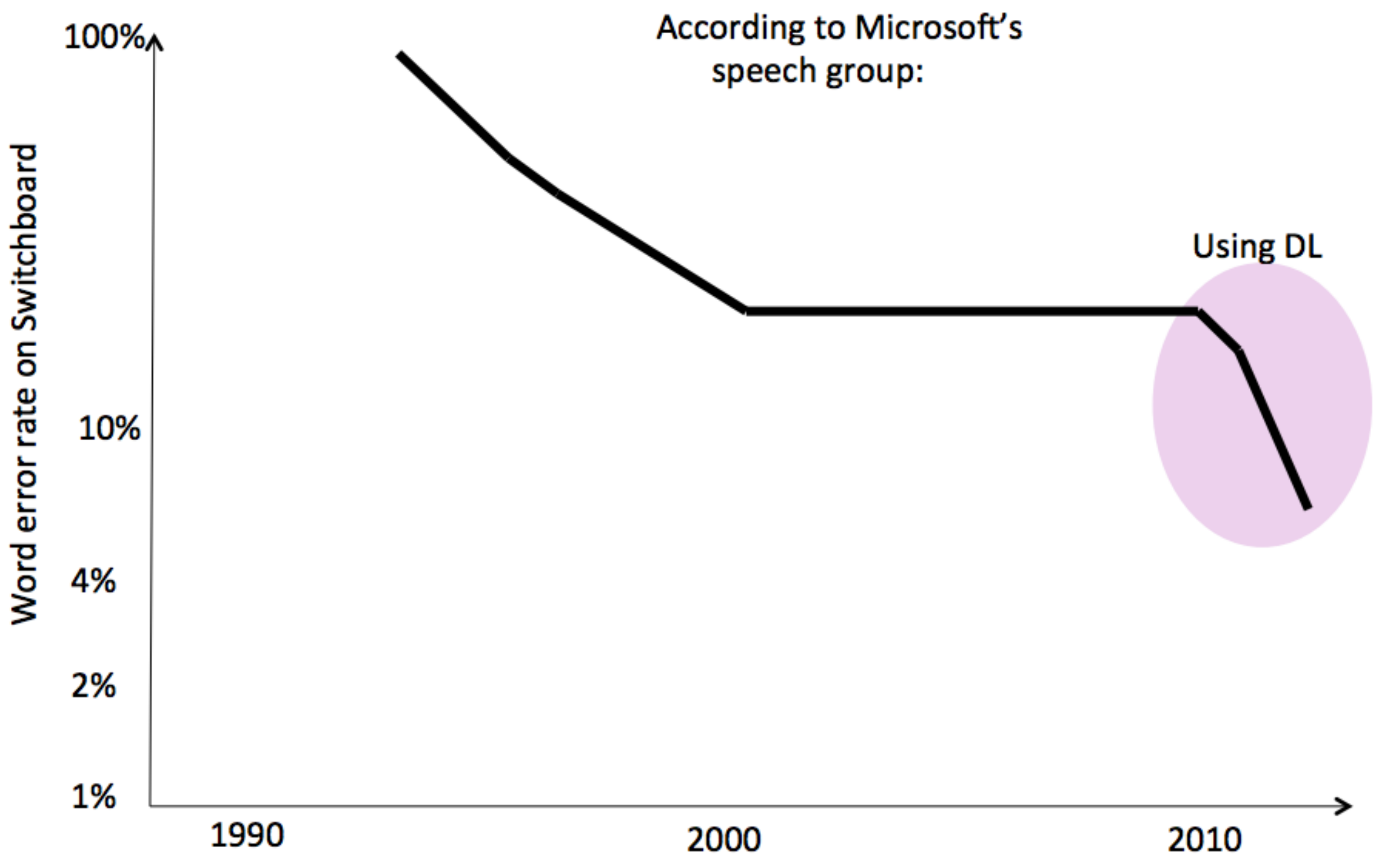


a ferry boat on a marina
with a group of people .



a little boy with a bunch
of friends on the street .

Speech Recognition



Merck Molecular Activity Challenge

Tutorials and Winners' Interviews

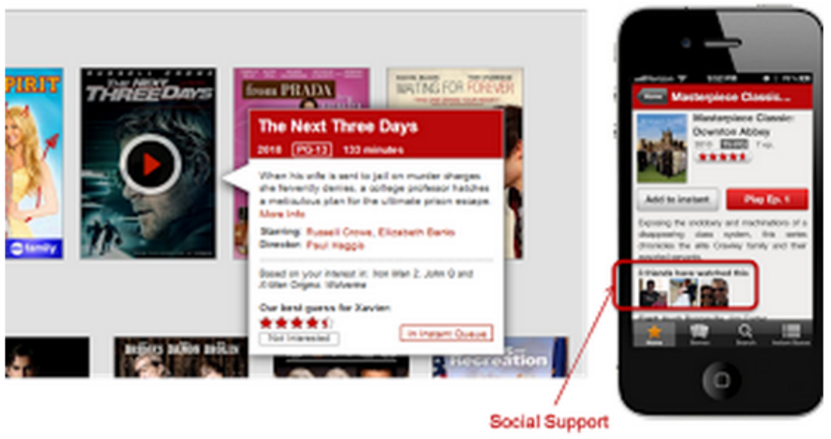
Deep Learning How I Did It: Merck 1st place interview

- 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 The Netflix Tech Blog



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:

$$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \ldots$$

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

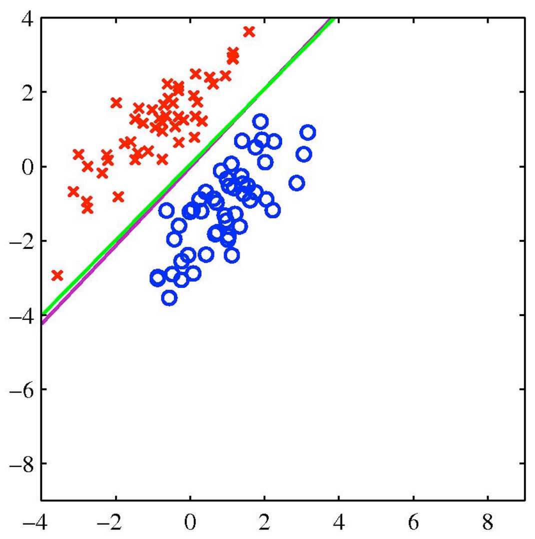
- **Unsupervised Learning**: The goal is to build a statistical model of x, which can be used for making predictions, decisions.

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

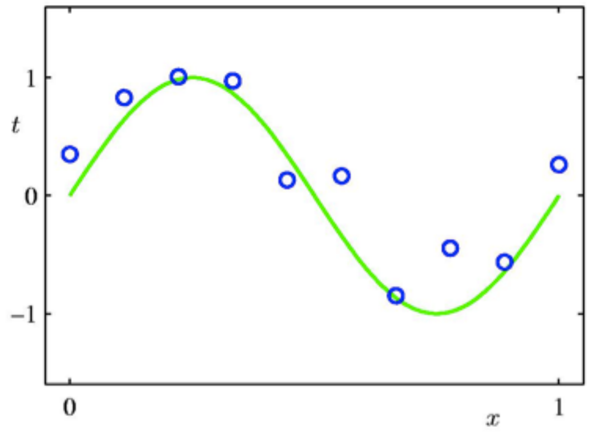
- **Semi-supervised Learning**: We are given only a limited amount of labels, but lots of unlabeled data.

Supervised Learning

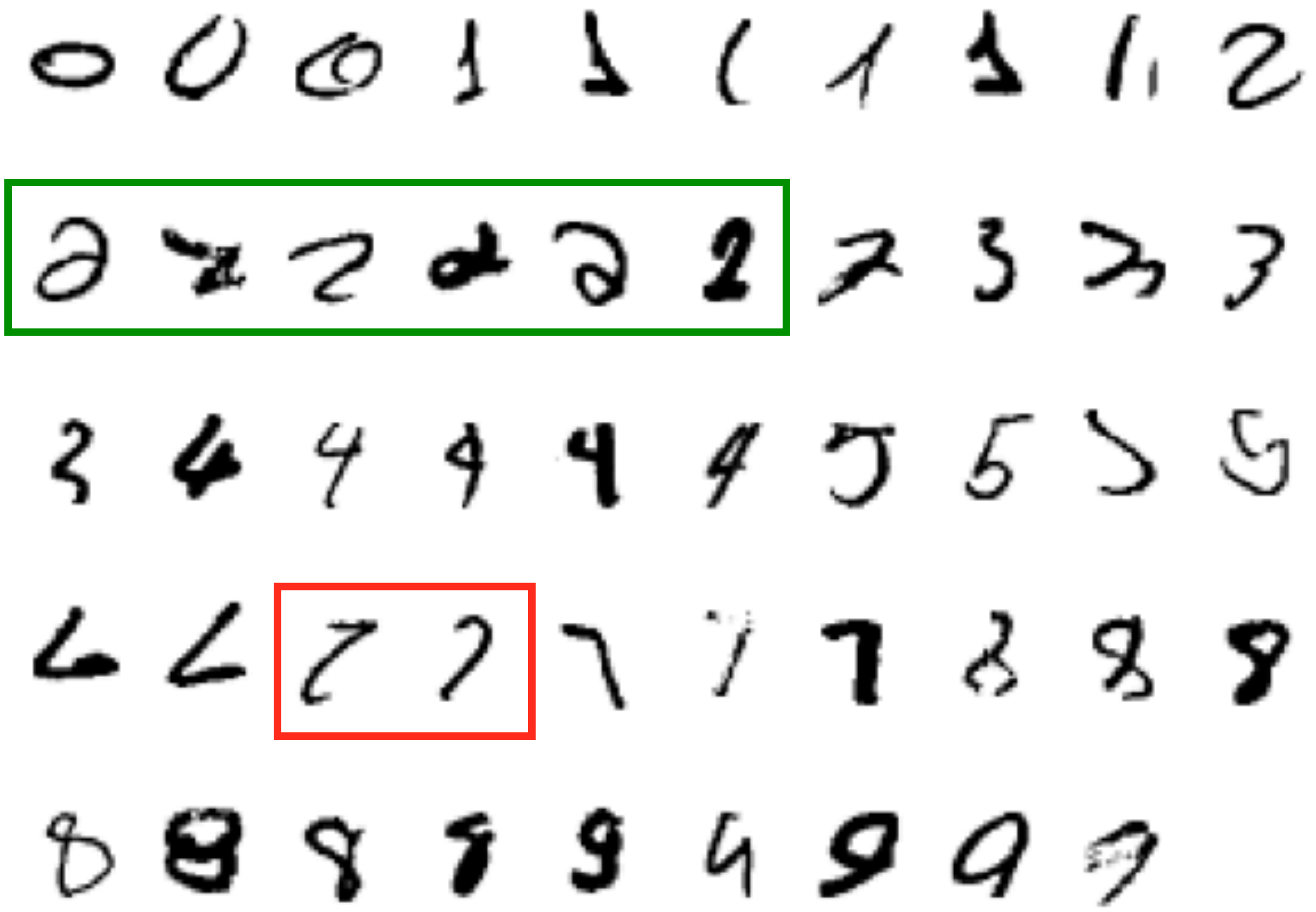
Classification: target outputs y_i are discrete class labels. The goal is to correctly classify new inputs.



Regression: target outputs y_i are continuous. The goal is to predict the output given new inputs.



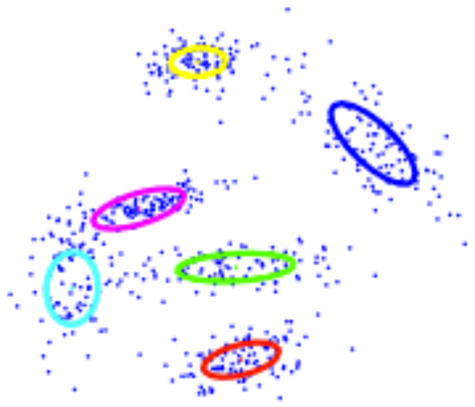
Handwritten Digit Classification



Unsupervised Learning

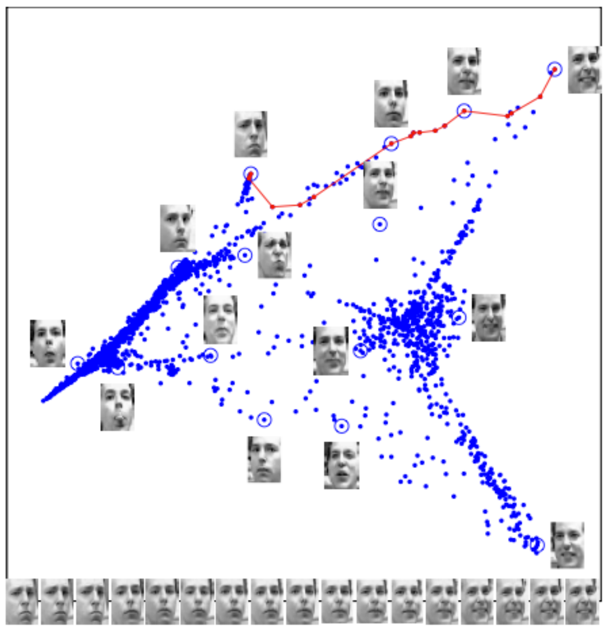
The goal is to construct statistical model that finds useful representation of data:

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

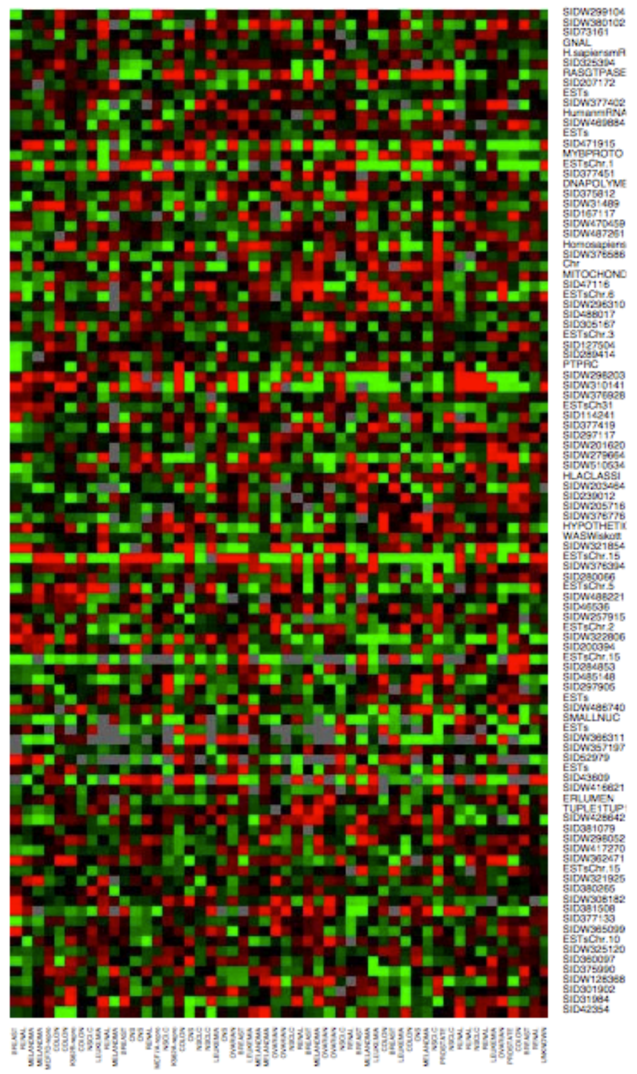


Unsupervised Learning can be used for:

- 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).

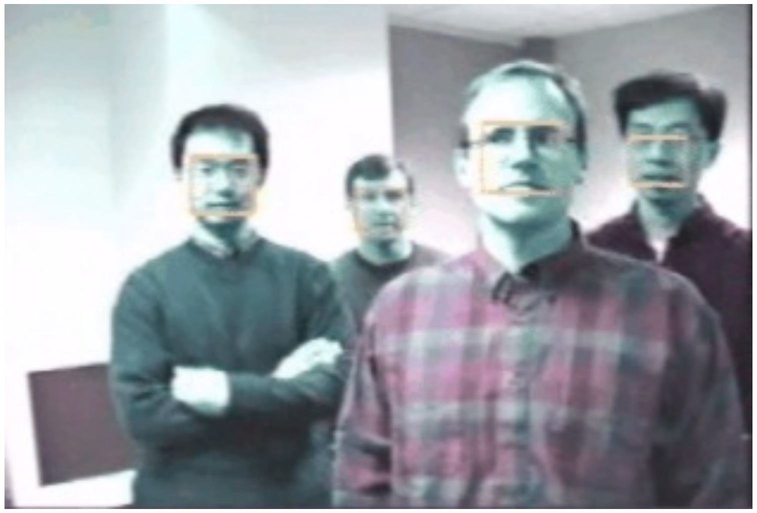
Questions we may ask:

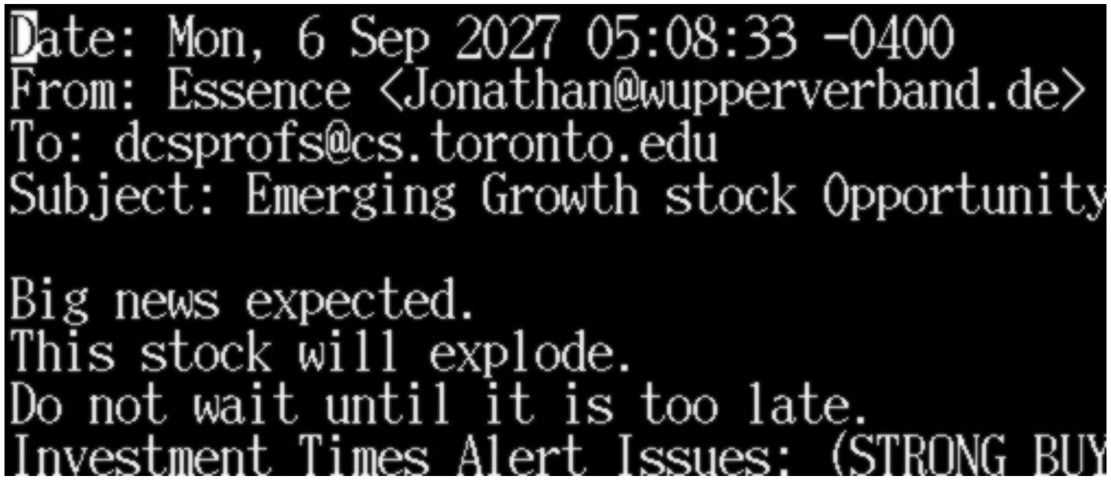
- Which samples are similar to other samples in terms of their expression levels across genes.
- Which genes are similar to each other in terms of their expression levels across samples.

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





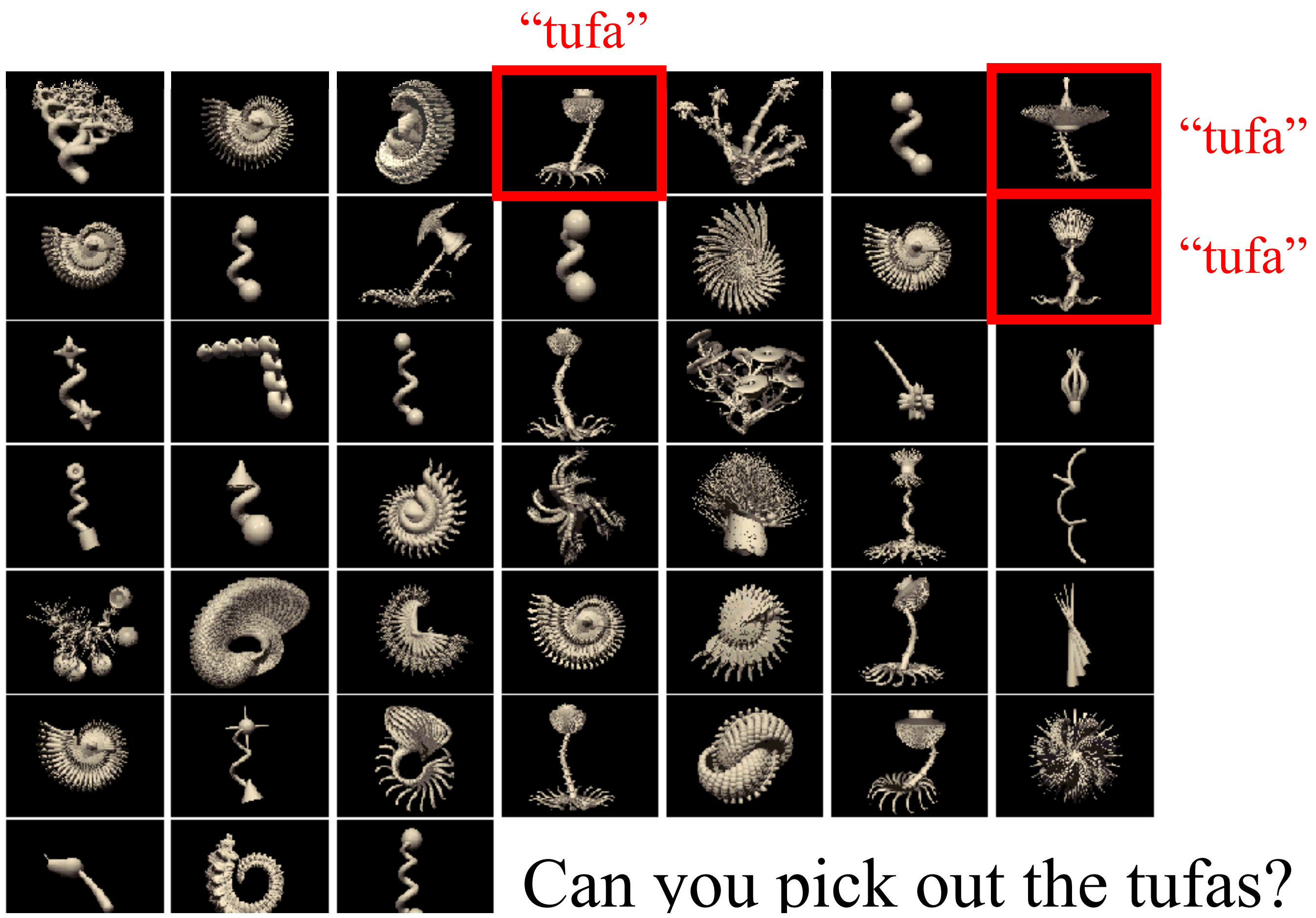
```
Date: Mon, 6 Sep 2027 05:08:33 -0400
From: Essence <Jonathan@wupperverband.de>
To: dcsprofs@cs.toronto.edu
Subject: Emerging Growth stock Opportunity

Big news expected.
This stock will explode.
Do not wait until it is too late.
Investment Times Alert Issues: (STRONG BUY
```

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

Human learning



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	model
weights	parameters
learning	fitting
generalization	test set performance
supervised learning	regression/classification
unsupervised learning.	density estimation, clustering
large grant: $1,000,000	large grant: $50,000
conference location: Snowbird, French Alps	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

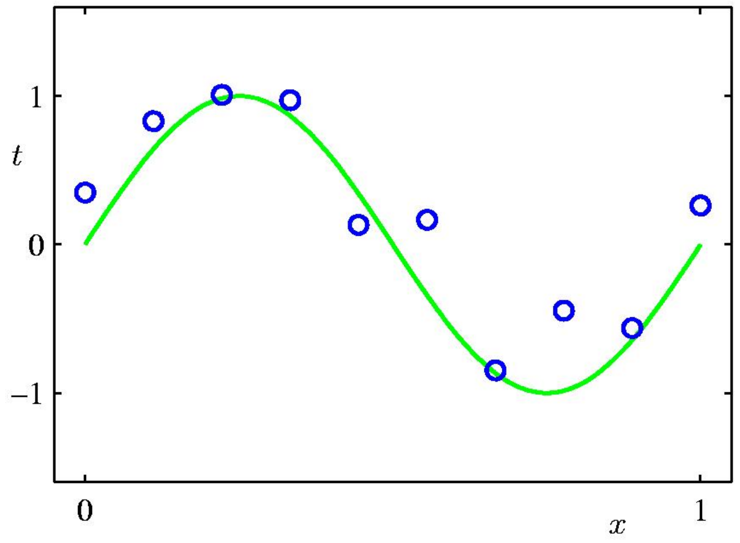
- Linear regression problem
  - continuous outputs
  - simple model

- Introduce key concepts:
  - loss functions
  - generalization
  - optimization
  - model complexity
  - regularization

Very simple example: 1-D regression

Green shows the true curve – not known

from Bishop



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

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

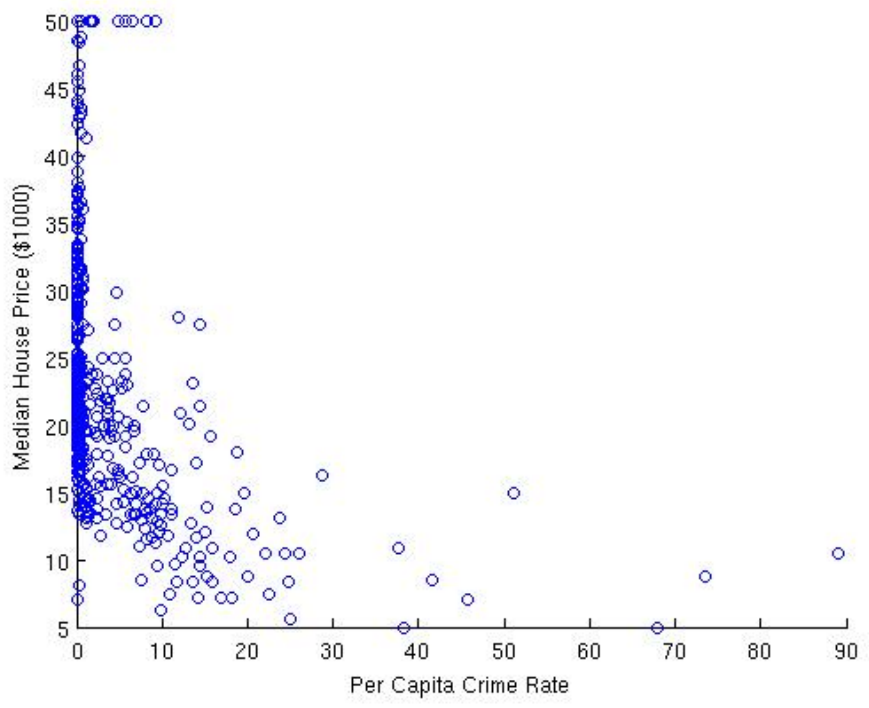
Aim: fit a curve to these points

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

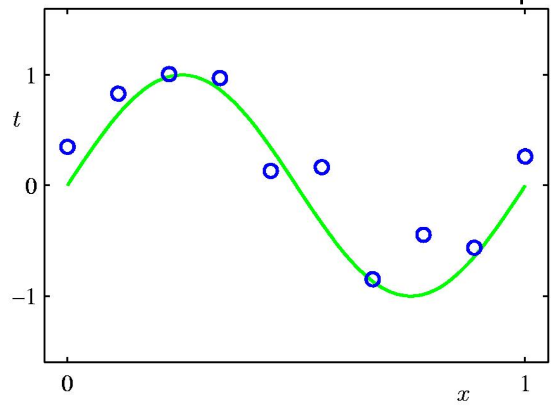
Sources of noise

- 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

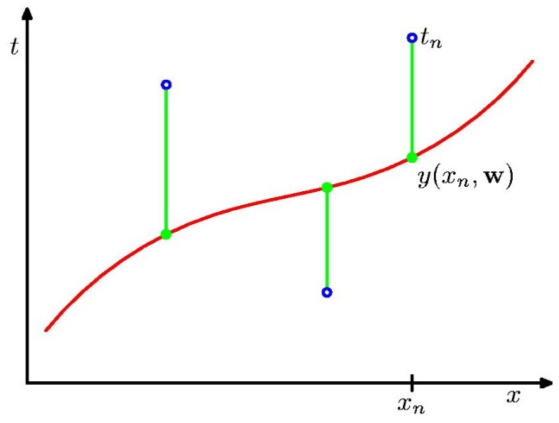
Least-squares Regression

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

from Bishop



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



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

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 λ 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

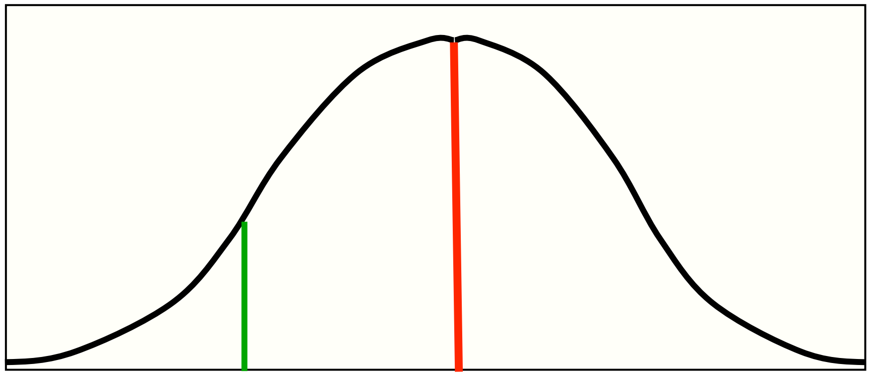
2. Batch updates: sum or average updates across every example i, then change the parameter values

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

- 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.



t = the correct answer

y = model's estimate of most probable value

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

$$p(t^{(n)} \mid y^{(n)}) = p(y^{(n)} + noise = t^{(n)} \mid \mathbf{x}^{(n)}, \mathbf{w}) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t^{(n)} - y^{(n)})^2}{2\sigma^2}}$$

$$-\log p(t^{(n)} \mid y^{(n)}) = \log\sqrt{2\pi} + \underset{\uparrow}{\log\sigma} + \frac{(t^{(n)} - y^{(n)})^2}{2\sigma^2}$$

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 **x** and write:

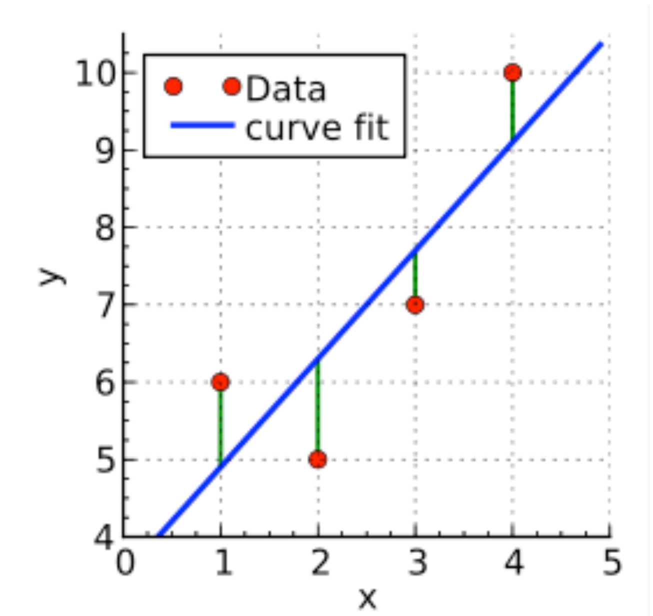
$$y(\mathbf{x}, \mathbf{w}) = \mathbf{x}^T \mathbf{w}.$$

- 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 **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 $y(\mathbf{x}_n, \mathbf{w})$ for each data point x_n and the corresponding real-valued targets t_n.



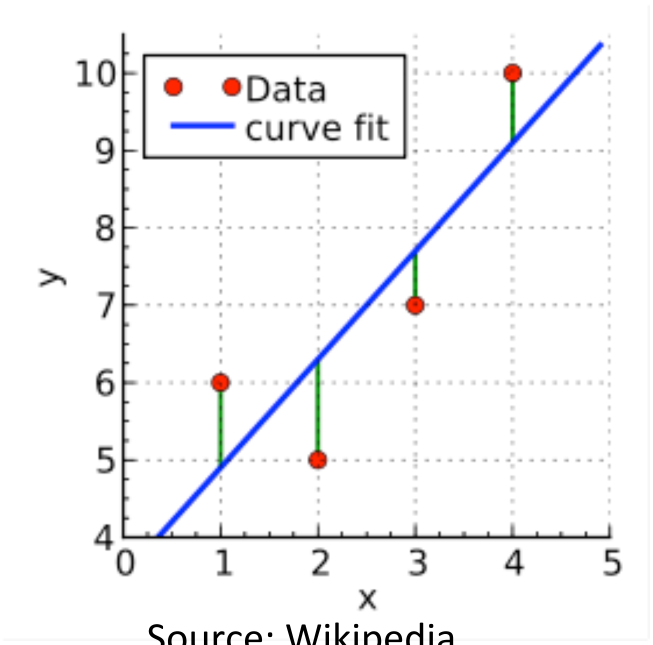
Source: Wikipedia

Loss function: sum-of-squared error function:

$$\begin{aligned} E(\mathbf{w}) &= \frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n^T\mathbf{w} - t_n)^2 \\ &= \frac{1}{2}(\mathbf{X}\mathbf{w} - \mathbf{t})^{\mathbf{T}}(\mathbf{X}\mathbf{w} - \mathbf{t}). \end{aligned}$$

Linear Least Squares

If $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is nonsingular, then the unique solution is given by:



Source: Wikipedia

optimal weights

vector of target values

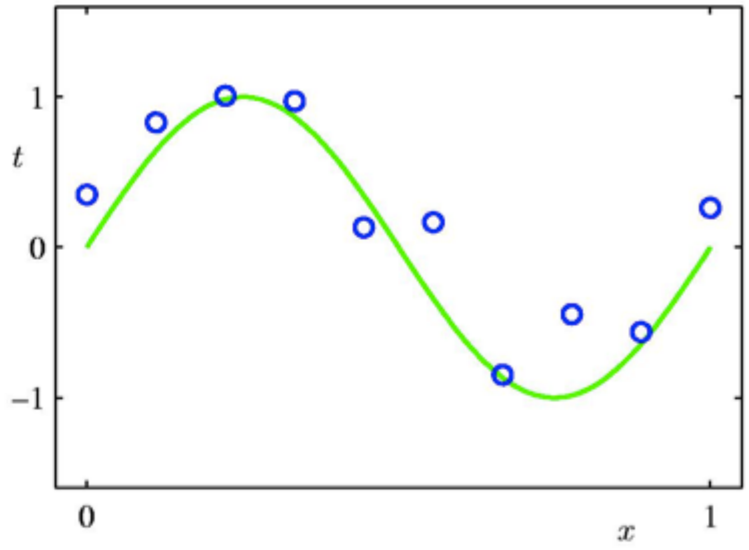
$$\mathbf{w}^* = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{t}$$

the design matrix has one input vector per row

- At an arbitrary input $\mathbf{x}_0$, the prediction is $y(\mathbf{x}_0, \mathbf{w}) = \mathbf{x}_0^T \mathbf{w}^*$.
- 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: $\mathbf{x} = (x_1, x_2, ..., x_N)^T$, together with corresponding real-valued targets: $\mathbf{t} = (t_1, t_2, ..., t_N)^T$.



- 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.

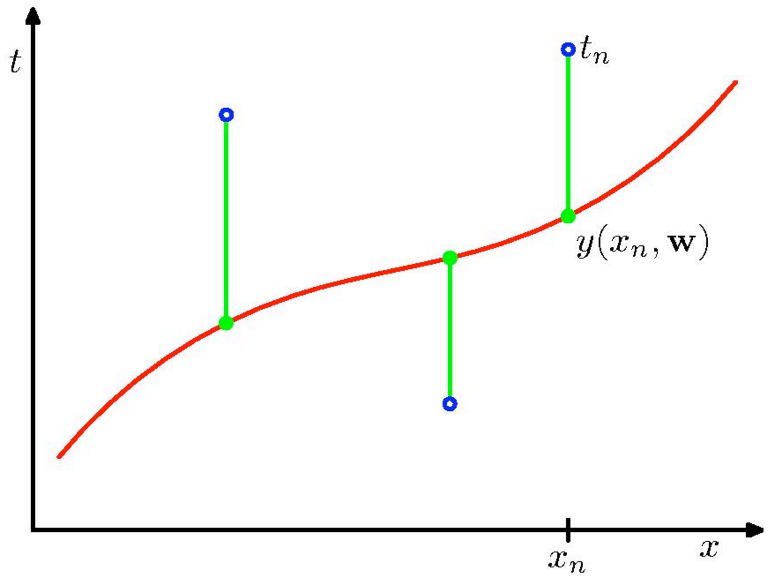
Goal: Fit the data using a polynomial function of the form:

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + ... + w_M x^M = \sum_{j=0}^{M} w_j x^j.$$

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 $y(x_n, \mathbf{w})$ for each data point x_n and the corresponding target values t_n.



Loss function: sum-of-squared error function:

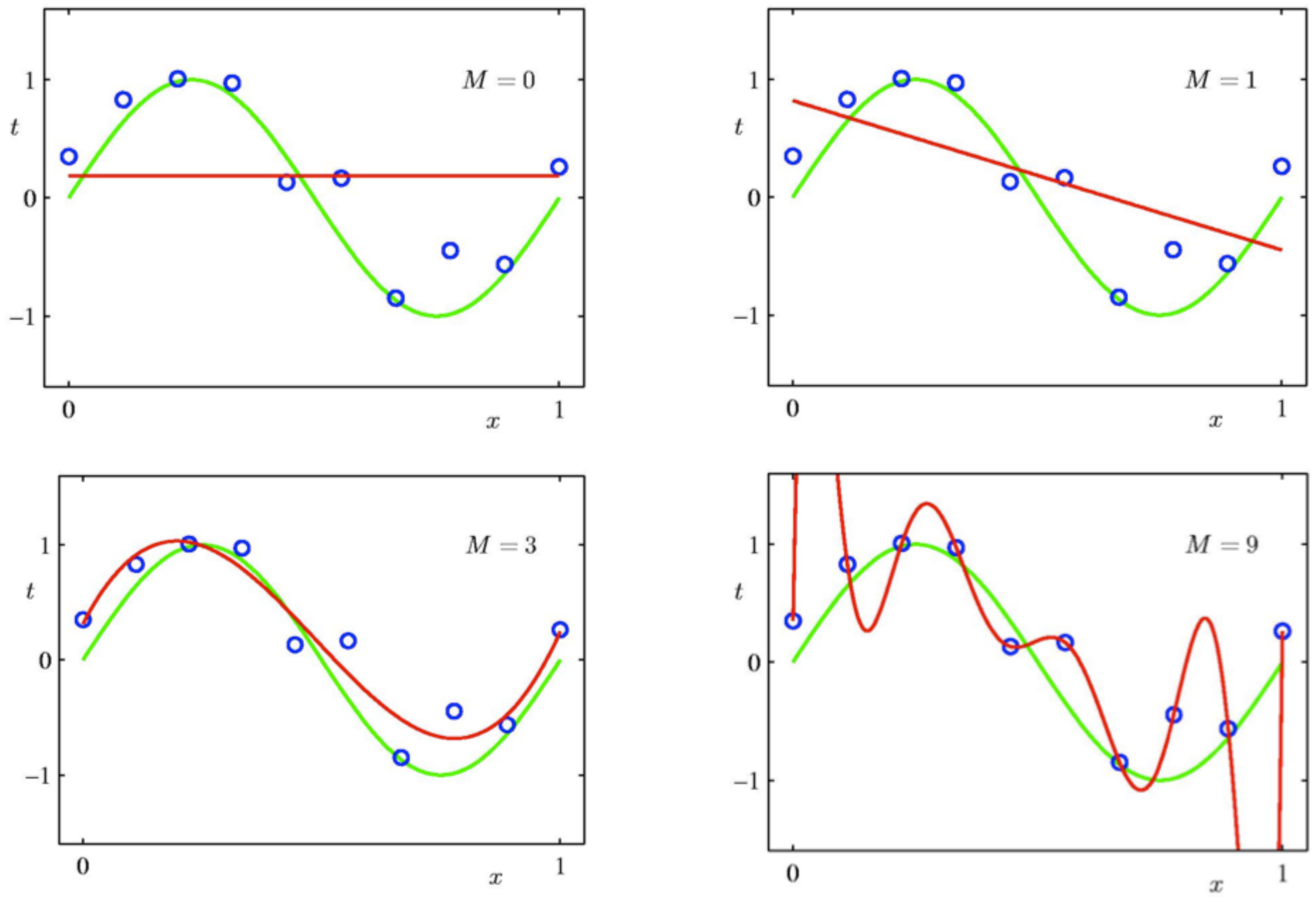
$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} (y(x_n, \mathbf{w}) - t_n)^2.$$

• 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? $\rightarrow$ **Model Selection**.

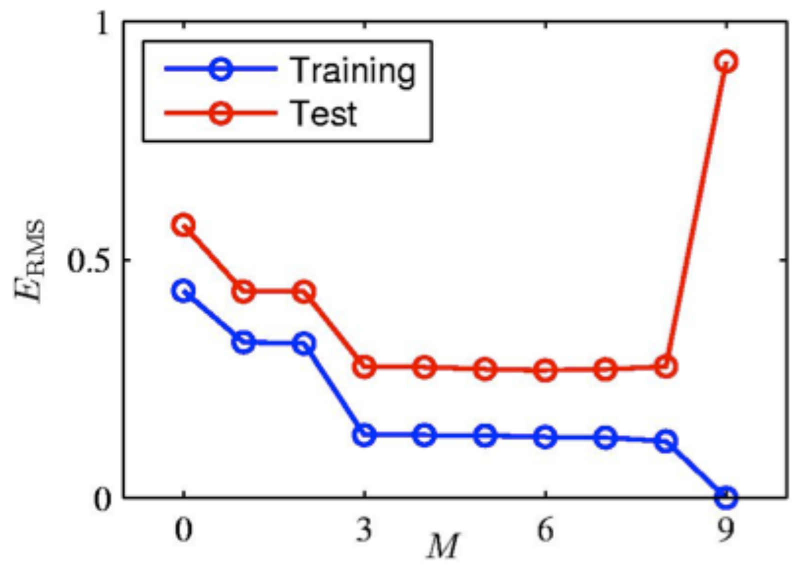
Some Fits to the Data

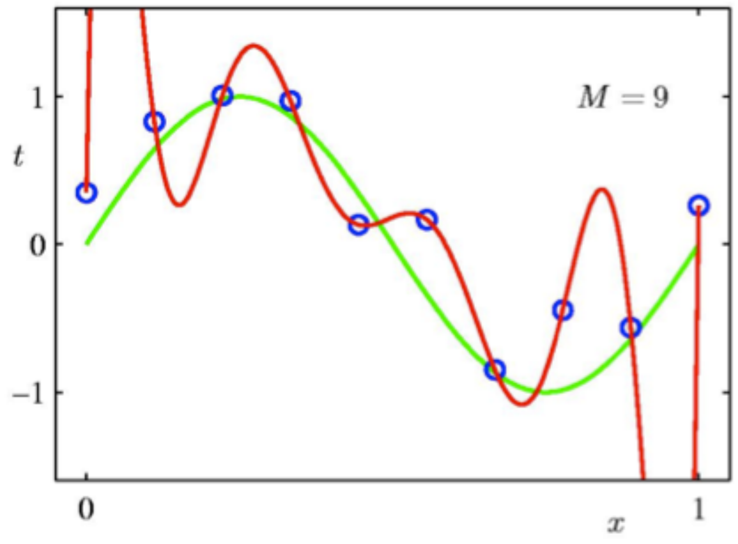


For M=9, we have fitted the training data perfectly.

Overfitting

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



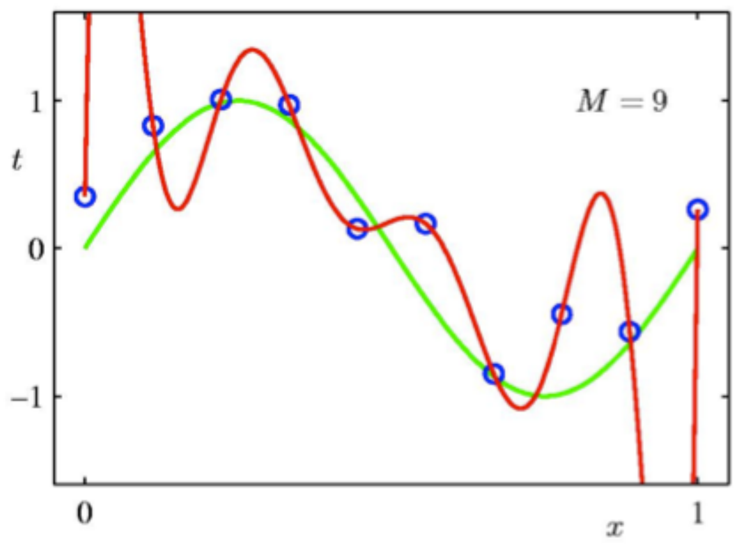


• For M=9, the training error is zero $\rightarrow$ The polynomial contains 10 degrees of freedom corresponding to 10 parameters **w**, and so can be fitted exactly to the 10 data points.

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

Overfitting

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

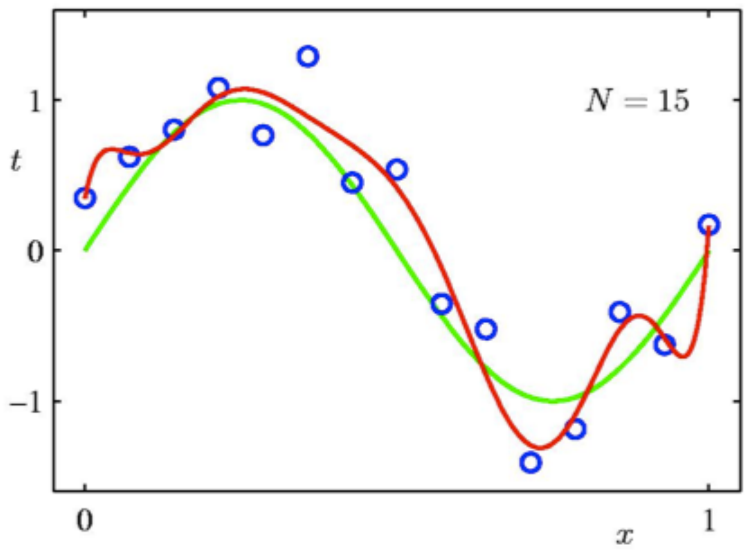


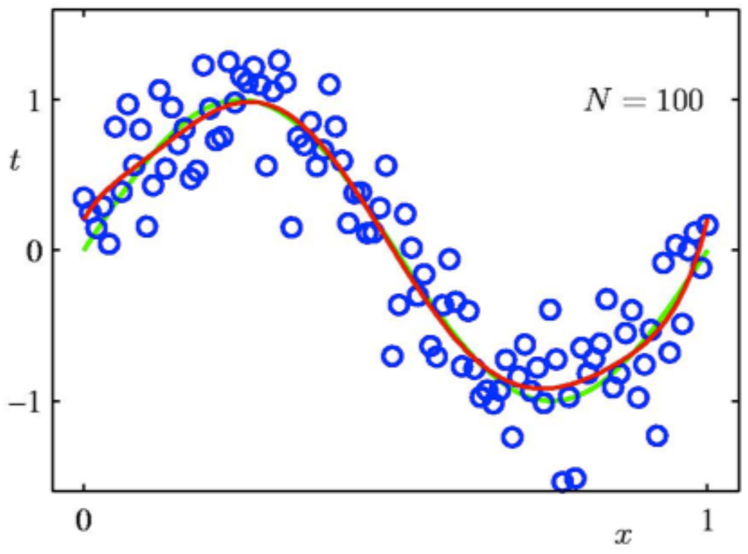
- As M increases, the magnitude of coefficients gets larger.
- For M=9, the coefficients have become finely tuned to the data.
- Between data points, the function exhibits large oscillations.

More flexible polynomials with larger M tune to the random noise on the target values.

Varying the Size of the Data

9th order polynomial



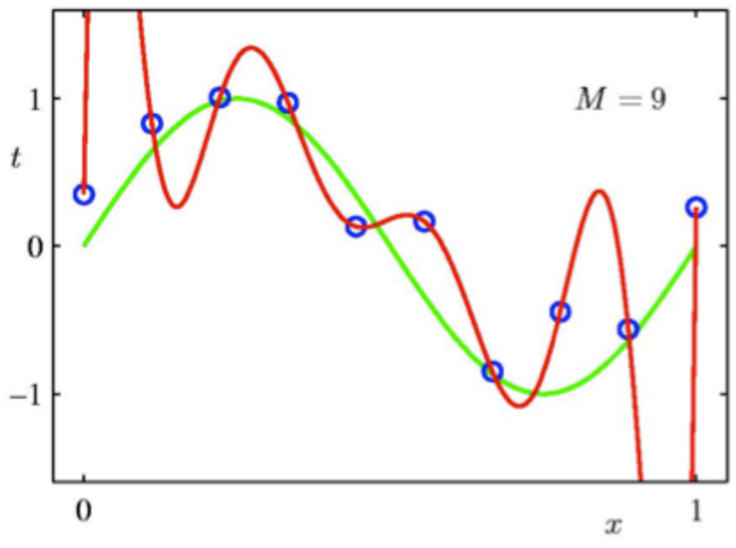


• For a given model complexity, the overfitting problem becomes less severe as the size of the dataset increases.

• 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.

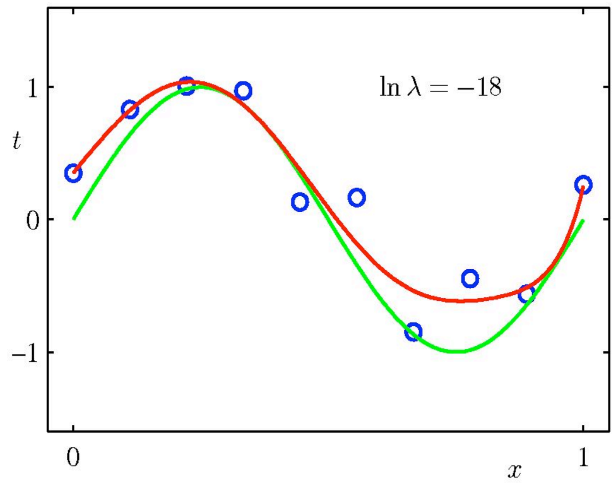
penalized error function

target value

regularization parameter

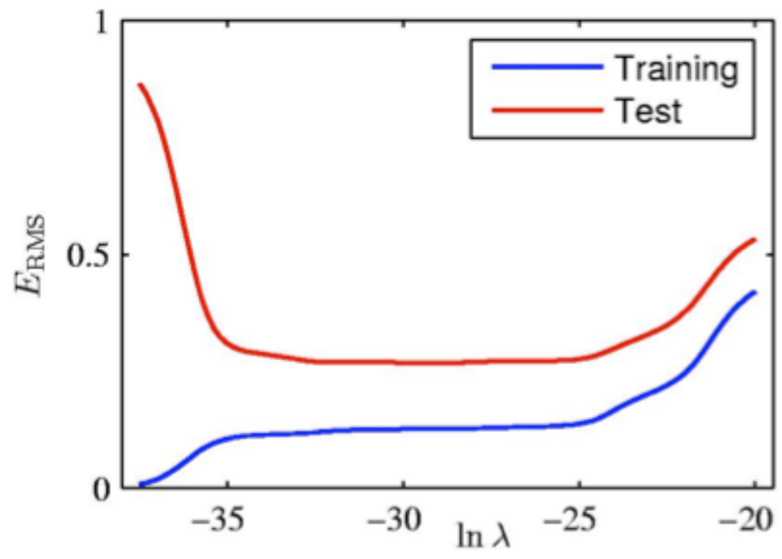
$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

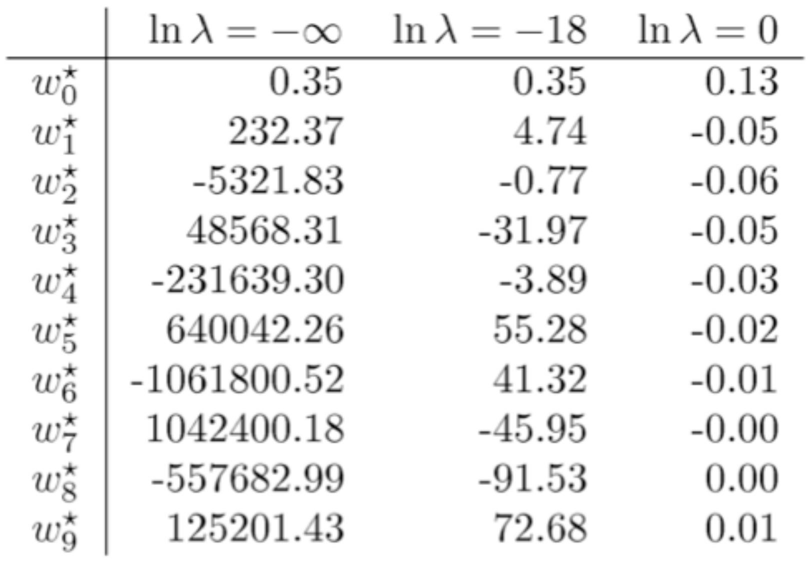
where $||\mathbf{w}|| = \mathbf{w}^T\mathbf{w} = w_1^2 + w_2^2 + ... + w_M^2$ called the regularization term. Note that we do not penalize the bias term w_0.



- 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**".

Regularization





	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Graph of the root-mean-squared training and test errors vs. lnλ for the M=9 polynomial.

How to choose λ?

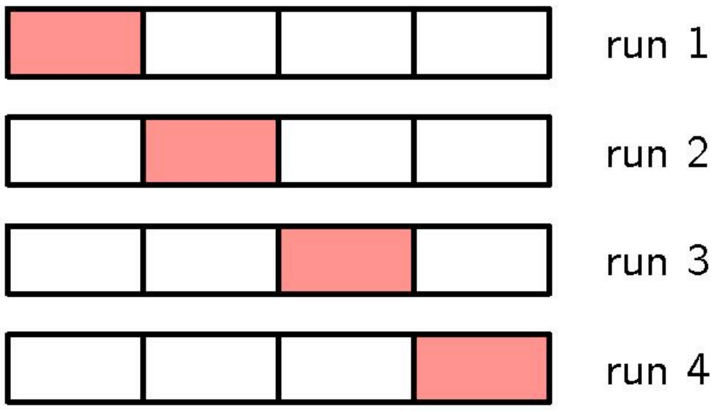
Cross Validation

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.

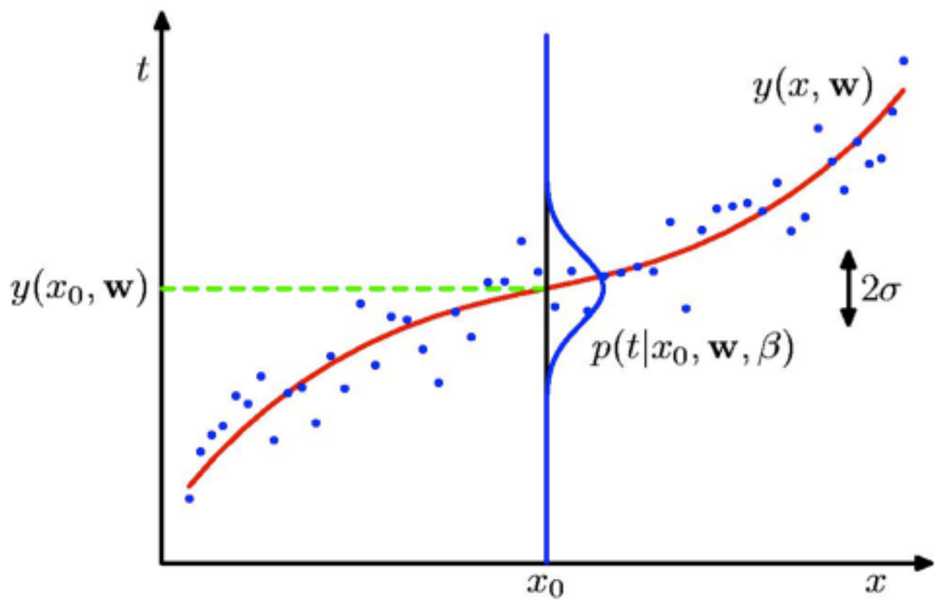
Probabilistic Perspective

- 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:

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon,$$

where ϵ is a random error having Gaussian distribution with zero mean, and is independent of **x**.



Thus we have:

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

where β 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:

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \prod_{i=1}^{N} \mathcal{N}(t_n | y(\mathbf{x}_n, \mathbf{w}), \beta^{-1}).$$

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

$$\ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \underbrace{-\frac{\beta}{2} \sum_{n=1}^{N} (y(\mathbf{x}_n, \mathbf{w}) - t_n)^2}_{\beta E(\mathbf{w})} + \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi).$$

- 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 $\mathbf{w}_{ML}$ by maximizing log-likelihood. Then maximizing w.r.t. β:

$$\frac{1}{\beta_{ML}} = \frac{1}{N} \sum_{n} (y(\mathbf{x}_n, \mathbf{w}_{ML}) - t_n)^2.$$

Predictive Distribution

Once we determined the parameters **w** and β, we can make prediction for new values of **x**:

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