LUND UNIVERSITY

FMAN-45: Machine Learning

Lecture 1: Introduction and Overview

Cristian Sminchisescu

Machine Learning Course

- Lecturer
  - Prof. Cristian Sminchisescu

    (about me in 2 lines: PhD Applied Math & CS, INRIA, 99-02; Postdoc, AI lab, University of Toronto, 02-04; 04-, Faculty at Chicago, Toronto, Bonn, Lund)

- Teaching Assistants
  - Jakob Bergstedt, David Nilsson, Aleksis Pirinen, Maria Priisalu

- Course web page and email
  - Web: http://www.ctr.maths.lu.se/course/machinlearn/2016/
  - Email: ml-2016@maths.lth.se
  - Send all course correspondence (but not the assignments!) to the above email not to our personal emails (will not be able to answer)
  - Please, **use this address only sparingly** for exceptional communication -- best use Moodle (see next slides)

Course Organization

- Lectures
  - 2 x 2 hours per week
  - Time: Mondays 1:15 – 3pm, Tuesdays 8:15-10 am
  - Rooms E:A then V:A
- Topics
  - Foundations of machine learning and statistical modeling
- Grading
  - 5 assignments (for P/F) + final exam (higher grades)
  - To pass (P) you need proficiency > 75% in each assignment

Administrative

- **Assignments**
  - Please read instructions for each assignment carefully
  - Solutions for each assignment included in a single pdf
  - For each assignment that includes code, **submit two files: a pdf of the report and a zip archive with code**
    - Make sure you do not archive image sub-directories which are big
  - Need over 75% success **in each assignment** to pass
  - No exceptions, no late policy! Assignments submitted after deadline will not be considered

  - Assignments 1-2 given Nov 1st
  - Assignment 3 given Nov 8th
  - Assignments 1-3 due Nov 22nd, 11:59 pm
  - Assignments 4, 5, introduced on Nov. 14th and 15th
  - Assignments 4, 5, due December 6th, 11:59 pm

- **Exam** (beyond Pass) on December 13th
  - in class, open book, no computer, 2 hours

Registration in Moodle

- Information for registration
  - Go to this address
    http://moodle.maths.lth.se/course/view.php?id=30
  - Follow instructions

- Use Moodle to
  - View the pdfs of lectures
  - Submit your assignments
  - Get feedback and grades
  - Do not send assignments to the course or our personal emails, as **they will not be considered**

Course Readings

- **Textbooks**
  - Christopher Bishop. *Pattern Recognition and Machine Learning*, Springer, 2006
  - R. Sutton and A. Barto. *Reinforcement learning: An Introduction*, MIT Press, MIT Press, 1998 (2nd edition 2017 , see online version)
- Other useful references
  - T. Hastie, R. Tibshirani, J. Friedman: *The elements of statistical learning, data mining, inference and prediction*, 2nd edition, Springer, 2009
  - D. Koller and N.Friedman: *Probabilistic Graphical Models, Principles and Techniques*, MIT Press, 2009

Course Topics

- Training, testing, generalization, hypothesis spaces
- Linear regression and classification
- (Deep) neural networks
- Markov decision processes
- Kernel methods and support vector machines
- Graphical models
- Mixture models, Expectation Maximization
- Variational and sampling methods (time permitting)
- Reinforcement learning

Slide credits: I will sometimes use or adapt slides provided with the textbook, the ML Group in Toronto, or other authors. In such cases, whenever possible, I will also aim to preserve the original formatting, to further signal it.

What is Machine Learning?

- **Machine learning** is concerned with the design and development of algorithms and techniques that allow computers to "learn“

- The major focus of machine learning research is to extract information from data automatically, by computational and statistical methods

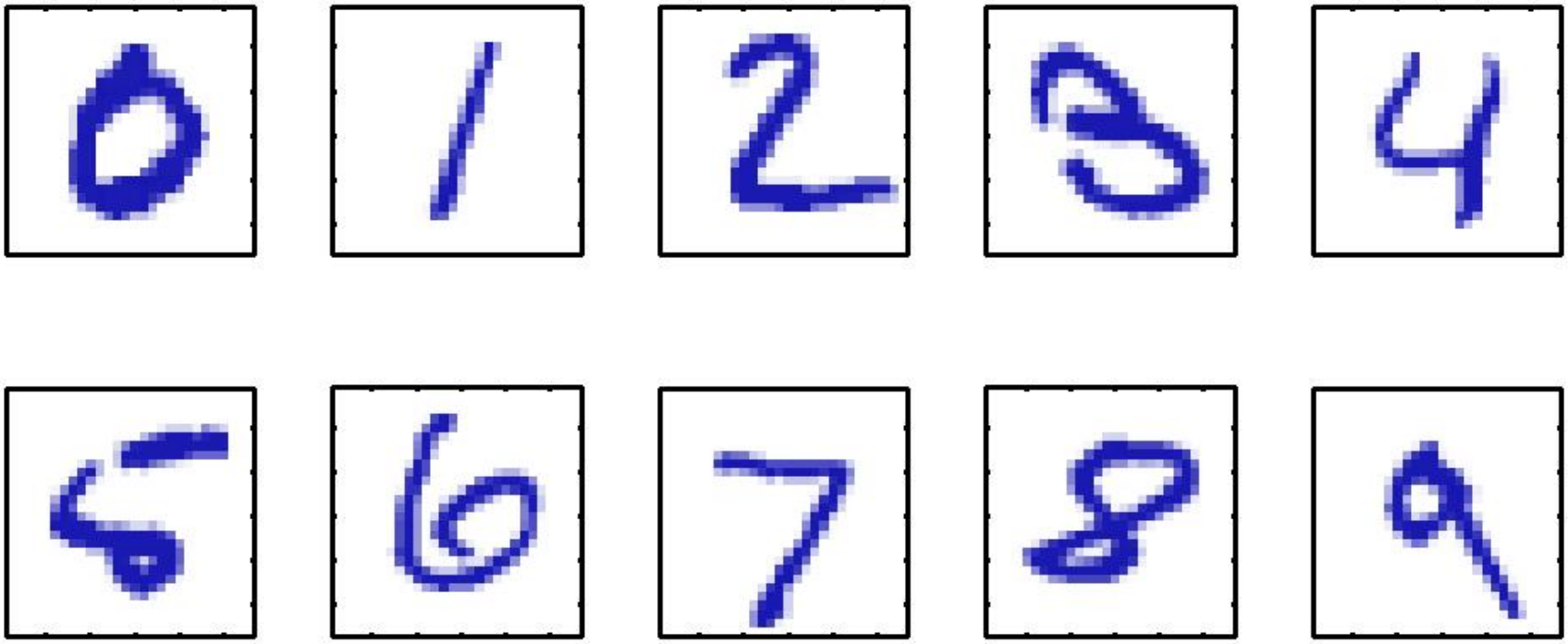
- Machine learning is closely related to statistics, data mining, but also to theoretical computer science

Related Areas of Study

- Stochastic signal processing
  - denoising, source separation, scene analysis, morphing
- Data mining
  - (Relatively) simple machine learning on huge datasets
- Data compression and coding
  - state of the art methods for image compression and error correcting codes use learning methods
- Decision making, planning
  - use both utility and uncertainty optimally
- Adaptive software agents, auctions or preferences
  - action choice under limited resources and reward signals

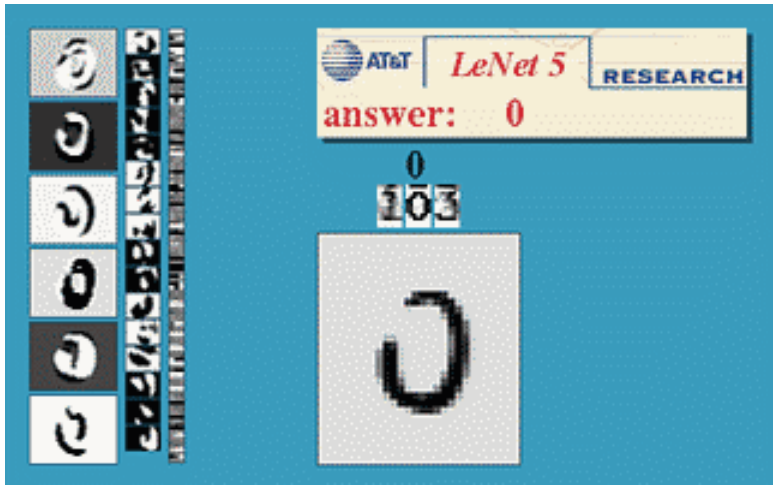
Task: Handwritten Digit Recognition

Difficult to characterize what is specific to each digit



Optical character recognition (OCR)

Technology to convert scanned images/docs to text



Digit recognition, AT&T labs
http://yann.lecun.com/exdb/lenet/

Automatic license plate reading

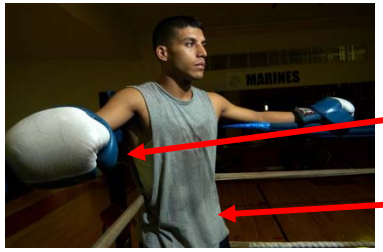
Visual Recognition of People in Images

General poses, high-dimensional *(30-100 dof)*

Self-occlusions

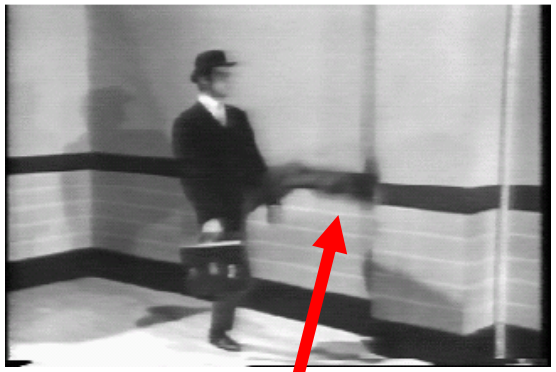
Difficult to segment the individual limbs

Different body sizes



Loss of Information in the perspective projection

Partial views



Accidental allignments
Motion blur

Several people, occlusions

Reduced observability of body parts due to loose fitting clothing

Machine Learning Approach

- It is difficult to explicitly design programs that can recognize people or digits in images
  - Modeling the object structure, the physics, the variability, and the image formation process can be very difficult
- Instead of designing the programs by hand, we collect many examples that specify the correct outputs for different inputs
- A machine learning algorithm takes these examples and produces a program trained to give the answers we want
  - If designed properly, the program may work in new situations, not just the ones it was trained on

When to use Machine Learning?

- A machine learning approach is most effective when the structure of the task is not well understood - or too difficult to model explicitly -, but can be characterized by a dataset with strong statistical regularity

- Machine learning is also useful in dynamic situations, when the task is constantly changing
  - e.g. a robot operating in an unknown environment

A spectrum of machine learning tasks

Statistics----------Artificial Intelligence

- Low-dimensional data (e.g. less than 100 dimensions)
- Lots of noise in the data
- There is not much structure in the data, and what structure there is, can be represented by a fairly simple model.
- The main problem is distinguishing true structure from noise.

- High-dimensional data (e.g. more than 100 dimensions)
- The noise is not sufficient to obscure the structure in the data if we process it right.
- There is a huge amount of structure in the data, but the structure is too complicated to be represented by a simple model.
- The main problem is figuring out a way to represent the complicated structure that allows it to be learned.

slide from MLG@Toronto

Machine Learning Applications

- natural language processing
- speech and handwriting recognition
- object recognition in computer vision
- image retrieval
- robotics
- medical diagnosis
- bioinformatics
- classifying DNA sequences
- detecting credit card fraud
- stock market analysis
- recommender systems
- analyzing social patterns and networks
- visualization, game playing and many more…

Sample Applications of Machine Learning

Online search engines

flickr

webshots

Ask Images

Cydral Image & Site Search

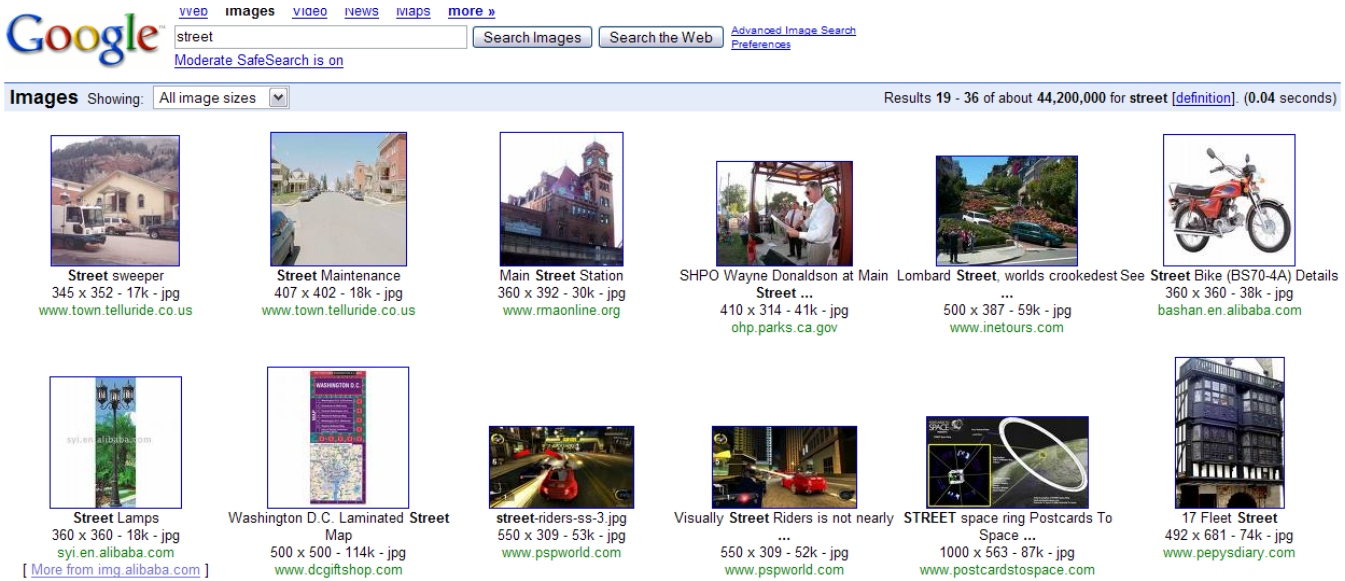
picsearch

Google Image Search

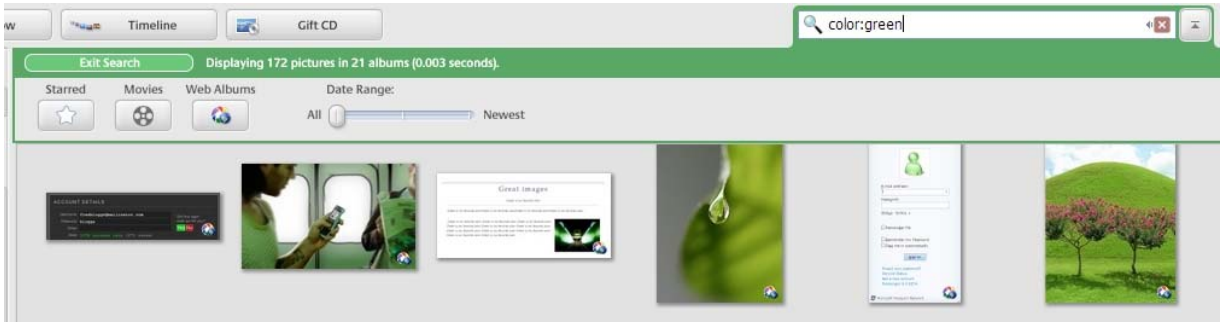
bing

altavista

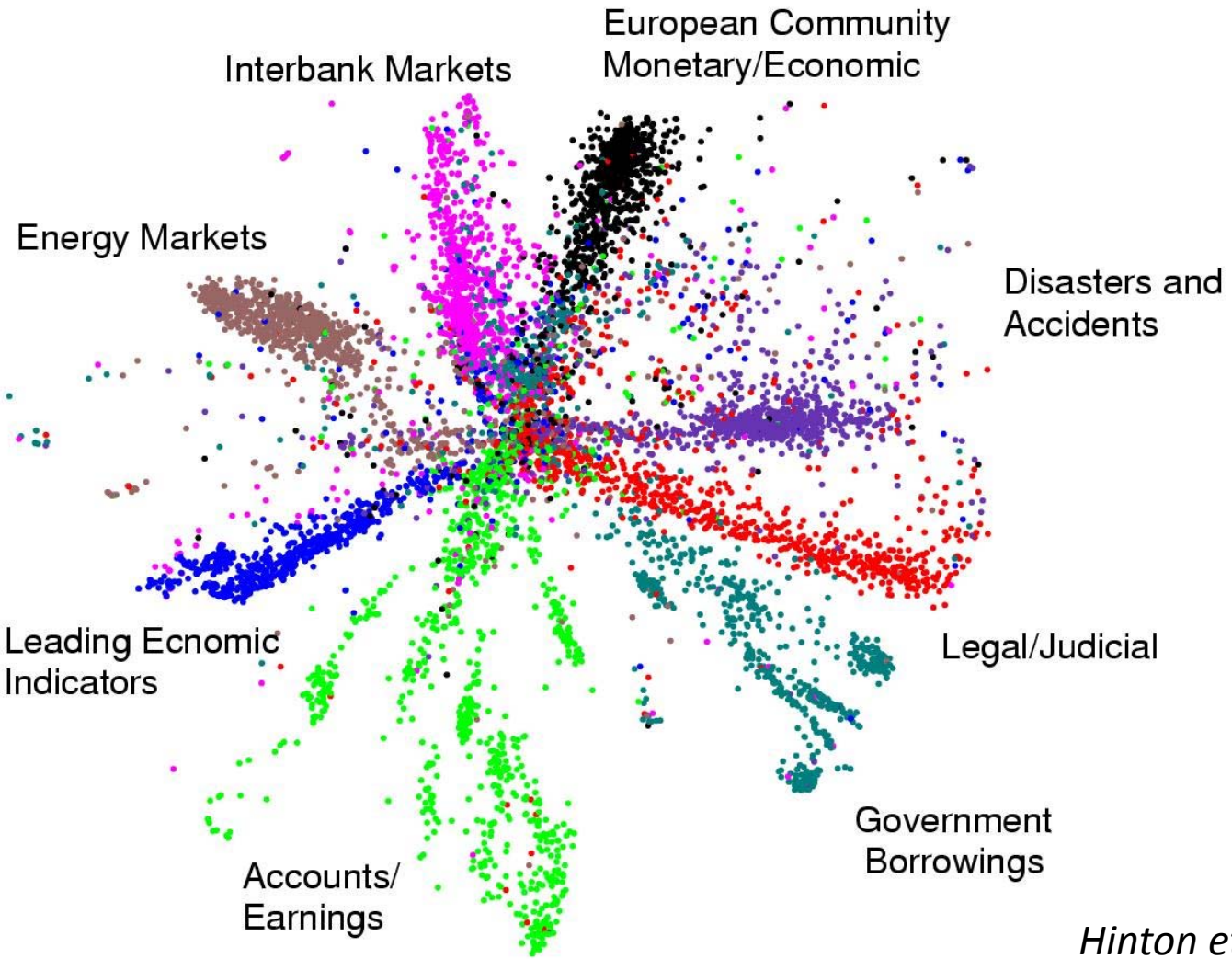
Query: STREET



Organizing photo collections



Displaying the structure of a set of documents using a deep neural network



Hinton et al., 2007

IBM's Watson (not) in Jeopardy (2011)

90 IBM servers, 2880 cores (@3.55Gb), 15TB RAM

Cities question: *Its largest airport was named for a World War II hero; its second largest, for a World War II battle*

Champions *answer 2/3 questions with 85-95% accuracy*

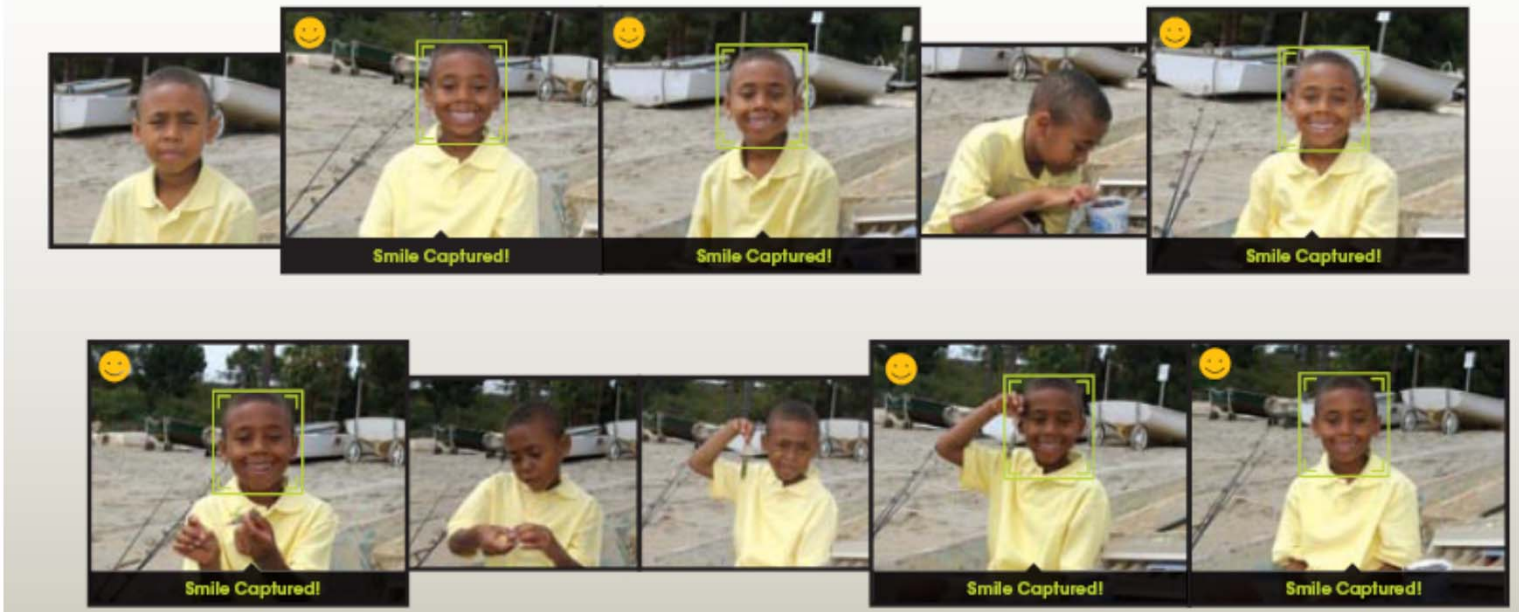
Face detection

Most digital cameras now detect faces

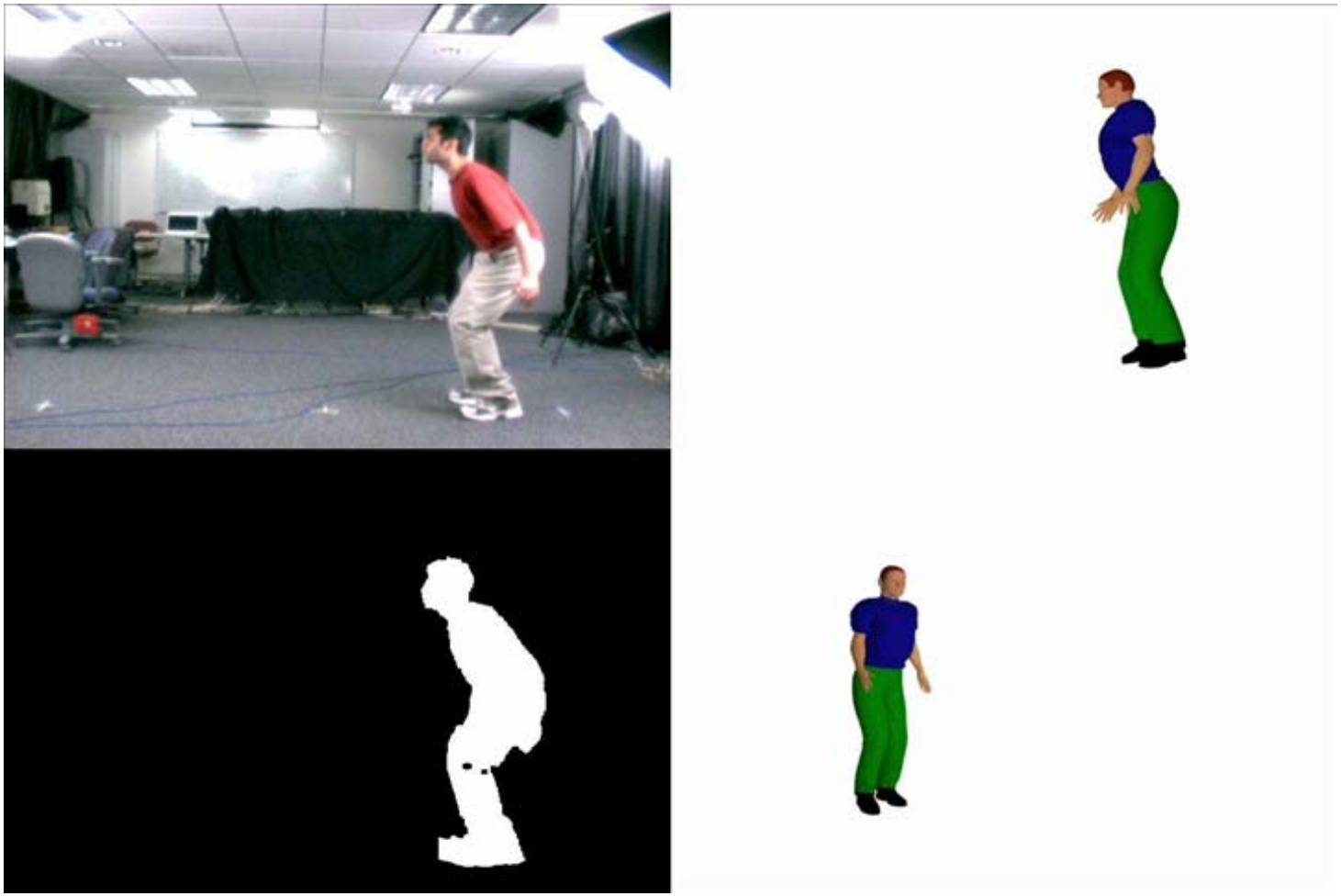
Smile Detection

The Smile Shutter flow

Imagine a camera smart enough to catch every smile! In Smile Shutter Mode, your Cyber-shot® camera can automatically trip the shutter at just the right instant to catch the perfect expression.



Monocular 3D Human Pose Reconstruction



Sminchisescu, Kanujia, Metaxas, PAMI 2007

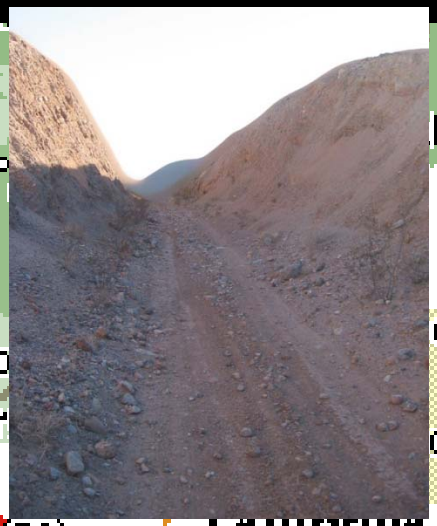
3D Human Pose – Microsoft's Kinect (2011)

Shotton et al., CVPR 2011

Grand Challenge

following (black) slides from Sebastian Thrun

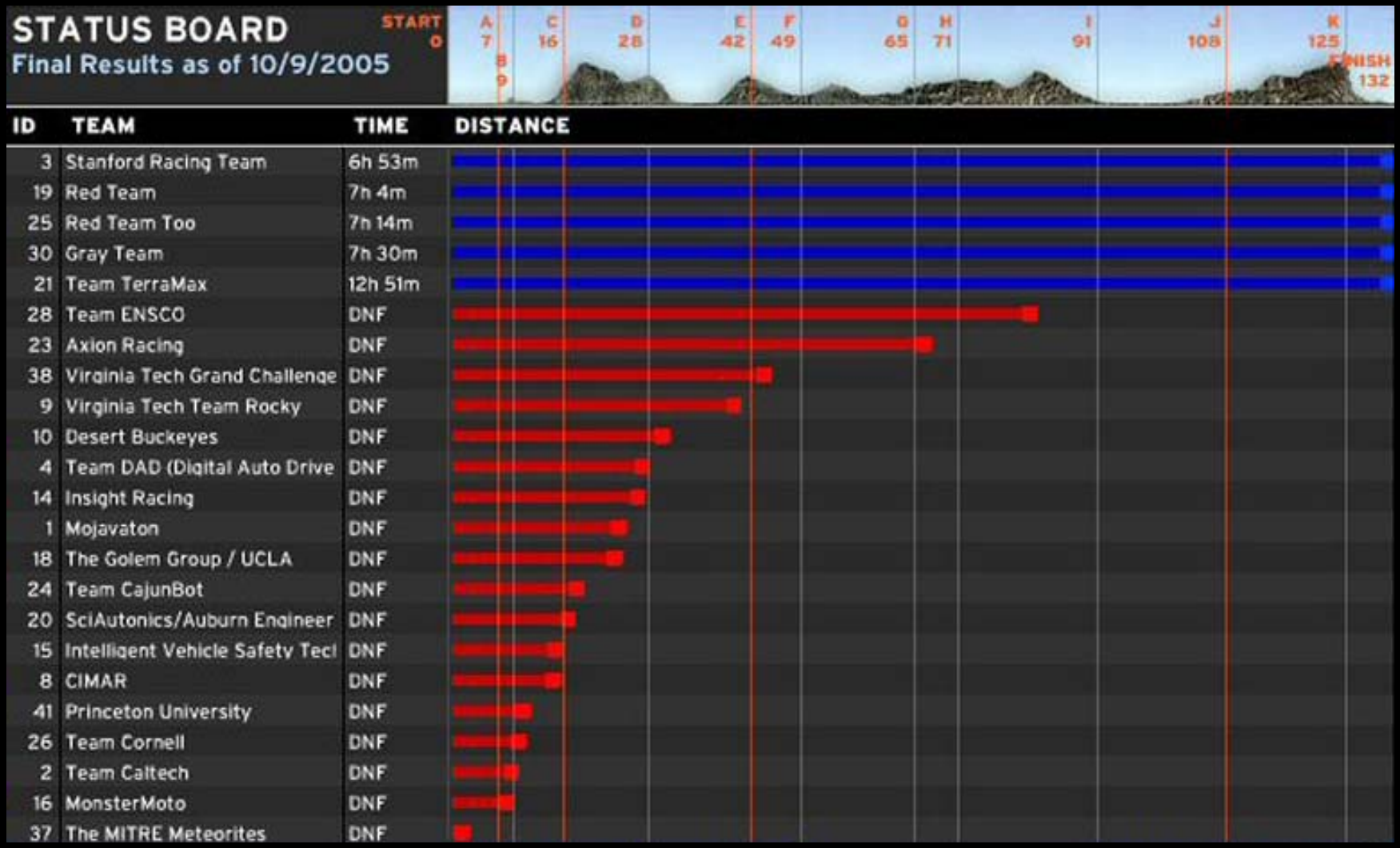
2004: Barstow, CA, to Primm, NV



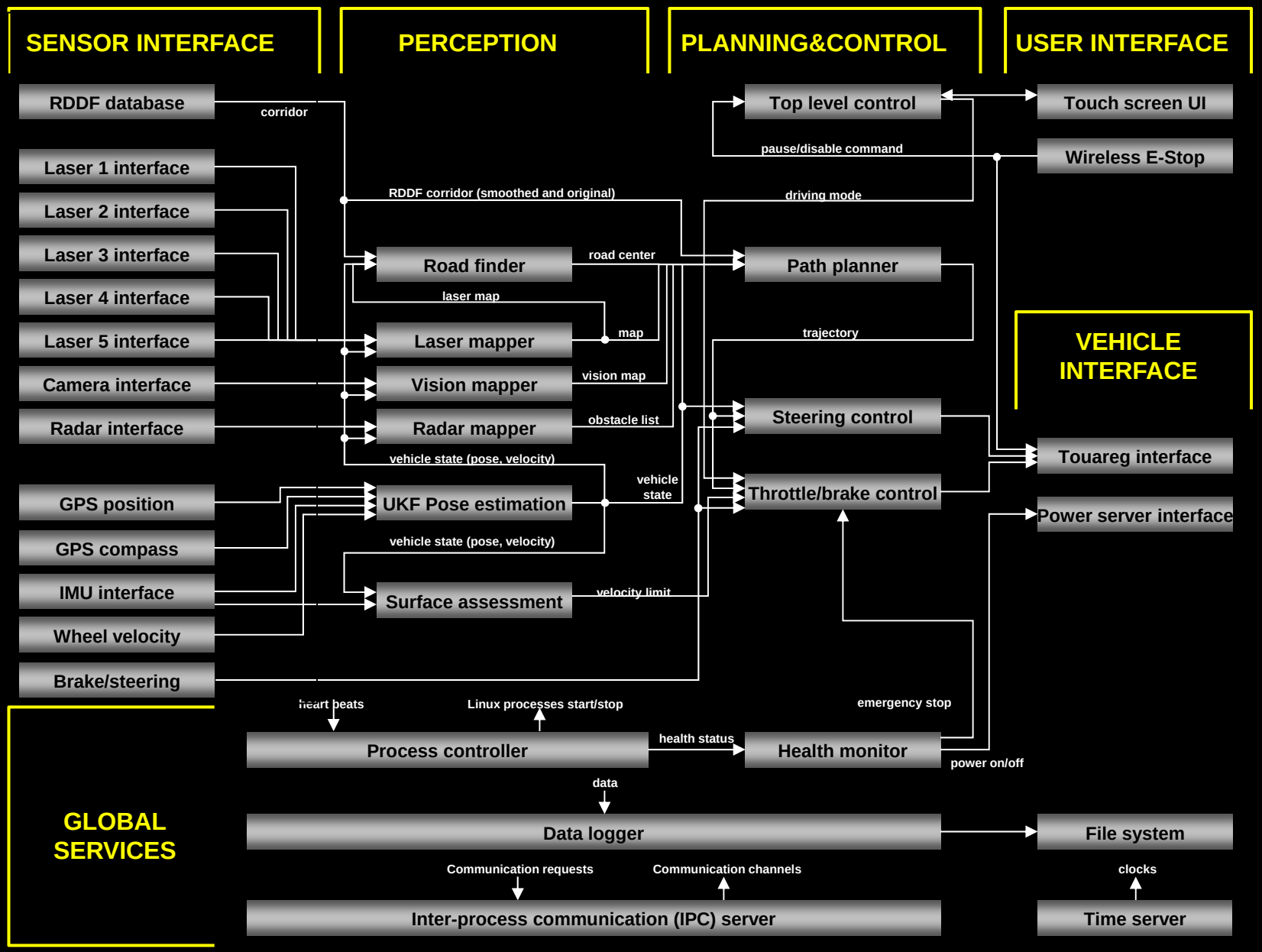
- 150 mile off-road robot race across the Mojave desert
- Natural and manmade hazards
- No driver, no remote control
- Fastest vehicle wins the race (and 2 million dollar prize)

Grand Challenge 2005: 195 Teams

Final Result: Five Robots finished!



Stanley Software Architecture

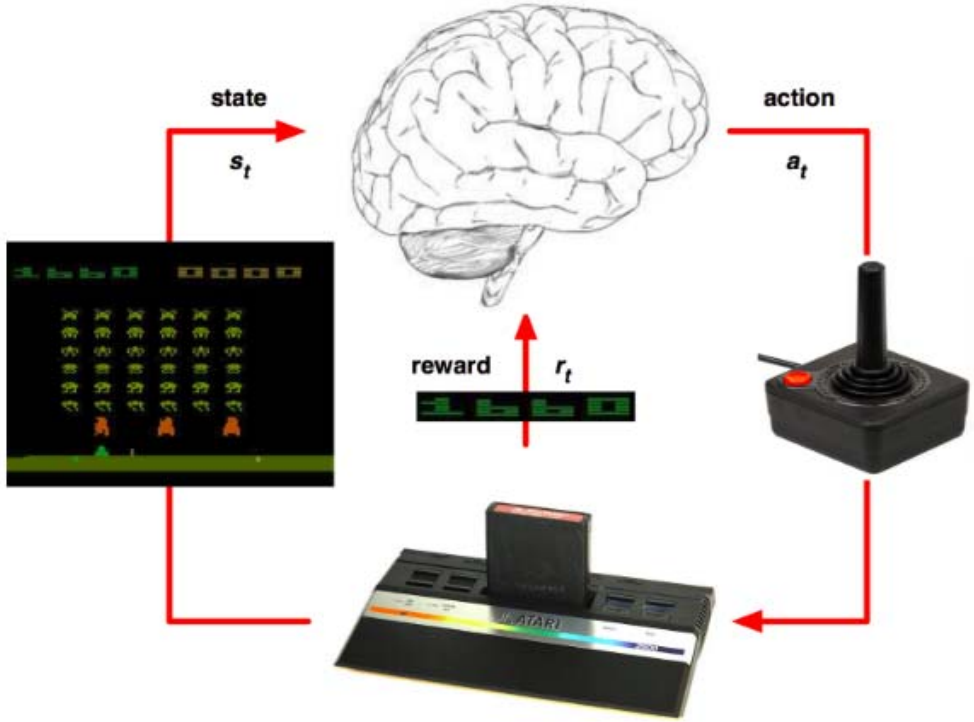


Google’s Self-Driving Car

All major car manufacturers now focus on `self-driving’ research

Game Playing, Reinforcement Learning

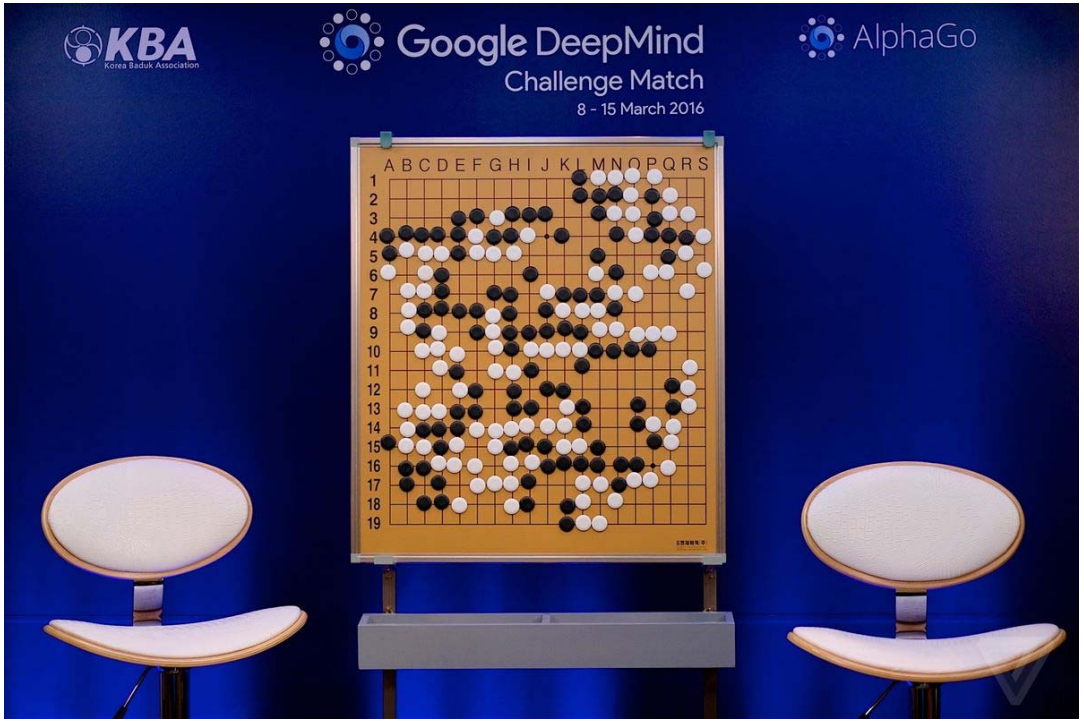
Atari

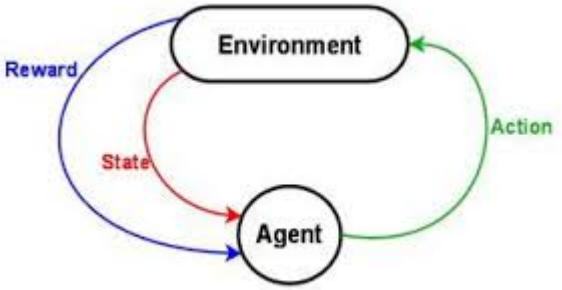


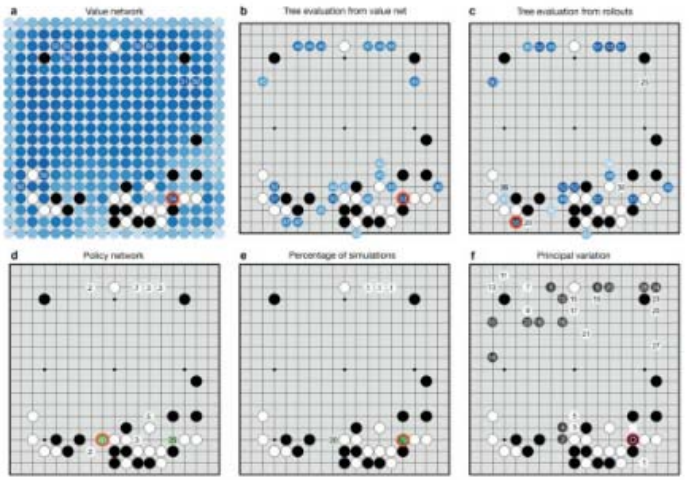
Learning to Play Atari using Deep Reinforcement Learning

Game Playing, Reinforcement Learning

Go







AlphaGo

Long Term

- Robot operating in an unknown environment
- Avoids obstacles, climbs stairs, recognizes faces, gestures…

Is Machine Learning Solved?

Lots of success already but…

- Many existing systems lag behind human performance
  - Comparatively, see how fast children learn
- Handling large, unorganized data repositories, under uncertain and indirect supervision is an open problem
- Designing complex systems (e.g. a robot with sophisticated function and behavior) is an open problem
- Fundamental advances necessary at all levels
  - computational, algorithmic, representational, implementation and integration

Standard Learning Tasks

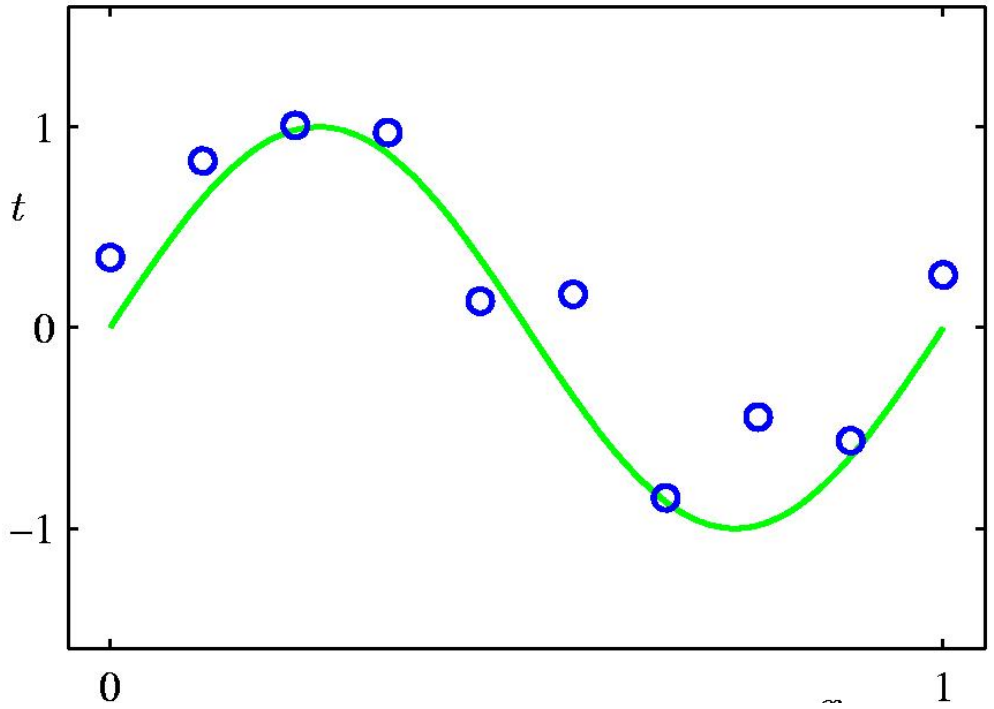
- ***Supervised Learning:*** given examples of inputs and desired outputs (labeled data), predict outputs on future inputs
  - Ex: classification, regression, time series prediction
  - Who provides the correct outputs? Can these always be measured? Can the process scale?
- ***Unsupervised Learning*****:** given only inputs (unlabeled data), automatically discover hidden representations, features, structure, etc.
  - Ex: clustering, outlier detection, compression
  - How can we know that representation is good?
- ***Semi-supervised Learning:*** given both labeled and unlabeled data, leverage information to improve both tasks
  - Somewhat practical compromise for data acquisition and modeling

Standard Learning Tasks

- ***Reinforcement Learning*:** given sequences of inputs, actions from a fixed set, and scalar rewards and punishments, learn to select action sequences in a way that maximizes expected reward
  - Not much information in the reward, which is often delayed
  - e.g. game playing, robot operating in a structured environment

Case Study
Polynomial Curve Fitting

Polynomial Curve Fitting



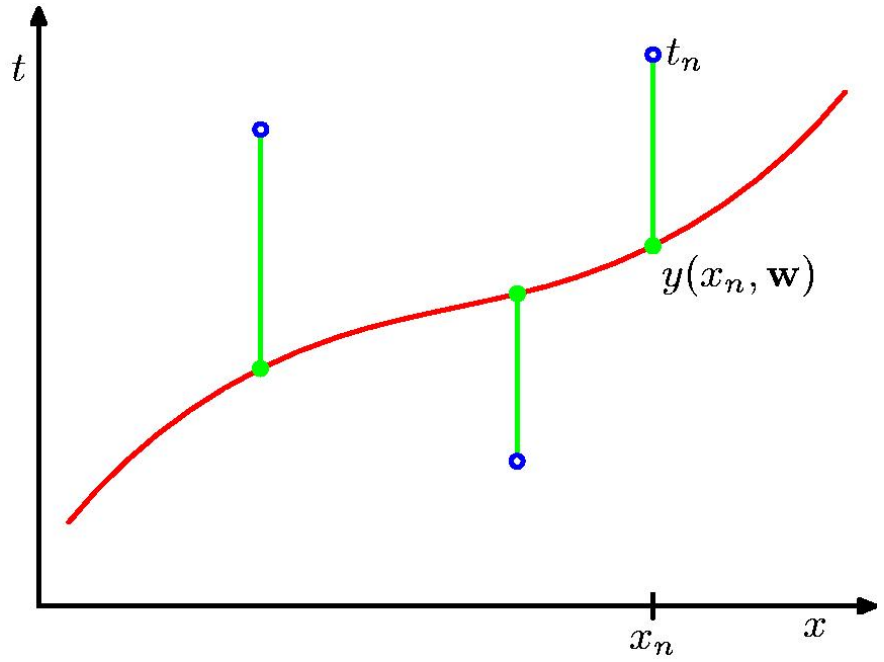
Synthetic data generated from:

$t(x) = sin2\pi x + \epsilon,$
$\epsilon \sim N\ (0, 0.3)$

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

Model is **linear** in the parameters *w* and **non-linear** in the input x

Sum-of-Squares Error Function

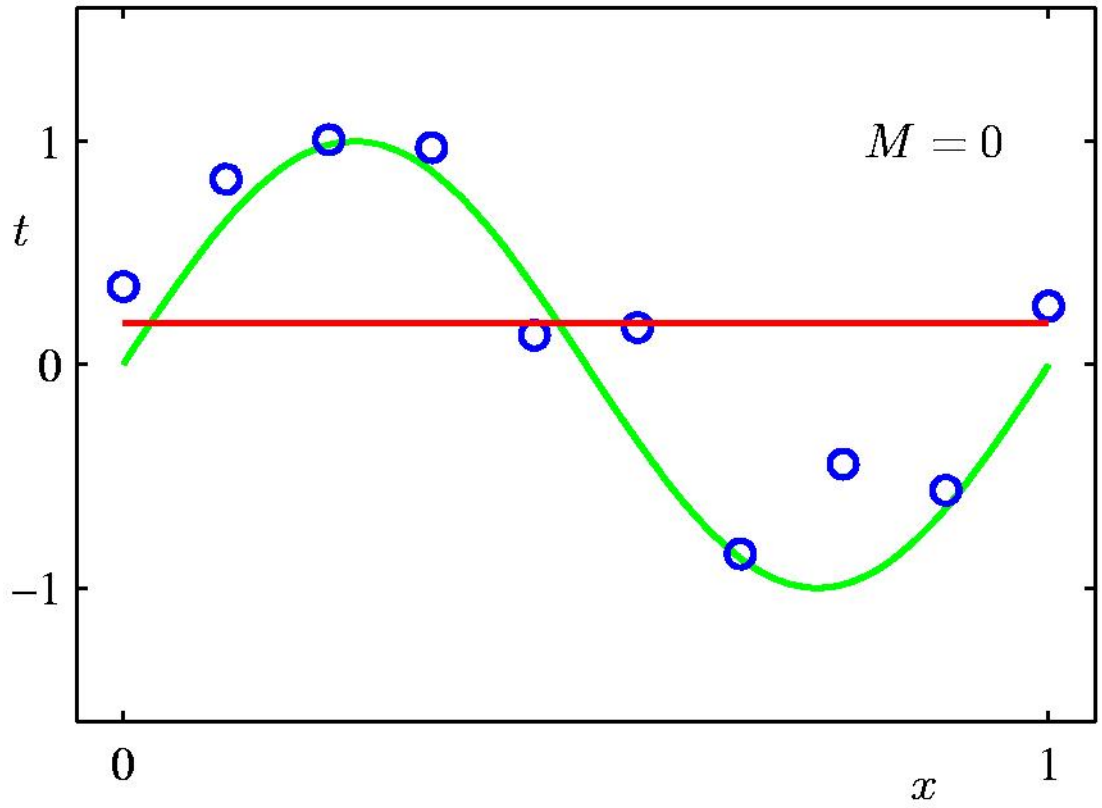


$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

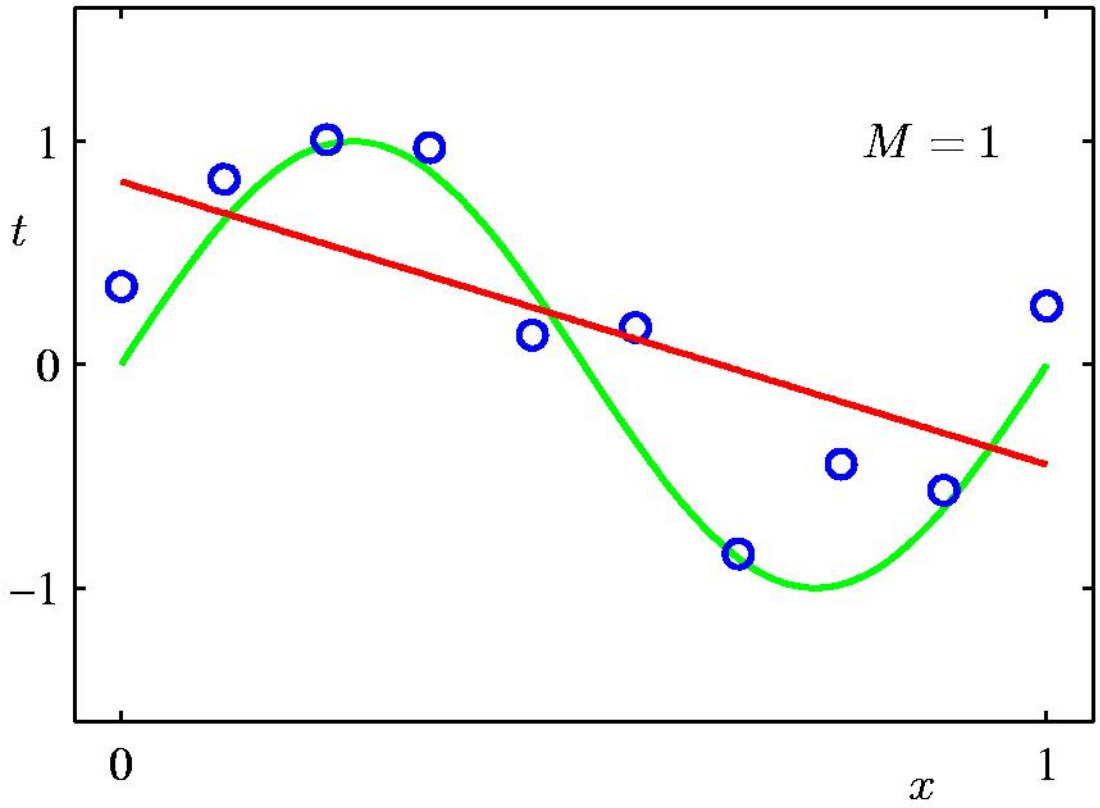
$$\sum_{n=1}^{N}\left(\sum_{j=0}^{M} w_j x_n^j - t_n\right) x_n^i = 0.$$

Closed-form solution to minimize *E* (take derivative w.r.t *w*, and set to 0 to find *w**)

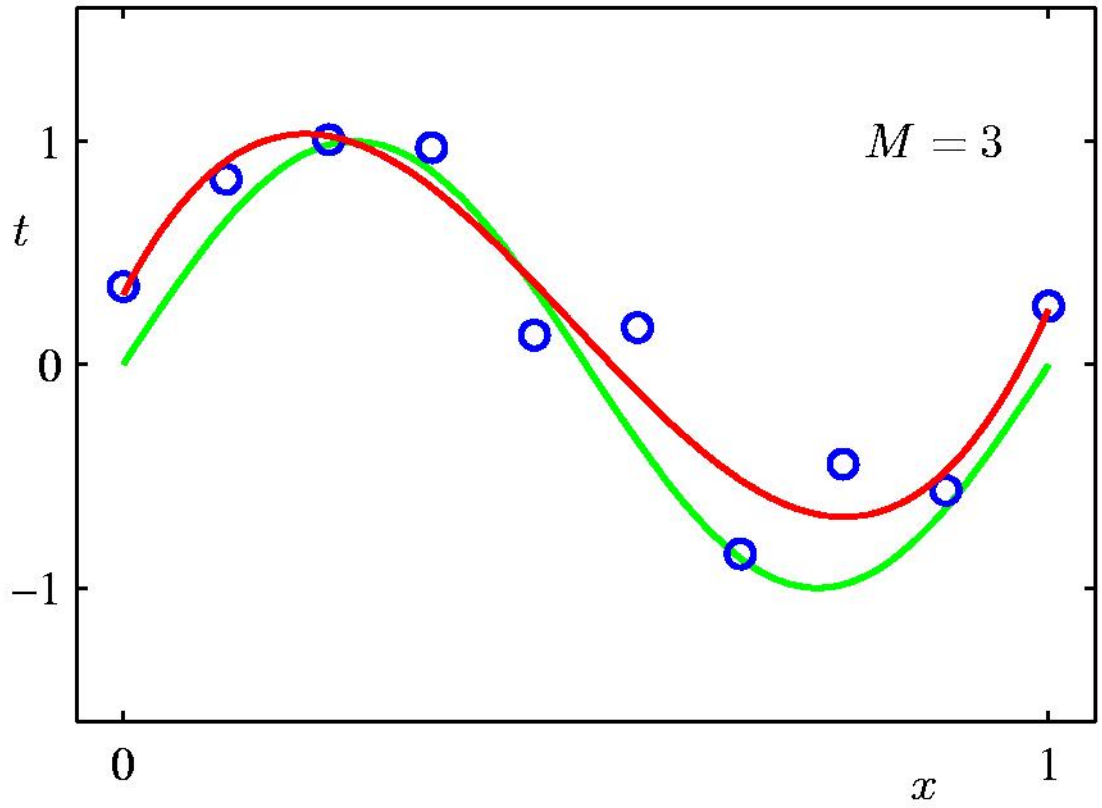
0^{th} Order Polynomial



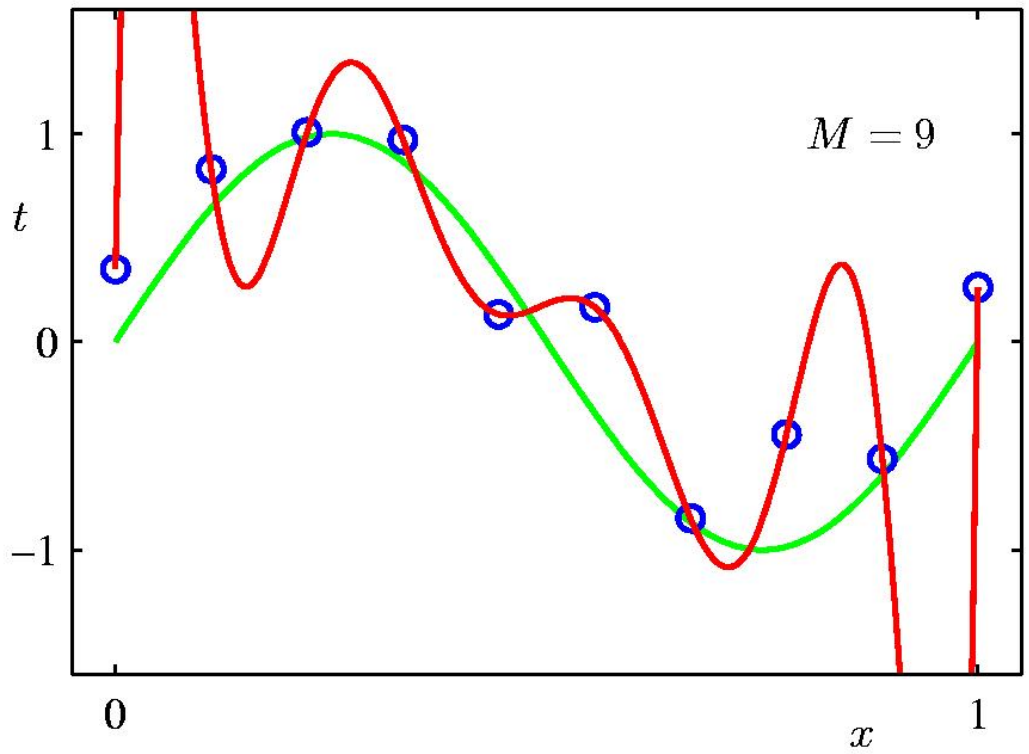
1st Order Polynomial



3rd Order Polynomial



9th Order Polynomial (Over-fitting)



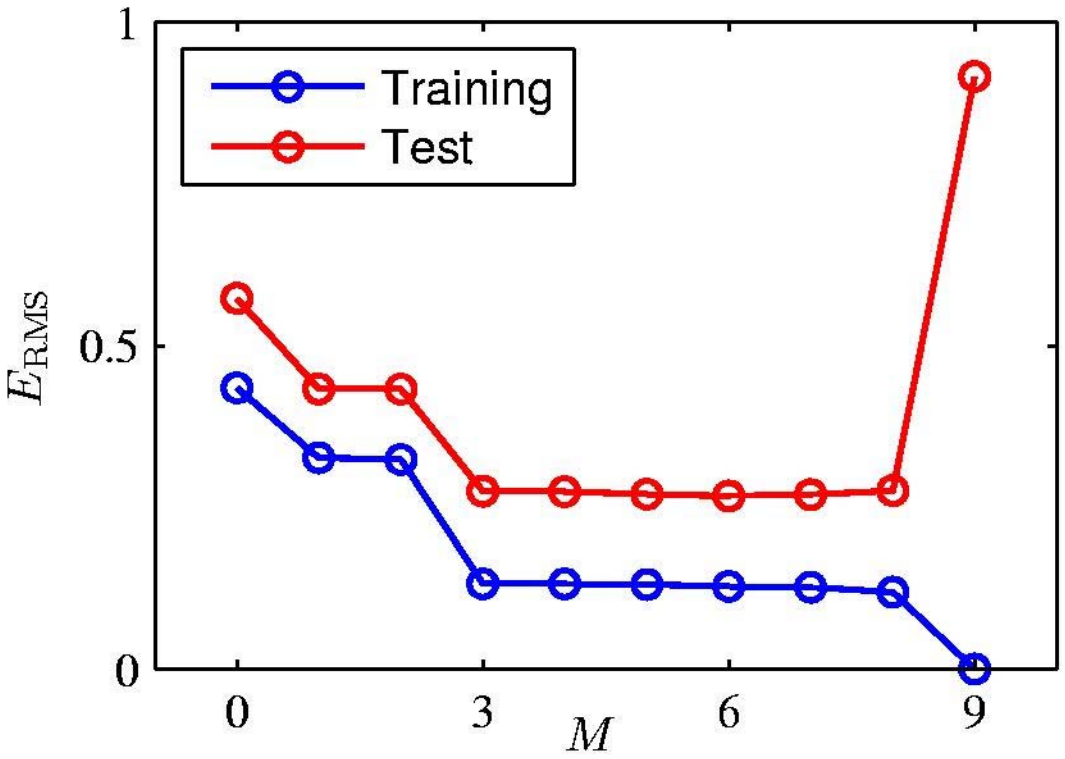
Curve oscillates wildly

Zero training error, but poor representation of data generating function

Poor prediction on new inputs!

Power series expansions of the function sin(2πx) contain terms of all orders, so shouldn't results improve monotonically as we increase M?

Over-fitting



Unintuitive?

- Higher-order polynomials include lower order ones
- Power expansion of sin contains terms of all orders

What's wrong then?

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

Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$

Division by *N* allows us to compare different sizes of data sets on equal footing, square root ensures that E_{RMS} is measured on the same scale (and in the same units) as the target variable *t*.

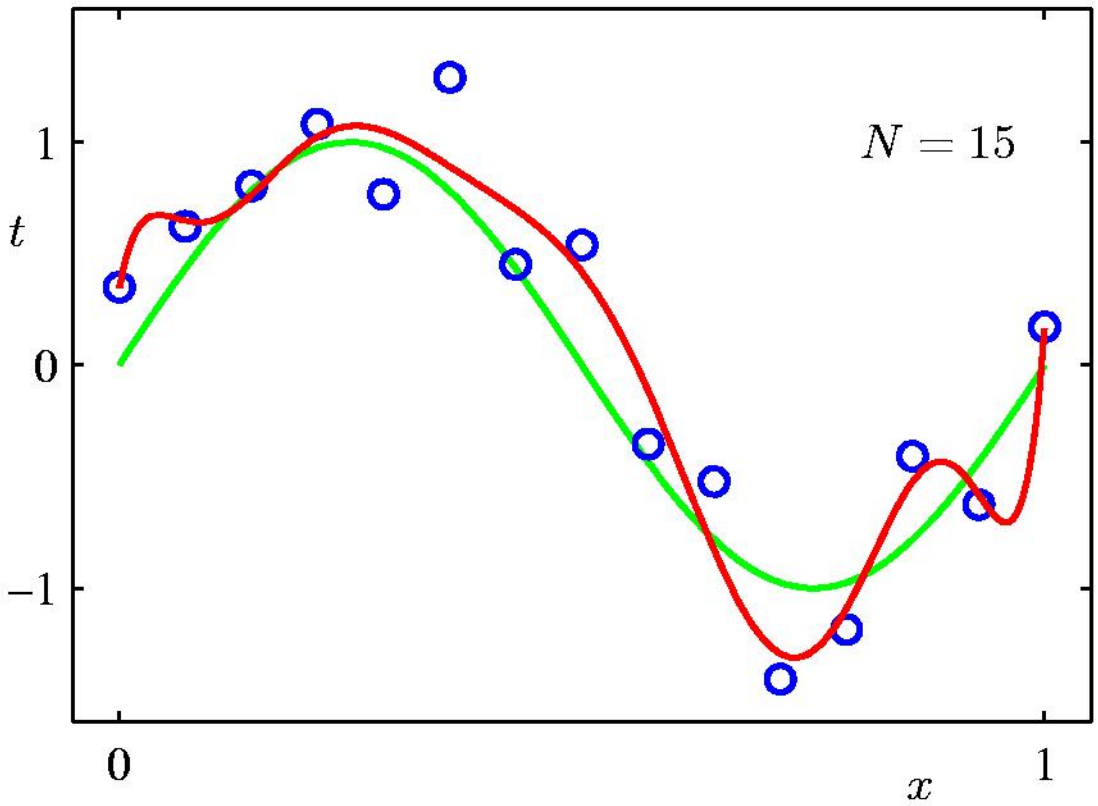
Polynomial Coefficients

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	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

Higher-order polynomials tune to random noise

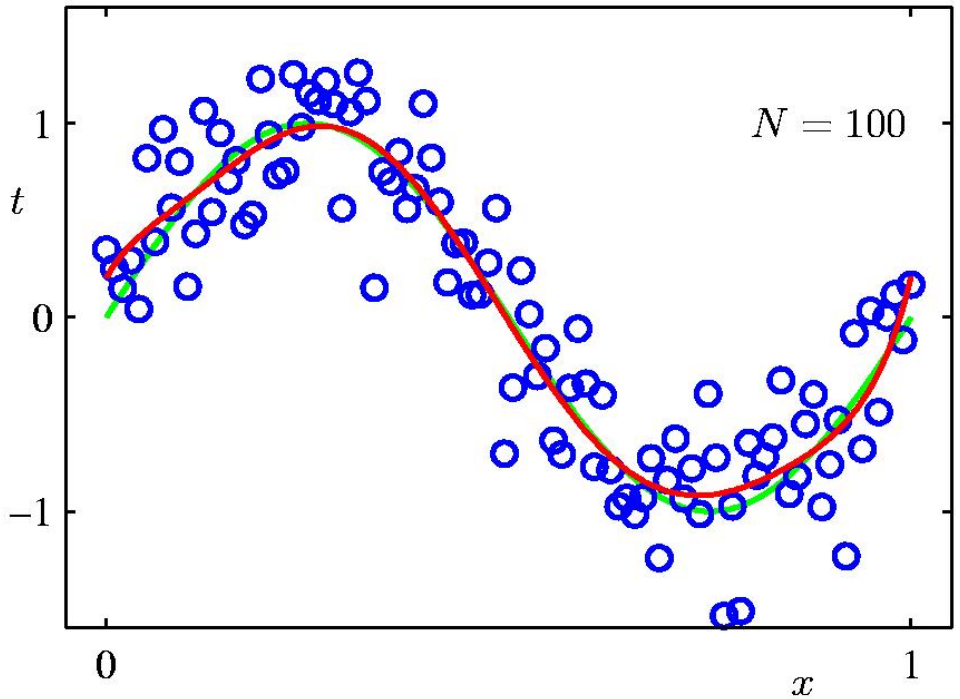
Data Set Size: $N = 15$

9th Order Polynomial



Data Set Size: $N = 100$

9th Order Polynomial



The larger the data set, the more complex (flexible) the model that we can afford to fit. One heuristic is that the number of data points should be no less than some multiple (say 5 or 10) of the number of adaptive parameters in the model.

More data helps, but it should be better to adjust model capacity based on the intrinsic complexity of the problem to be solved, not just the datset size

Spline Fitting: Regularization

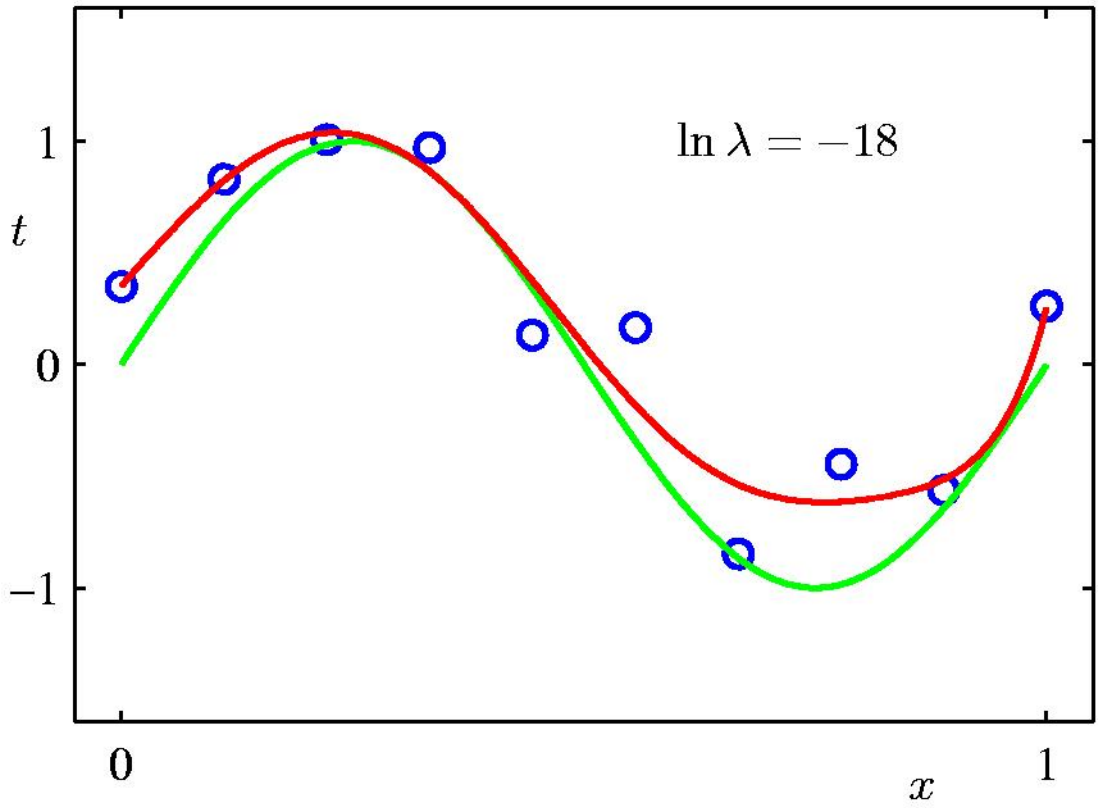
Penalize large coefficient values

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

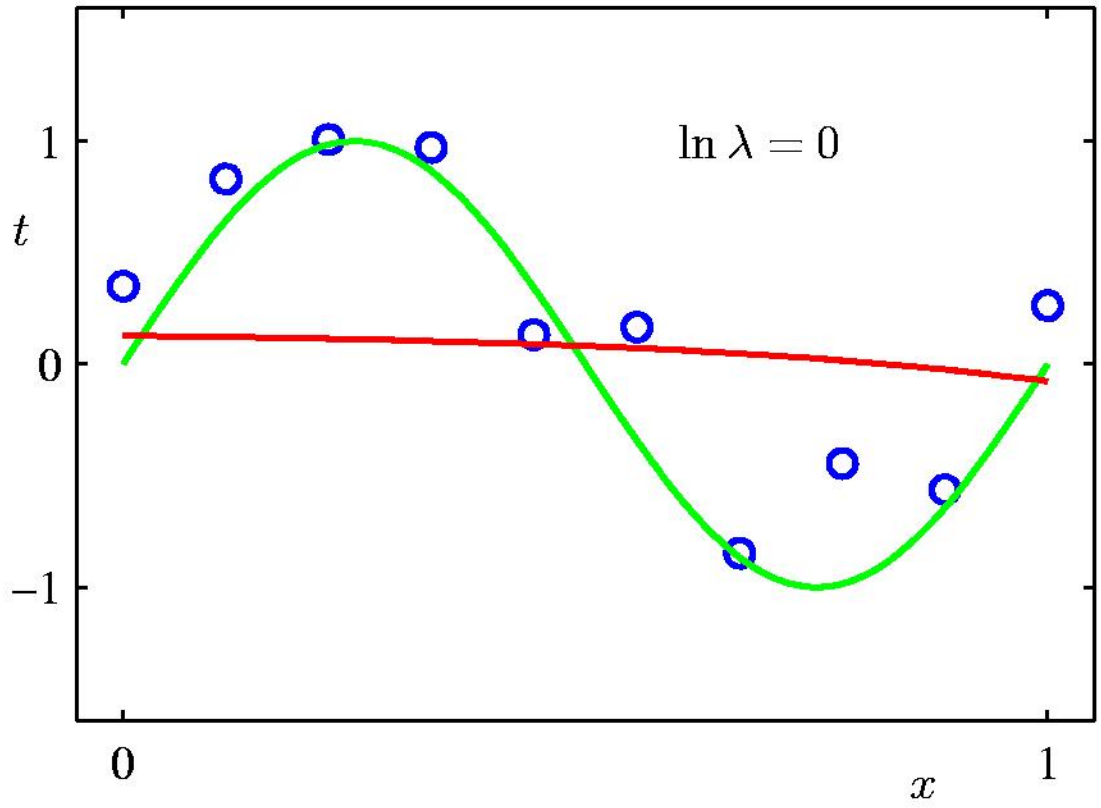
Shrinkage approaches

For quadratic regularizers: ridge regression, weight decay (neural networks)

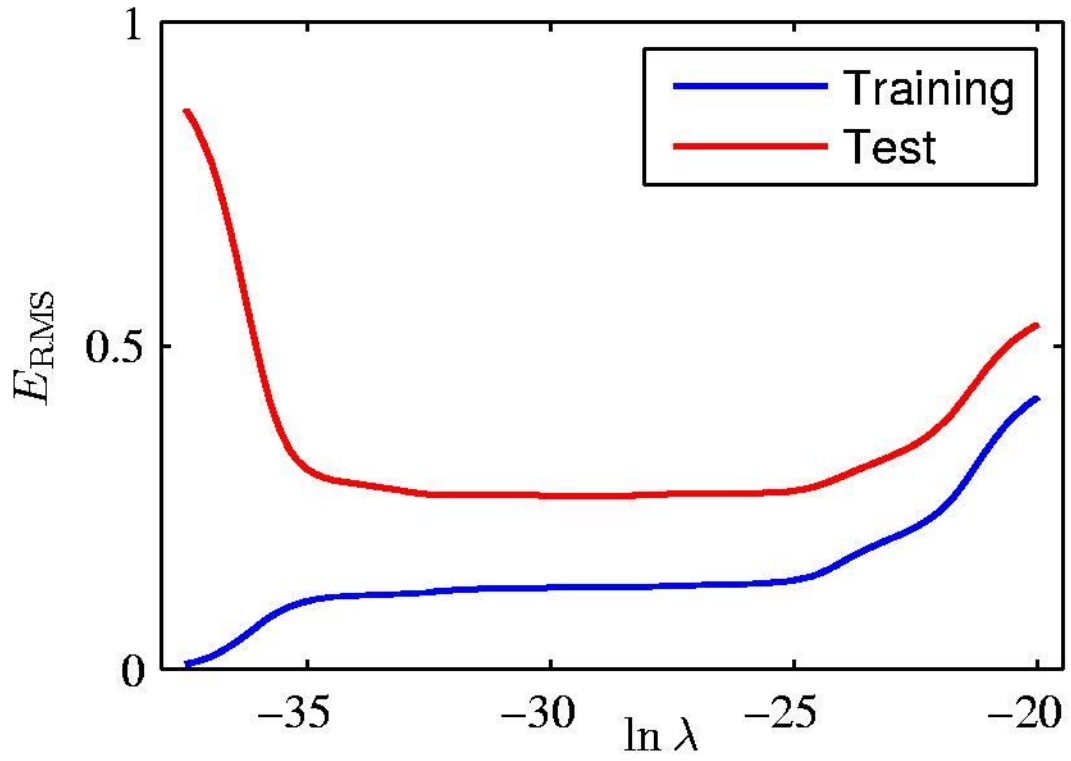
Regularization ($M = 9$): $\ln \lambda = -18$



Regularization ($M = 9$) : $\ln \lambda = 0$



Regularization: E_{RMS} vs. $\ln \lambda$



How to set λ ? Use validation or hold-out set (see later)

Polynomial Coefficients

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	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

Basic Concepts

Modeling with Random Variables

- We use random variables to encode the learning task
  - variables can take a set of possible values, each with an associated probability
  - inputs, outputs, internal states of learning machine

- Random variables may be discrete or continuous

  - discrete quantities take one of a fixed set of values
    e.g. {0,1}, {smile, not-smile}, {email, spam}

  - continuous quantities take real values
    e.g. elbow joint=30; temp=12.2; income=38,231

Hypothesis Spaces and Learning Machines

We can view learning as the process of exploring a *hypothesis space*

- We select functions from a specified set, the *hypothesis space*
- Hypothesis space is indexed by a set of *parameters* w**,** which are variables we can adjust (search) to create different machines
- In supervised learning, each hypothesis is a function that maps inputs to outputs
- A challenge of machine learning is deciding how to represent inputs and outputs, and how to select the hypothesis space that would be appropriate for a given task
- The central trade-off is in selecting a hypothesis space that is powerful enough to represent the relationships between inputs and outputs, yet simple enough to be searched efficiently

Loss Functions for Parameter Estimation

- Given an input and an output representation, and a hypothesis space, how do we set the machine parameters w?
- We will quantify performance on task defined by loss function $L(x, y, t, w)$
  - Inputs x, outputs of machine y, desired targets t, parameters w
- We will set the parameters w to minimize this average loss function measured empirically over some dataset

Possible Loss Functions

- Squared difference between predicted $y_n(x_n)$ and target (ground truth) real-valued outputs t_n

$$L = \sum_n \| t_n - y_n(x_n) \|^2$$

- Number of classification errors, makes optimization harder due to non-smooth derivative

$$L = \sum_n [t_n \neq y_n(x_n)]$$

- Negative log probability assigned to the correct answer
  - Principled, statistically justified
  - For some models, the same as squared error (regression with Gaussian output noise)
  - For other models, different expressions (cross-entropy error, for classification with discrete classes)

Training and Testing Expectations

- **Training data:** examples we are provided with
- **Testing data:** data we will see in the future
- **Training error:** average value of loss function on training data
  Test error: the average value of loss function on test data

- Our goal is, primarily, not to do well on training data
  We already have the answers (outputs) for that data

- We want to perform well on *future unseen data*
  We wish to minimize the test error

- How can we guarantee it, if we do not have test data?
  We will rely on probabilistic assumptions on data variability

I.I.D. Sampling Assumption

- Assume that data is generated randomly, from a joint, **unknown** probability distribution $p(x, t)$

- We are given a finite, possibly noisy, training sample

$$\{(x_1, t_1), \ldots, (x_N, t_N)\}$$

with elements generated **i.i.d.** i.e. independently from the same distribution $p(x, t)$

independently: training examples are drawn independently from the set of all possible examples

identically **d**istributed: each time a training example is drawn, it comes from an identical distribution

Training and Testing Process

- **In training**, based only on the training data, construct a machine that generates outputs given inputs
  - One option is to build machines with small training loss
  - Ideally we wish the machine to model the main regularities in the data and ignore the noise. However, if the machine has as many degrees of freedom as the data, it can fit perfectly. We saw the polynomial case study
  - Avoiding this usually requires model complexity control (regularization)

- **In testing**, a new sample is drawn i.i.d. from the same distribution as the training data
  - This assumption makes it unlikely that important regularities in the test data were missed in the training data
  - We run the machine on new sample and evaluate loss: this is the test error

Overfitting and Generalization

- The central problem with any learning setup is to do well on the training data but poorly on test data.

  This is called *overfitting*

- This has to do with the set of assumptions about the target function made by the learning algorithm, known as its *inductive bias*
  - complexity of hypothesis space, regularization, etc.

- The ability of a learning machine to achieve a small loss on test data is called *generalization*

- *Generalization is possible*, in principle, given our i.i.d. assumption: if a machine performs well on most of a sufficiently large training set, and is not too complex, it is likely to do well on similar test data

Probabilistic Guarantees

$$E_{test} \leq E_{train} + \left(\frac{h + h\log(2N/h) - \log(p/4)}{N} \right)^{\frac{1}{2}}$$

where N = size of training set
h = VC dimension of the model class = complexity
p = upper bound on probability that this bound fails

If we train models of different complexities, we should ideally select the one that minimizes the type of bound above

We do not have such bounds for all models, and even when we do, the procedure would be effective only if the bounds are tight. But in general they are not. The theory offers insight, but using such results in practice will not be straightforward

adapted from MLG@Toronto

General Objective Functions

- The general structure of our learning objective function is

$$f(x, y, t, w) = L(x, y, t, w) + R(w)$$

  L is the loss function, and R is a regularizer (penalty, or prior over functions), which discourages overly complex models

- **Intuition**
  - It is good to fit the data well and achieve low training loss
  - But it is also good to bias the machine towards simpler models, in order to avoid overfitting

- Setup allows to decouple optimization from choice of training loss

Bayesian view of Hypothesis Spaces

- So far, we have seen how we can formulate a loss function and then adjust the parameters of the machine to minimize it
  - Single (point) estimate for the parameters of the learning machine

- In Bayesian modeling, we do not search for only a single set of parameter values that optimize the loss function
  - Instead, we start with a prior distribution over parameter values and use the training data to compute a posterior distribution over the entire parameter space
  - We use this distribution for prediction on new test data

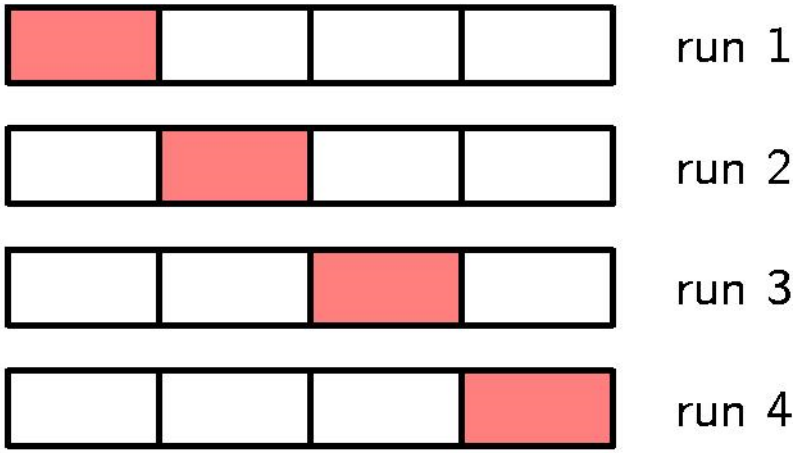
Practice: Use of a Validation Set

- Divide the available data into **three subsets**
  - **Training data** is used to learn the parameters of the model
  - **Validation data** is used to decide the type of model and the amount of regularization that works best
  - **Testing data** is used to get a final, unbiased estimate of how well the learning machine works
- We can re-divide the complete dataset and get another unbiased estimate of the true error rate. We can then average multiple estimates to decrease variance. This can be expensive

adapted from MLG@Toronto

Model Selection

Cross-Validation (rotated estimation)



One round of cross-validation involves partitioning a sample of the data into complementary subsets, estimate the model on one subset (called the *training/validation set*), and test it on the other subset (called the *testing set*). To reduce variance, multiple rounds of cross-validation are performed using different partitions, and results are averaged over all rounds.

Readings

- Bishop, Ch. 1 (without 1.2.6 and 1.6) and from Ch. 2, only 2.3.1 - 2.3.3.