Principles of Robot Autonomy II

Course overview and intro to

machine learning for robot autonomy

Stanford University

iliad

IPRL

ASL Autonomous Systems Lab

Team

Instructors

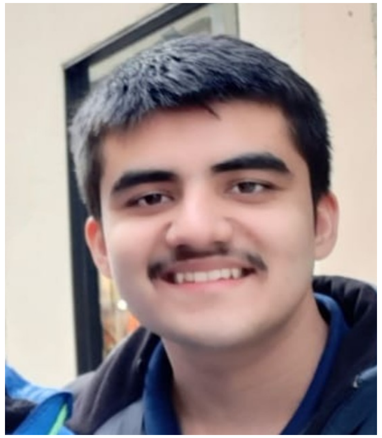
Jeannette Bohg
Assistant Professor CS

Marco Pavone
Associate Professor AA, and CS/EE (by courtesy)

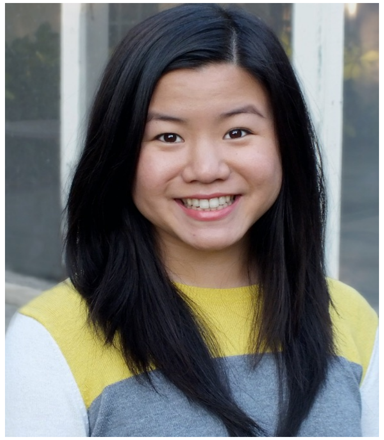
Dorsa Sadigh
Assistant Professor CS and EE

CAs

Tanmay Agarwal



Abhyudit Manhas



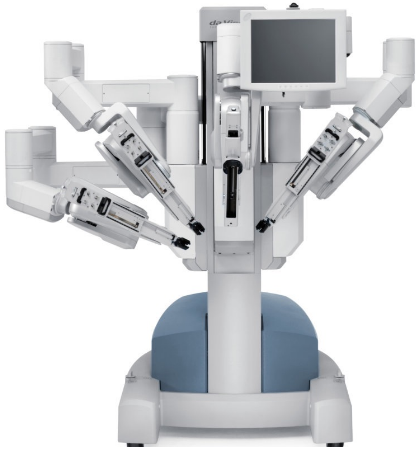
Claire Chen

From automation…

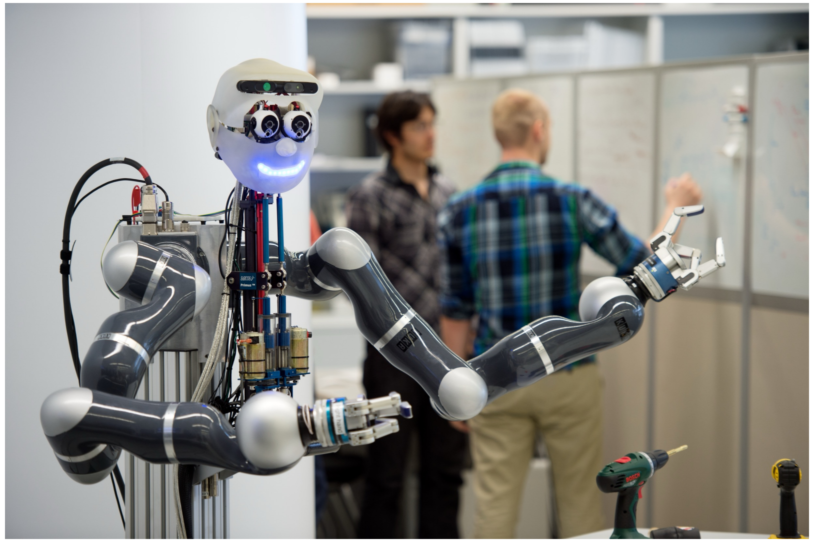
...to autonomy

Waymo Self-Driving Car

Intuitive DaVinci Surgical Robot



Apollo Robot at MPI for Intelligent Systems

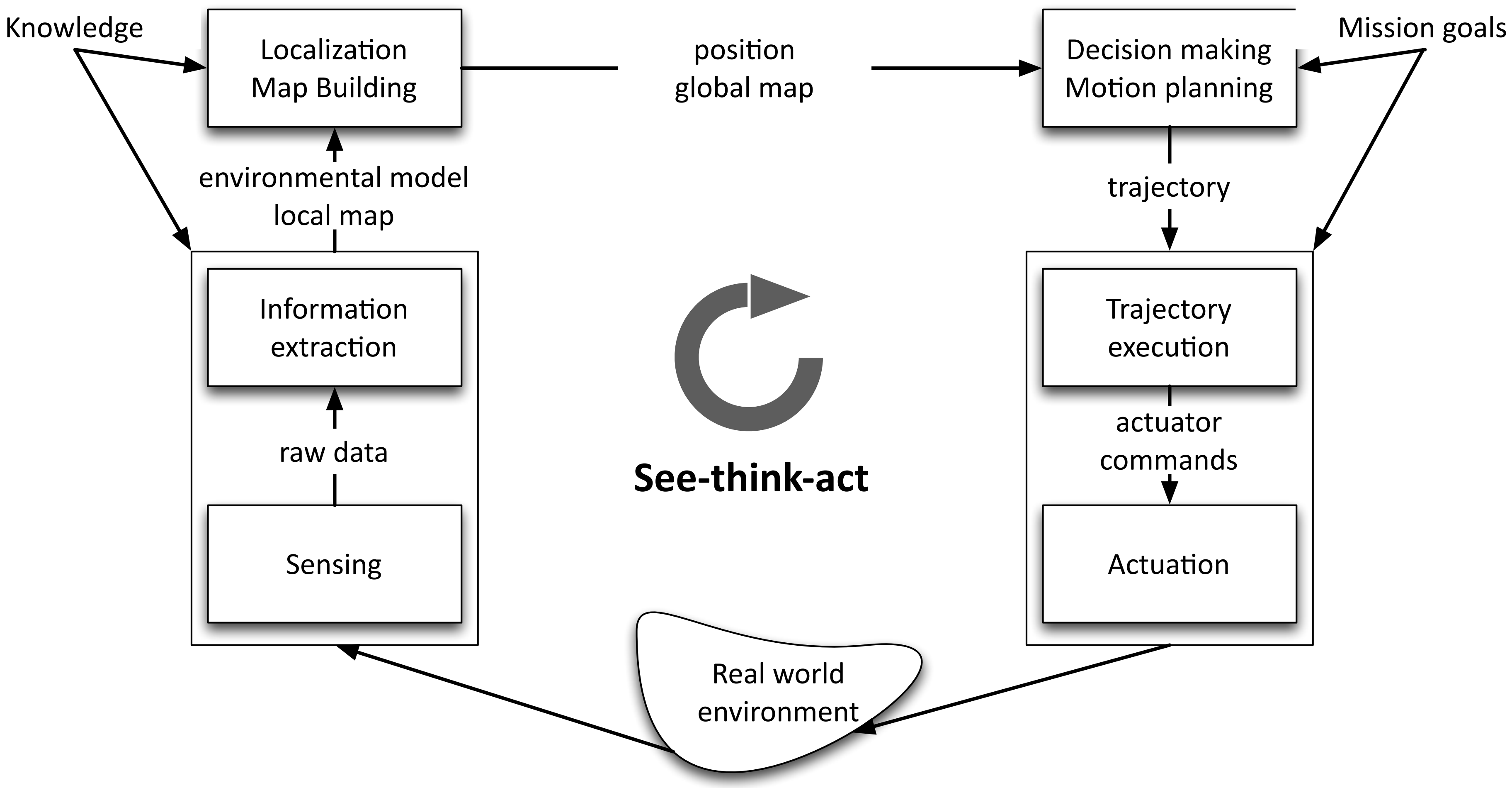


Boston Dynamics – Spot Mini

Astrobee - NASA

Zipline

From Principles of Robot Autonomy I: the see-think-act cycle



Outstanding questions and new trends

- How do we build models for complex tasks? Can we use data / prior experience?
- How should the robot reason in terms of actively interacting with the environment?
- And how should the robot reason when interacting with other decision-making agents?

Course goals

- Obtain a fundamental understanding of *advanced* principles of robot autonomy, including:
  1. robot learning
  2. physical interaction with the environment
  3. interaction with humans

Course structure

- Three modules, roughly of equal length
  1. learning-based control and perception
  2. interaction with the physical environment
  3. interaction with humans

- Requirements
  - AA 174A / AA 274A / CS 237A / EE 260A
  - CS 106A or equivalent, CS106B highly recommended
  - CME 100 or equivalent (for linear algebra)
  - CME 106 or equivalent (for probability theory)

Logistics

- Lectures: Monday and Wednesday, 1:30pm - 2:50pm
- Information about office hours available in the Syllabus: https://web.stanford.edu/class/cs237b/pdfs/syllabus.pdf
- Course websites:
  - https://cs237b.stanford.edu (course content and announcements)
  - https://canvas.stanford.edu/courses/182770 (course-related discussions)
  - https://www.gradescope.com/courses/689252 (HW submissions)
  - https://canvas.stanford.edu/courses/182770 (Panopto Course Videos)
- To contact the teaching staff, use the email: cs237b-win2324-staff@lists.stanford.edu

Grading and units

- Course grade calculation
  - (60%) homework
  - (40%) exams (for each student, the lowest exam grade will be dropped)
  - (extra 5%) participation on EdStem

- Units: 3 or 4. Taking this class for 4 units entails additionally presenting a paper at the end of the quarter

Schedule

Date	Topic	Assignment
01/08	Course Overview, Intro to ML for Robotics	
01/10	Neural Networks and TensorFlow Tutorial	
01/12		HW1 out
01/15	Martin Luther King, Jr. Day (no classes)	
01/17	Markov Decision Processes	
01/22	Intro to RL	
01/24	Model-based and Model-free RL for Robot Control	
01/29	Learning-based Perception	
01/31	Fundamentals of Grasping and Manipulation (1)	
02/02		HW1 due, HW2 out
02/05	Fundamentals of Grasping and Manipulation (2)	
02/07	Learning-based Grasping and Manipulation	
02/09		Exam 1
02/12	Interactive Perception	
02/14	Imitation Learning (1)	
02/16		HW2 due, HW3 out
02/19	President's Day (no classes)	
02/21	Guest Lecture (TBD)	
02/23		Exam 2
02/26	Imitation Learning (2)	
02/28	Learning from Human Feedback	
03/04	Interaction-Aware Learning, Planning, and Control	
03/06	Shared Autonomy	
03/08		HW3 due
03/11	Guest Lecture (Sidd Karamcheti)	
03/13	Paper Presentations	
03/15		Exam 3

Intro to Machine Learning (ML)

- Aim
  - Present and motivate modern ML techniques

- Courses at Stanford
  - EE 104: Introduction to Machine Learning
  - CS 229: Machine Learning

- Reference
  - Hastie, Tibshirani, and Friedman: The elements of statistical learning: data mining, inference, and prediction (2009). Available here: https://web.stanford.edu/~hastie/ElemStatLearn/

Machine learning

- Supervised learning (classification, regression)
  - Given $(x^1, y^1), \dots, (x^n, y^n)$, choose a function $f(x) = y$

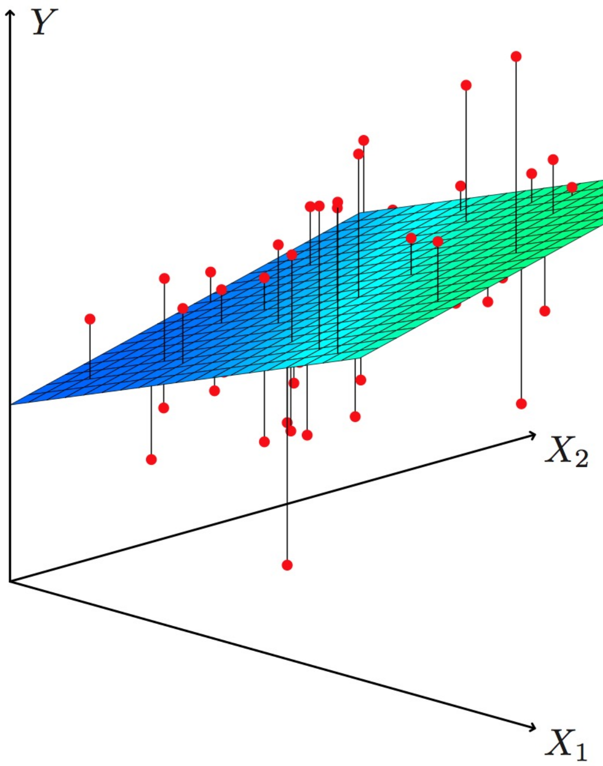
$$x_i = \text{data point}$$

$$y_i = \text{class/value}$$

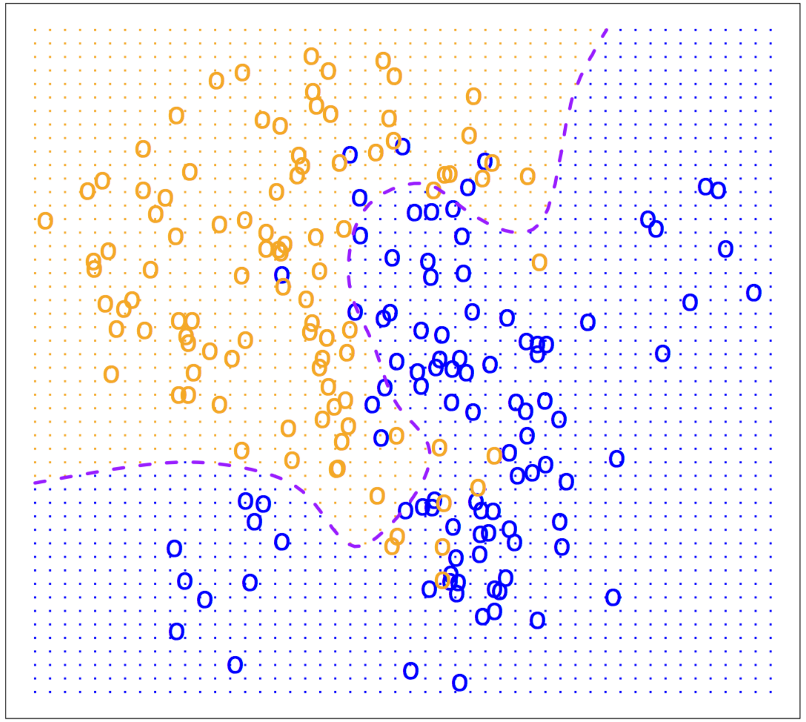
- Unsupervised learning (clustering, dimensionality reduction)
  - Given $(x^1, x^2, \dots, x^n)$ find patterns in the data

Supervised learning

- Regression

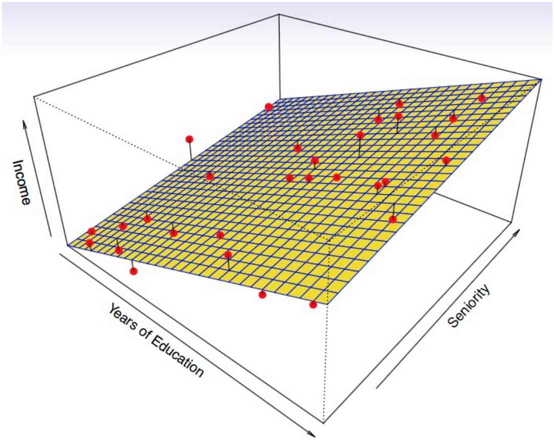


- Classification

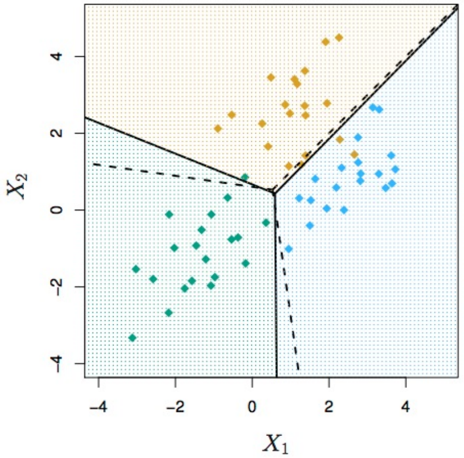


Learning models

Parametric models

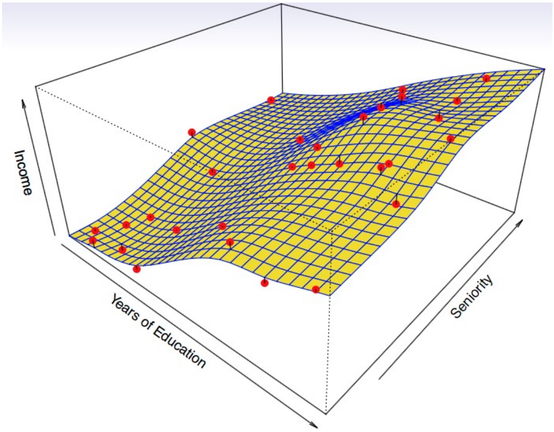


Linear regression

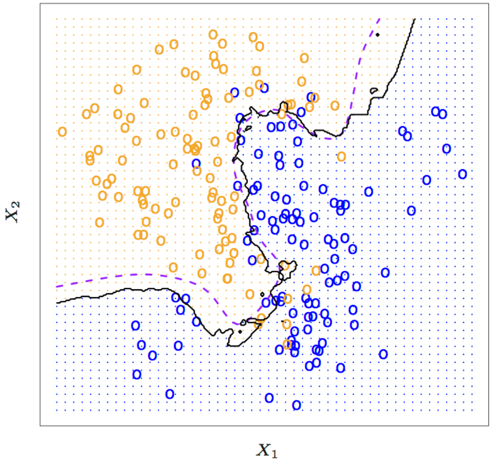


Linear classifier

Non-parametric models



Spline fitting



k-Nearest Neighbors

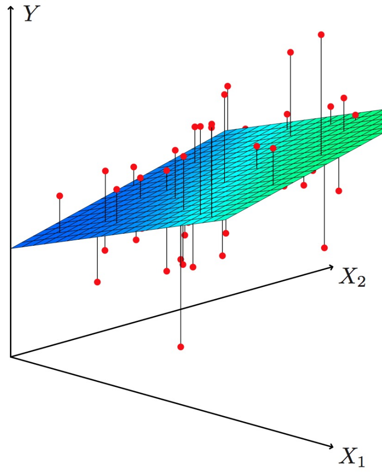
Loss functions

In selecting $f(x) \approx y$ we need a quality metric, i.e., a loss function to minimize

- Regression

$$\ell^2 \text{ loss} : \quad \sum_i |f(x^i) - y^i|^2$$

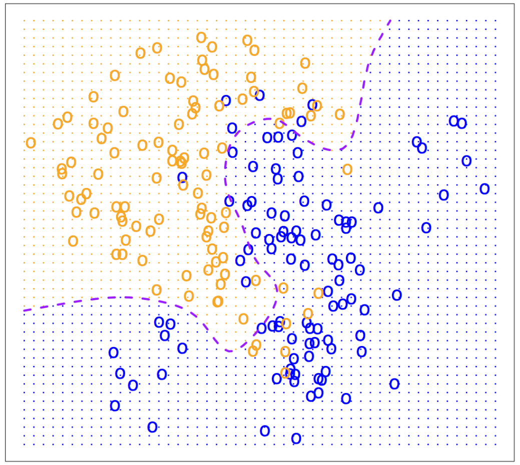
$$\ell^1 \text{ loss} : \quad \sum_i |f(x^i) - y^i|$$



- Classification

$$0 - 1 \text{ loss} : \quad \sum_i \mathbf{1}\{f(x^i) \neq y^i\}$$

$$\text{Cross entropy loss} : \quad -\sum_i (y^i)^T \log(f(x^i))$$



Machine learning as optimization

How can we choose the best (loss minimizing) parameters to fit our training data?*

Analytical solution

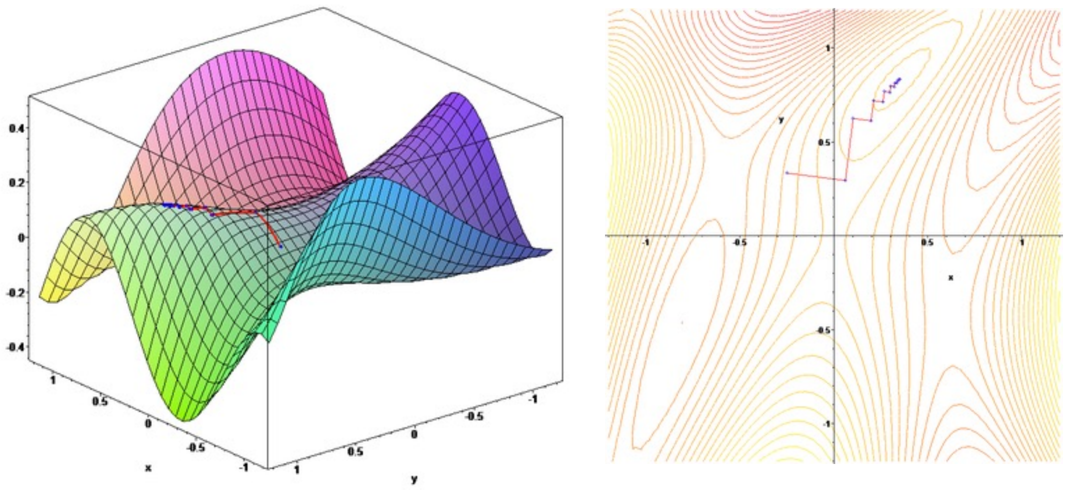
$$\begin{bmatrix} y_1^1 & y_2^1 \\ y_1^2 & y_2^2 \\ \vdots & \\ y_1^n & y_2^n \end{bmatrix} \approx \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_k^1 \\ x_1^2 & x_2^2 & \cdots & x_k^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^n & x_2^n & \cdots & x_k^n \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ a_{11} & a_{12} \\ \vdots & \vdots \\ a_{k1} & a_{k2} \end{bmatrix}$$

$$f_A(x) = xA, \quad \ell^2 \text{ loss}$$

$$\hat{A} = (X^T X)^{-1} X^T Y$$

(example: linear least squares)

Numerical optimization



(example: gradient descent)

* we'll come back to worrying about test data

Stochastic optimization

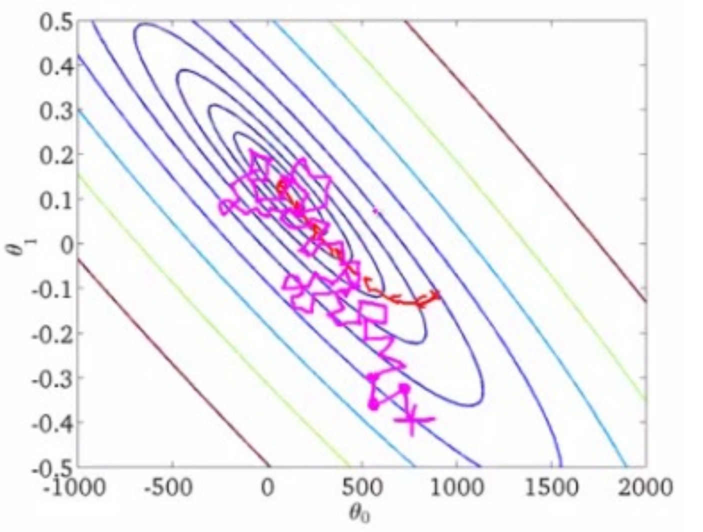
Our loss function is defined over the entire training dataset:

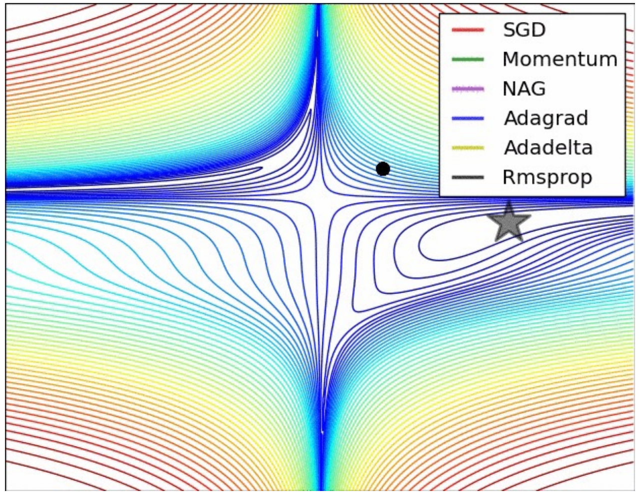
$$L = \frac{1}{n}\sum_{i=1}^{n}\left|f(x^i) - y^i\right|^2 = \frac{1}{n}\sum_{i=1}^{n} L_i$$

Computing ∇L could be very computationally intensive. We approximate:

$$\nabla L \approx \frac{1}{|S|}\sum_{i\in S\subset\{1,\ldots,n\}} \nabla L_i$$

Stochastic gradient descent





Other variants

Regularization

To avoid overfitting on the training data, we may add additional terms to the loss function to penalize "model complexity"

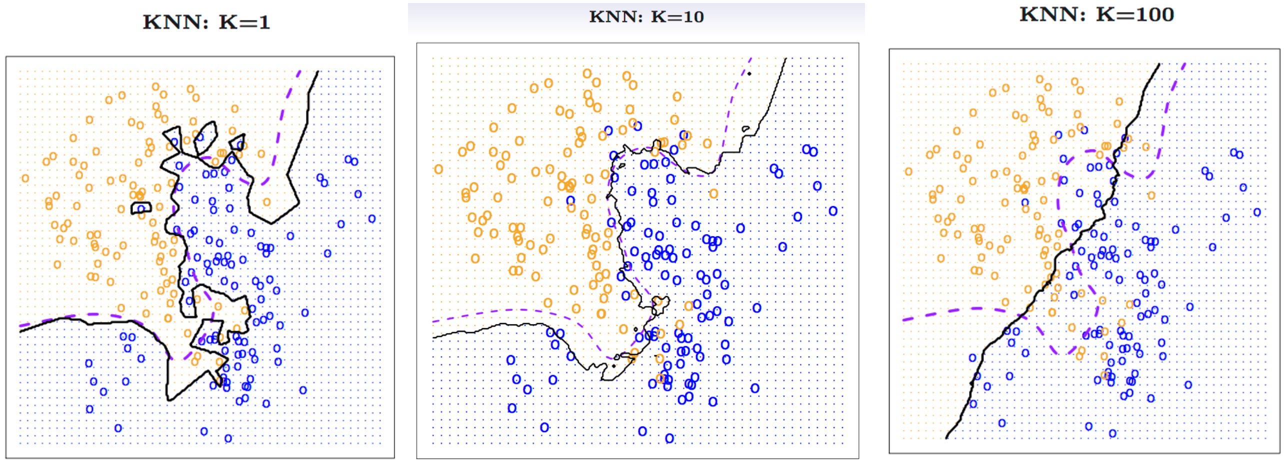
ℓ^2 regularization: $\|A\|_2$

often corresponds to a Gaussian prior on parameters *A*

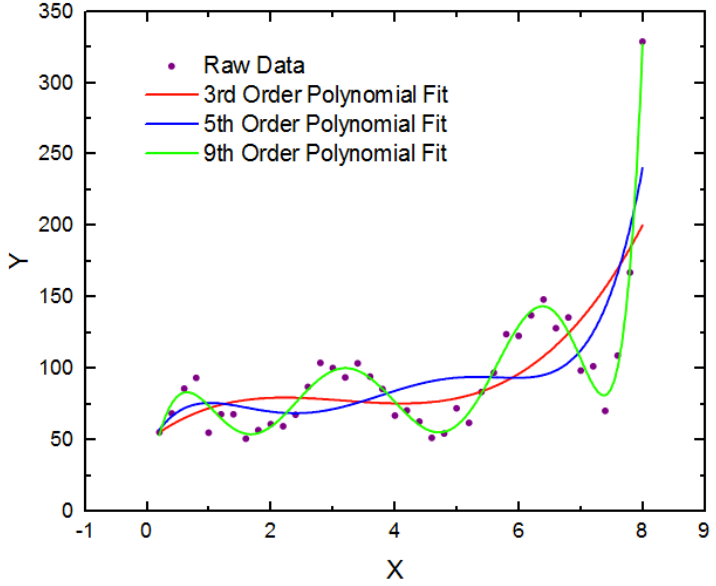
ℓ^1 regularization: $\|A\|_1$

often encourages sparsity in *A* (easier to interpret/explain)

Hyperparameter regularization:



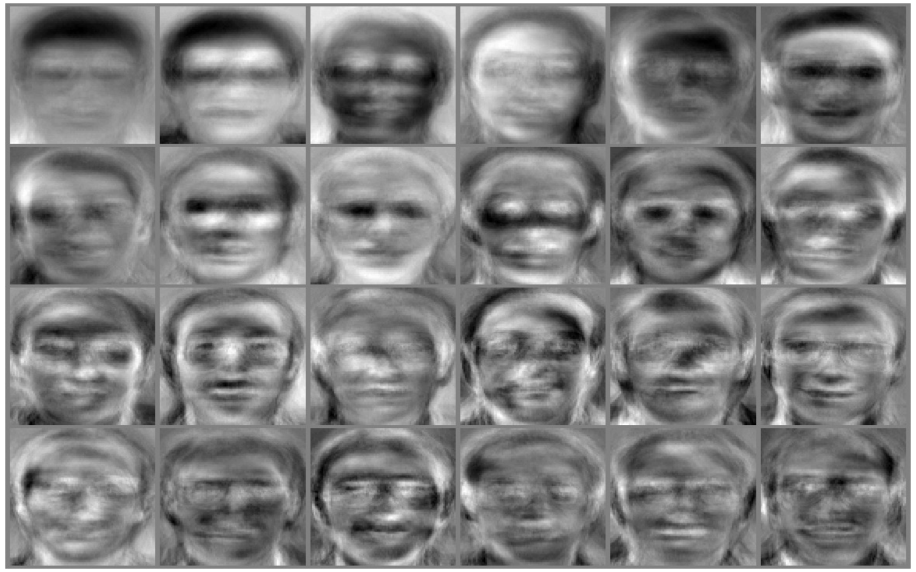
Generalizing linear models



$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} \approx \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ 1 & x_3 & x_3^2 & \dots & x_3^m \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}$$

Nonlinearity via basis functions

Linear regression/classification can be very powerful when empowered by the right features



Eigenfaces

Feature extraction

vector describing various
image statistics

Human
Ingenuity

[32x32x3]

Feature
Extraction

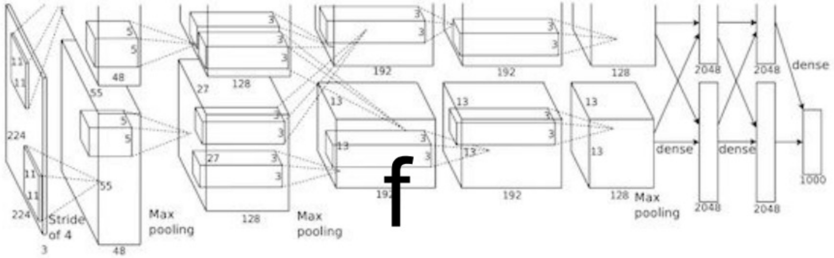
f

10 numbers, indicating
class scores

training

Gradient
Descent

[32x32x3]



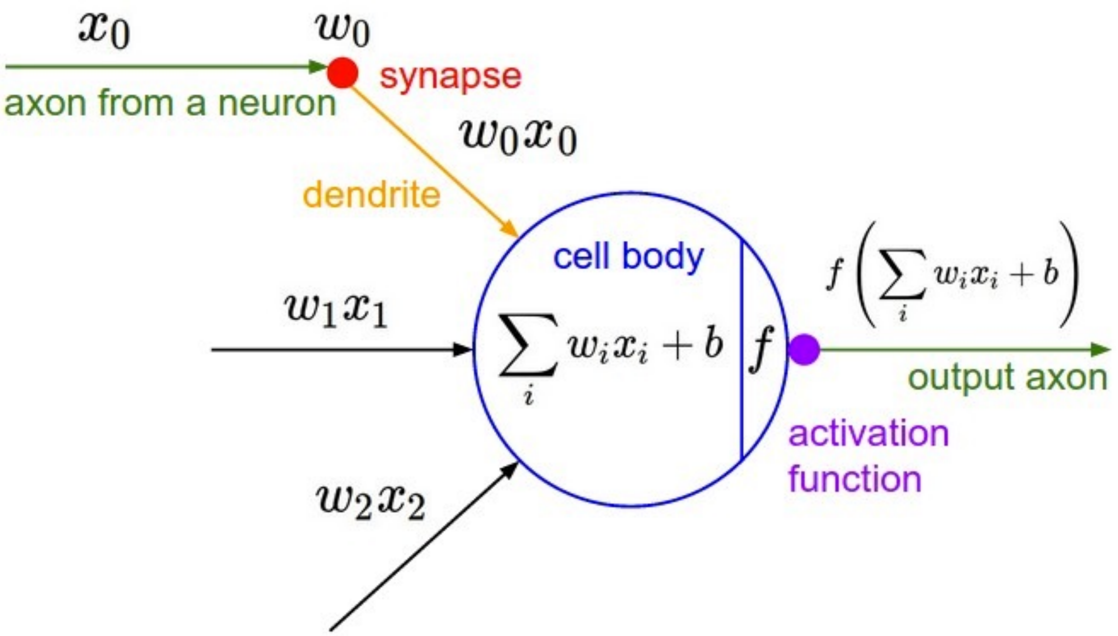
f

10 numbers, indicating
class scores

training

Next time

NNs and TensorFlow Tutorial



TensorFlow