CS 7301
Advanced Machine Learning

Nicholas Ruozzi
University of Texas at Dallas

Slides adapted from David Sontag and Vibhav Gogate

Course Info.

- Instructor: Nicholas Ruozzi
  - Office: ECSS 2.203
  - Office hours: Tues. 10am-11am
- TA: ?
  - Office hours and location ?
- Course website: www.utdallas.edu/~nrr150130/cs7301/2016fa/

Prerequisites

- "Mathematical sophistication"
  - Basic probability
  - Linear algebra
    - Eigenvalues, eigenvectors, matrices, vectors, etc.
  - Multivariate calculus
    - Derivatives, integration, gradients, Lagrange multipliers, etc.
- I'll review some concepts as we come to them, but you should brush up in areas that you aren't as comfortable

Grading

- 5-6 problem sets (50%)
  - See collaboration policy on the web
  - Mix of theory and programming (in MATLAB)
  - Available and turned in on eLearning
  - Approximately one assignment every two weeks
- Midterm Exam (20%)
- Final Exam (30%)

-subject to change-

Course Topics

- Dimensionality reduction
  - PCA
  - Matrix Factorizations
- Learning
  - Supervised, unsupervised, active, reinforcement, …
  - Learning theory: PAC learning, VC dimension
  - SVMs & kernel methods
  - Decision trees, k-NN, …
  - Parameter estimation: Bayesian methods, MAP estimation, maximum likelihood estimation, expectation maximization, …
  - Clustering: k-means & spectral clustering
- Graphical models
  - Neural networks
  - Bayesian networks: naïve Bayes
- Statistical methods
  - Boosting, bagging, bootstrapping
  - Sampling
- Ranking & Collaborative Filtering

What is ML?

What is ML?

"A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E."

- Tom Mitchell

Basic Machine Learning Paradigm

- Collect data
- Build a model using "training" data
- Use model to make predictions

Supervised Learning

- Input: $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)$
  - $x^{(i)}$ is the i^{th} data item and $y^{(i)}$ is the i^{th} **label**
- Goal: find a function f such that $f\left(x^{(i)}\right)$ is a "good approximation" to $y^{(i)}$
  - Can use it to predict y values for previously unseen x values

Examples of Supervised Learning

- Spam email detection
- Handwritten digit recognition
- Stock market prediction
- More?

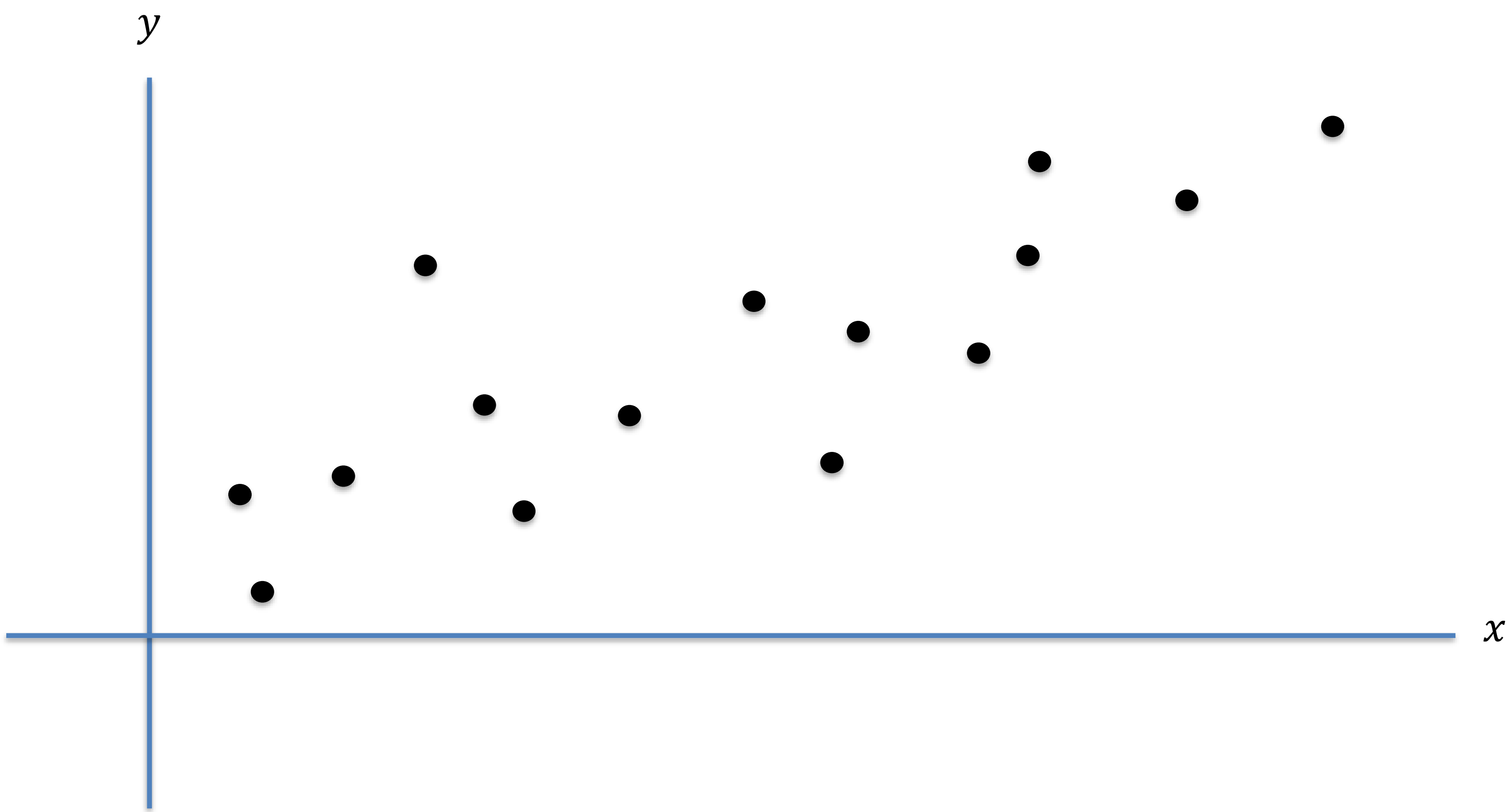
Supervised Learning

- Hypothesis space: set of allowable functions $f : X \to Y$
- Goal: find the "best" element of the hypothesis space
  - How do we measure the quality of f?

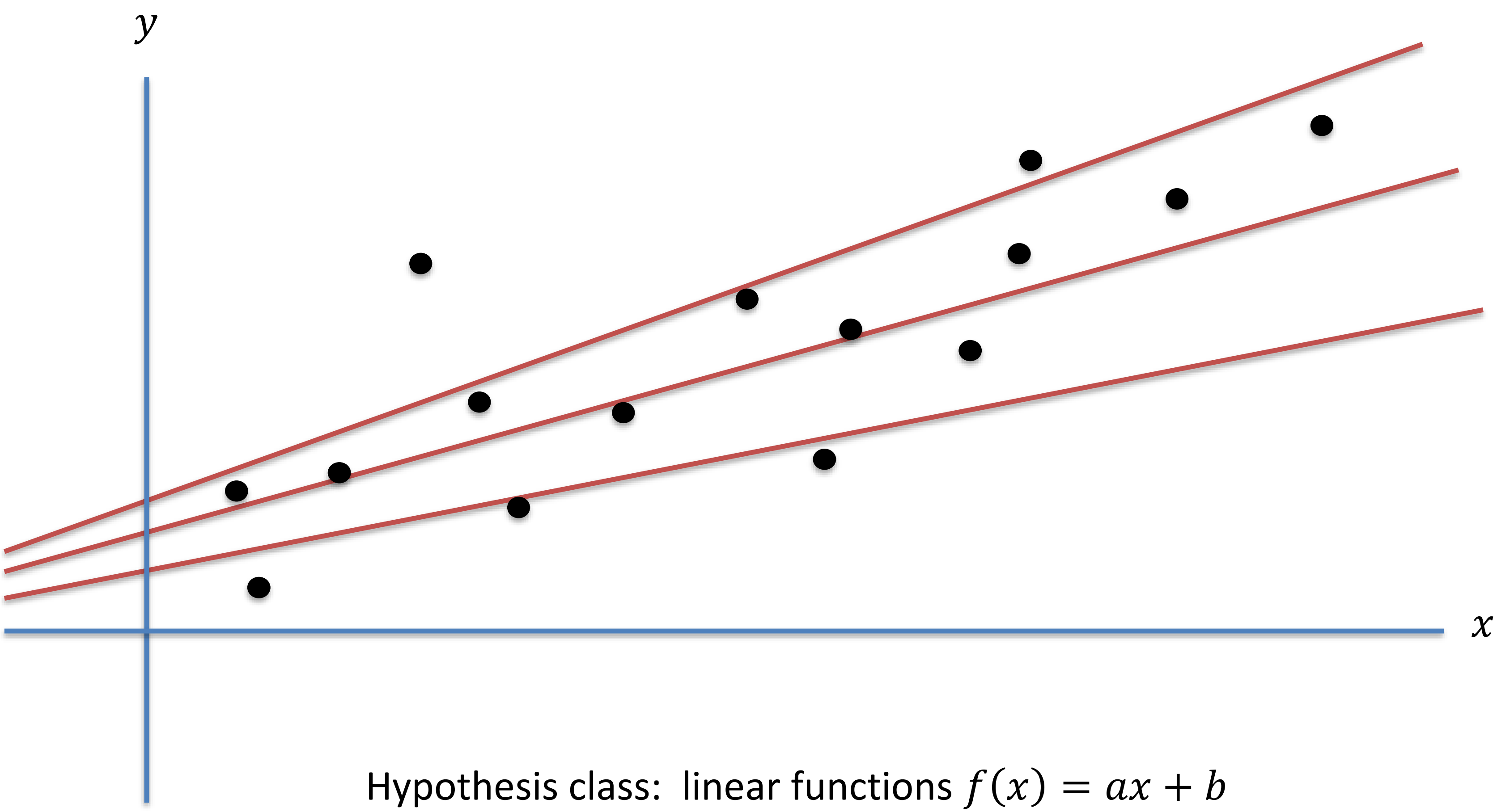
Types of Learning

- Supervised
  - The training data includes the desired output
- Unsupervised
  - The training data does not include the desired output
- Semi-supervised
  - Some training data comes with the desired output
- Active learning
  - Semi-supervised learning where the algorithm can ask for the correct outputs for specifically chosen data points
- Reinforcement learning
  - The learner interacts with the world via allowable actions which change the state of the world and result in rewards
  - The learner attempts to maximize rewards through trial and error

Regression



Regression



Hypothesis class: linear functions $f(x) = ax + b$

How do we measure the quality of the approximation?

Linear Regression

- In typical regression applications, measure the fit using a squared loss function

$$L(f, y_i) = \left(f\left(x^{(i)}\right) - y^{(i)}\right)^2$$

- Want to minimize the average loss on the training data

- For 2-D linear regression, the learning problem is then

$$\min_{a,b} \frac{1}{n} \sum_i \left(ax^{(i)} + b - y^{(i)}\right)^2$$

- For an unseen data point, x, the learning algorithm predicts $f(x)$

Linear Regression

$$\min_{a,b} \frac{1}{n} \sum_i \left(ax^{(i)} + b - y^{(i)}\right)^2$$

- How do we find the optimal a and b?

Linear Regression

$$\min_{a,b} \frac{1}{n} \sum_i \left(ax^{(i)} + b - y^{(i)}\right)^2$$

- How do we find the optimal a and b?
  - Solution 1: take derivatives and solve (there is a closed form solution!)
  - Solution 2: use gradient descent

Linear Regression

$$\min_{a,b} \frac{1}{n} \sum_i \left(ax^{(i)} + b - y^{(i)}\right)^2$$

- How do we find the optimal a and b?
  - Solution 1: take derivatives and solve (there is a closed form solution!)
  - Solution 2: use gradient descent
    - This approach is much more likely to be useful for general loss functions

Gradient Descent

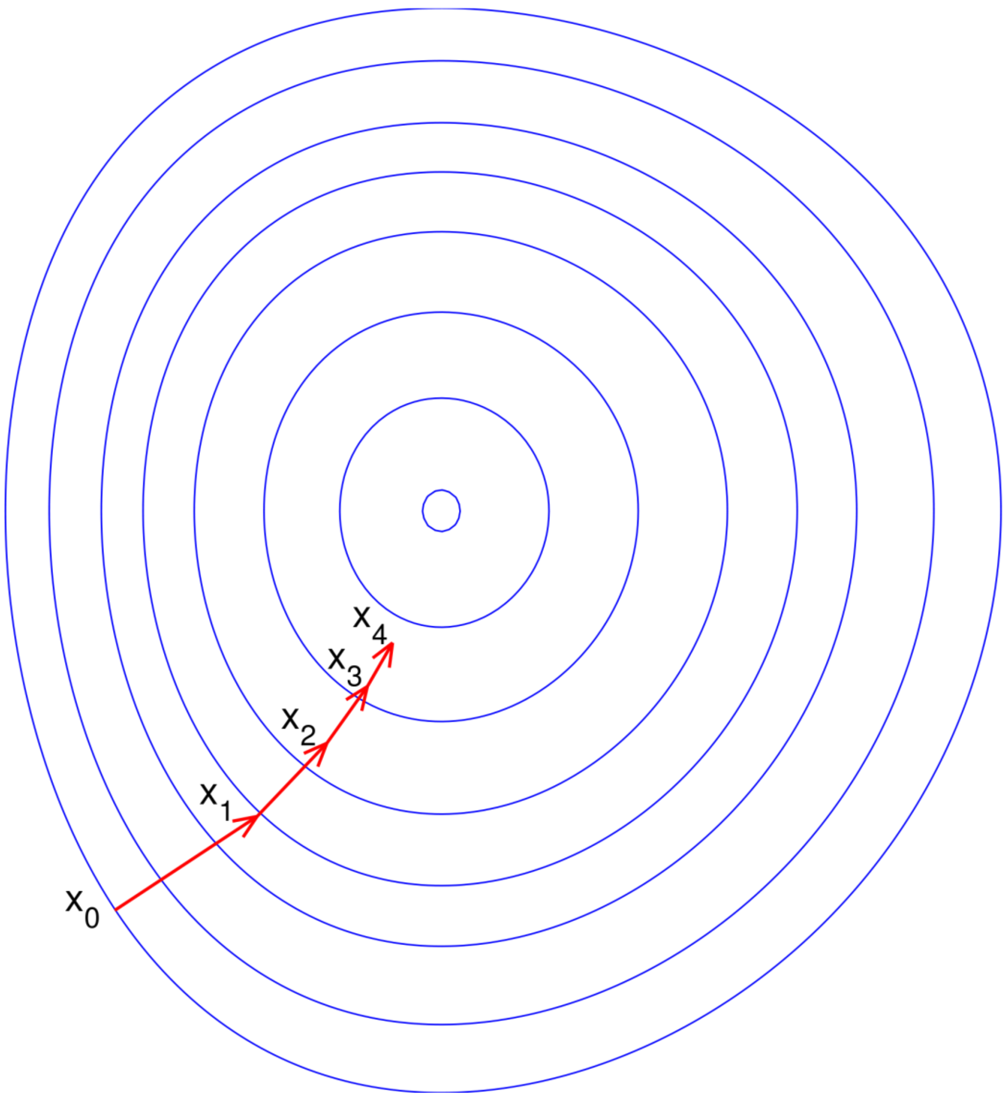
Iterative method to minimize a differentiable function f

- Pick an initial point x_0
- Iterate until convergence

$$x_{t+1} = x_t - \gamma_t \nabla f(x_t)$$

where γ_t is the t^{th} step size

Gradient Descent



source: Wikipedia

Gradient Descent

$$\min_{a,b} \frac{1}{n} \sum_i \left(ax^{(i)} + b - y^{(i)}\right)^2$$

- What is the gradient of this function?

- What does the gradient descent iteration look like for this simple regression problem?

Linear Regression

- In higher dimensions, the linear regression problem is essentially the same only $x^{(i)} \in \mathbb{R}^m$

$$\min_{a \in \mathbb{R}^m, b} \frac{1}{n} \sum_i \left(a^T x^{(i)} + b - y^{(i)}\right)^2$$

- Can still use gradient descent to minimize this
  - Not much more difficult than the $m = 1$ case

Gradient Descent

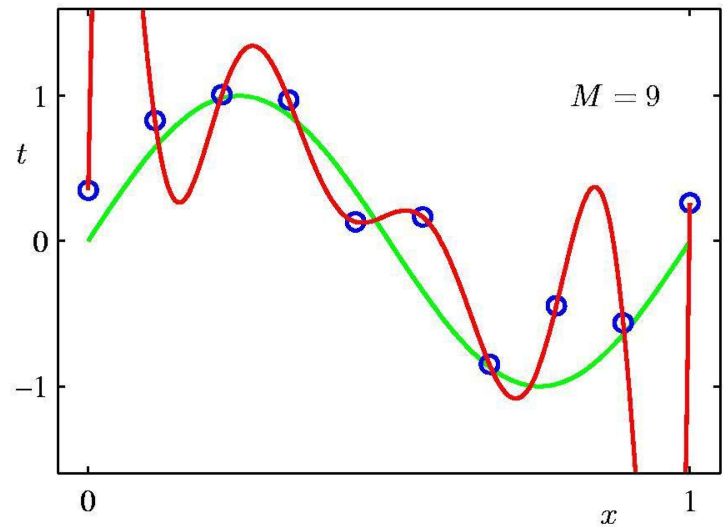
- Gradient descent converges under certain technical conditions on the function f and the step size γ_t
  - If f is convex, then any fixed point of gradient descent must correspond to a global optimum of f
  - In general, convergence is only guaranteed to a local optimum

Regression

- What if we enlarge the hypothesis class?
  - Quadratic functions
  - k degree polynomials
- Can we always learn better with a larger hypothesis class?

Regression

- What if we enlarge the hypothesis class?
  - Quadratic functions
  - k degree polynomials
- Can we always learn better with a larger hypothesis class?



Regression

- What if we enlarge the hypothesis class?
  - Quadratic functions
  - k degree polynomials
- Can we always learn better with a larger hypothesis class?
  - Larger hypothesis space always decreases the cost function, but this does NOT necessarily mean better predictive performance
  - This phenomenon is known as overfitting
    - Ideally, we would select the simplest hypothesis consistent with the observed data

Binary Classification

- Regression operates over a continuous set of outcomes
- Suppose that we want to learn a function $f: X \to \{0,1\}$
- As an example:

	$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	x_3	y
1	0	0	1	0
2	0	1	0	1
3	1	1	0	1
4	1	1	1	0

How do we pick the hypothesis space?

How do we find the best f in this space?

Binary Classification

- Regression operates over a continuous set of outcomes
- Suppose that we want to learn a function $f: X \to \{0,1\}$
- As an example:

	$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	x_3	y
1	0	0	1	0
2	0	1	0	1
3	1	1	0	1
4	1	1	1	0

How many functions with three binary inputs and one binary output are there?

Binary Classification

	x_1	x_2	x_3	y
	0	0	0	?
1	0	0	1	0
2	0	1	0	1
	0	1	1	?
	1	0	0	?
	1	0	1	?
3	1	1	0	1
4	1	1	1	0

2^8 possible functions

2^4 are consistent with the observations

How do we choose the best one?

What if the observations are noisy?

Challenges in ML

- How to choose the right hypothesis space?
  - Number of factors influence this decision: difficulty of learning over the chosen space, how expressive the space is
- How to evaluate the quality of our learned hypothesis?
  - Prefer "simpler" hypotheses (to prevent overfitting)
  - Want the outcome of learning to generalize to unseen data

Challenges in ML

- How do we find the best hypothesis?
  - This can be an NP-hard problem!
  - Need fast, scalable algorithms if they are to be applicable to real-world scenarios