CS 6820 – Machine Learning

Lecture 4 (Info only – not given in class)

Instructor: Eric S. Gayles, PhD.

Jan 16, 2018

ML Tools

- Matlab
- GNU Octave
- SciPy, Numpy, SciKits
  - Recommend installing Jupyter from www.anaconda.com (evolved from iPython)
  - Good Read–Eval–Print Loop (REPL) environment
  - https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks
- Basic Linear Algebra Subprograms (BLAS)
- LAPACK — Linear Algebra PACKage
- Shogun-toolbox
- MLlib – Apache
- Deeplearn.js
- ConvnetJS
- MLPack
- TensorFlow

NumPy and SciPy

- NumPy and SciPy are open-source add-on modules to Python that provide common mathematical and numerical routines in pre-compiled, fast functions. These are growing into highly mature packages that provide functionality that meets, or perhaps exceeds, that associated with common commercial software like MatLab.

- The NumPy (Numeric Python) package provides basic routines for manipulating large arrays and matrices of numeric data.

- The SciPy (Scientific Python) package extends the functionality of NumPy with a substantial collection of useful algorithms, like minimization, Fourier transformation, regression, and other applied mathematical techniques.

- https://docs.scipy.org/doc/scipy/reference/tutorial/

Math Essentials

- Machine learning algorithms execute numerical solutions from Optimization Theory (Calculus), Statistics, Linear Algebra, and Discrete Math.

- Linear Algebra
  - Useful for compact representation of linear transformations on data
  - Dimensionality reduction techniques
  - Solving systems of equations

Math Essentials

- To have maximum effectiveness developing Machine Learning solutions, you will need good mathematical intuitions about certain general machine learning principles and algorithms.

Math Essentials

- Choose the best algorithm for the problem
- Set parameter and initial conditions
- Estimate runtime for solutions
- Drive validation strategies and troubleshooting
- Recognize/avoid over- or underfitting
- Set confidence boundaries on solutions
- Choose the best coding approach, framework, language, etc.

Math Essentials

- $a \in A$ — *set membership*: a is *member* of set A
- $| B |$ — *cardinality*: number of items in set B
- $|| \mathbf{v} ||$ — *norm*: length of vector v
- $\sum$ — *summation*
- $\int$ — *integral*
- $\Re$ — the set of *real* numbers
- $\Re^n$ — *real number space* of dimension n

  $n = 2$: plane or 2-space
  $n = 3$: 3- (dimensional) space
  $n > 3$: n-space or *hyperspace*

Math Essentials

- $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, $\mathbf{u}$, $\mathbf{v}$ — *vector* (bold, lower case)
- $\mathbf{A}$, $\mathbf{B}$, $\mathbf{X}$ — *matrix* (bold, upper case)
- $y = f(x)$ — *function* (*map*): assigns unique value in range of y to each value in domain of x
- dy / dx — *derivative* of y with respect to single variable x
- $y = f(\mathbf{x})$ — *function* on multiple variables, i.e. a vector of variables; *function* in *n*-space
- $\partial y / \partial x_i$ — *partial derivative* of y with respect to element i of vector $\mathbf{x}$

Math Essentials

A *probability space* is a *random process* or *experiment* with three components:

- Ω, the set of possible *outcomes* O
  - number of possible outcomes = $|\ \Omega\ | = N$
- F, the set of possible *events* E
  - an event comprises 0 to N outcomes
  - number of possible events = $|\ F\ | = 2^N$
- P, the *probability distribution*
  - function mapping each outcome and event to real number between 0 and 1 (the *probability* of O or E)
  - probability of an event is *sum* of probabilities of possible outcomes in event

Math Essentials

Given:

- A discrete random variable X, with possible values $x = x_1, x_2, \ldots x_n$
- Probabilities $p(X = x_i)$ that X takes on the various values of x_i
- A function $y_i = f(x_i)$ defined on X

The *expected value* of f is the probability-weighted "average" value of $f(x_i)$:

$$E(f) = \sum_i p(x_i) \cdot f(x_i)$$

Math Essentials

- Addition of two matrices
  - matrices must be same size
  - add corresponding elements:
    $$c_{ij} = a_{ij} + b_{ij}$$
  - result is a matrix of same size

$$\mathbf{C} = \mathbf{A} + \mathbf{B} = \begin{pmatrix} a_{11} + b_{11} & \cdots & a_{1n} + b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & \cdots & a_{mn} + b_{mn} \end{pmatrix}$$

- Scalar multiplication of a matrix
  - multiply each element by scalar:
    $$b_{ij} = d \cdot a_{ij}$$
  - result is a matrix of same size

$$\mathbf{B} = d \cdot \mathbf{A} = \begin{pmatrix} d \cdot a_{11} & \cdots & d \cdot a_{1n} \\ \vdots & \ddots & \vdots \\ d \cdot a_{m1} & \cdots & d \cdot a_{mn} \end{pmatrix}$$

Math Essentials

The probability a discrete variable A takes value a is: $0 \le P(A=a) \le 1$ — Probability

Probabilities of alternative outcomes add: $P(A \in \{a, a'\}) = P(A=a) + P(A=a')$ — Alternatives

The probabilities of all outcomes must sum to one: $\sum_{\text{all possible } a} P(A=a) = 1$ — Normalization

$P(A=a, B=b)$ is the joint probability that both $A=a$ and $B=b$ occur. — Joint Probability

Variables can be "summed out" of joint distributions: — Marginalization

$$P(A=a) = \sum_{\text{all possible } b} P(A=a, B=b)$$

$P(A=a|B=b)$ is the probability $A=a$ occurs given the knowledge $B=b$. — Conditional Probability

$P(A=a, B=b) = P(A=a)\,P(B=b|A=a) = P(B=b)\,P(A=a|B=b)$ — Product Rule

Bayes rule can be derived from the above: — Bayes Rule

$$P(A=a|B=b, \mathcal{H}) = \frac{P(B=b|A=a, \mathcal{H})\,P(A=a|\mathcal{H})}{P(B=b|\mathcal{H})} \propto P(A=a, B=b|\mathcal{H})$$

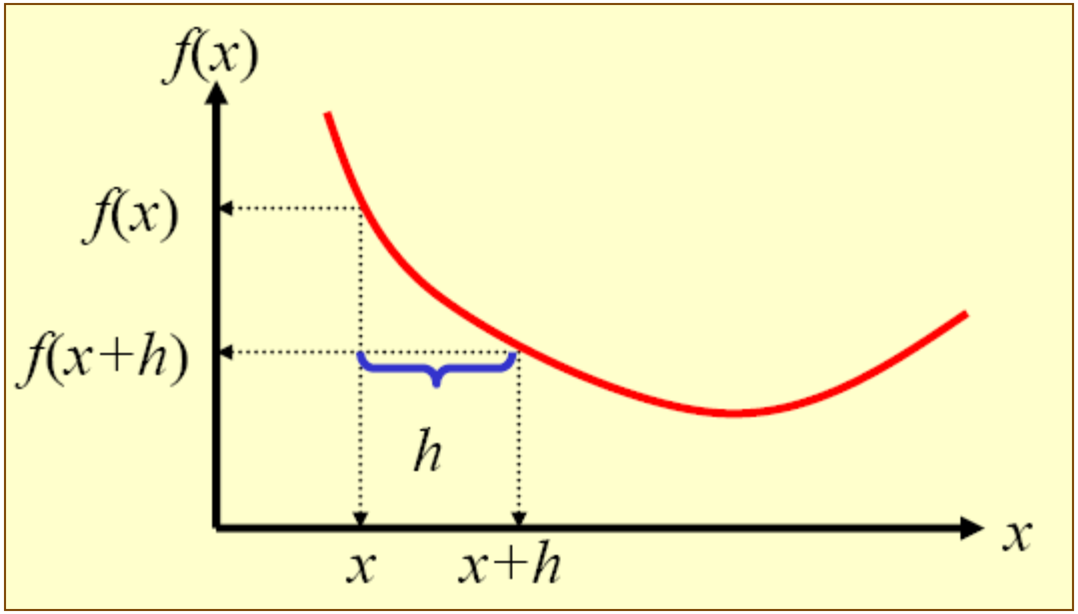
* Thanks to A Yu

Math Essentials

The derivative of *f*: *R* → *R* is a function *f* ′: *R* → *R* st.

$$f'(x) = \frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

if the limit exists.



Math Essentials

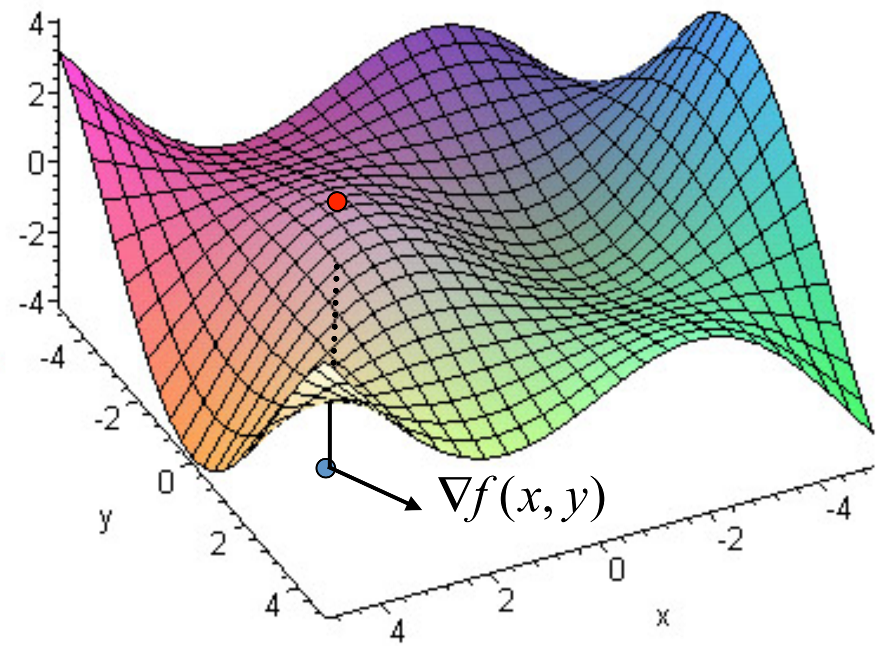
- The gradient is an important concept from calculus in the context of machine learning.
- Gradients generalize derivatives to scalar functions of several variables.

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \quad \text{i.e.} \quad [\nabla f]_i = \frac{\partial f}{\partial x_i}$$

- <u>Definition</u>: The gradient of f: in $R^2 \rightarrow R$:

It is a function ∇f: $R^2 \rightarrow R^2$ given by

$$\nabla f(x, y) := \left(\frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial y} \right)^T$$



In the plane

- Definition: The gradient of f: $R^n \to R$ is a function ∇f: $R^n \to R^n$ given by

$$\nabla f(x_1,\ldots,x_n) := \left(\frac{\partial f}{\partial x_1},\ldots, \quad \frac{\partial f}{\partial x_n} \right)^T$$

Math Essentials

The **Hessian** matrix of $f : \mathbb{R}^d \to \mathbb{R}$ is a matrix of second-order partial derivatives:

$$\nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_d^2} \end{bmatrix} \quad \text{i.e.} \quad [\nabla^2 f]_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

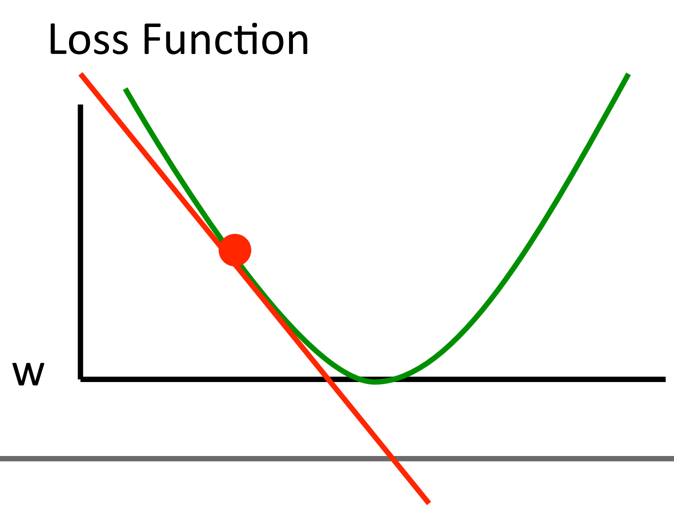
Basic Intro: Gradient Descent

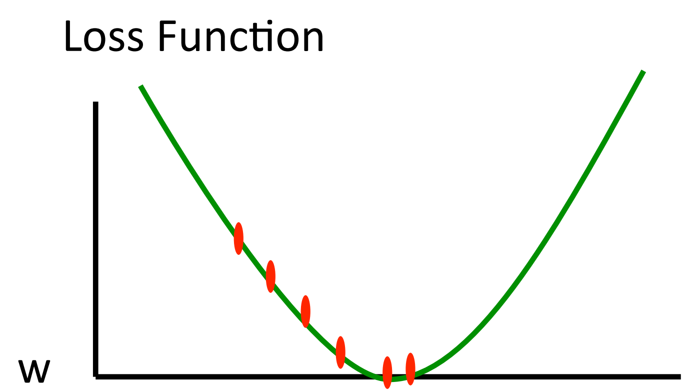
Partial derivatives provide the slope to direct the solution toward minimization of the loss function.

- Pick initial solution
- Repeat until optimized {
  - Evaluate slope and minima conditionality
  - Pick dimension(s)
  - Move a small amount in the direction decreasing loss

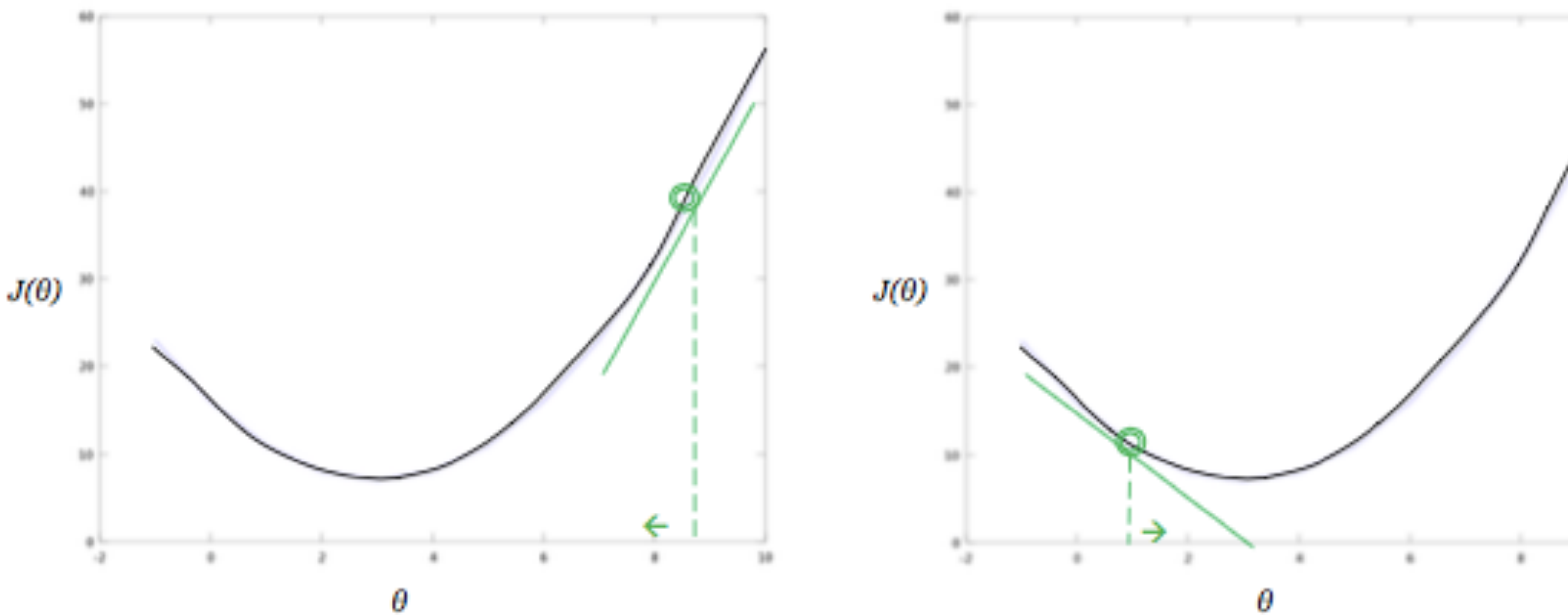
  }

$$\theta_1 := \theta_1 - \alpha \frac{\partial J}{\partial \theta_1}$$
$$\theta_2 := \theta_2 - \alpha \frac{\partial J}{\partial \theta_2}$$
$$\vdots$$
$$\theta_k := \theta_k - \alpha \frac{\partial J}{\partial \theta_k}$$



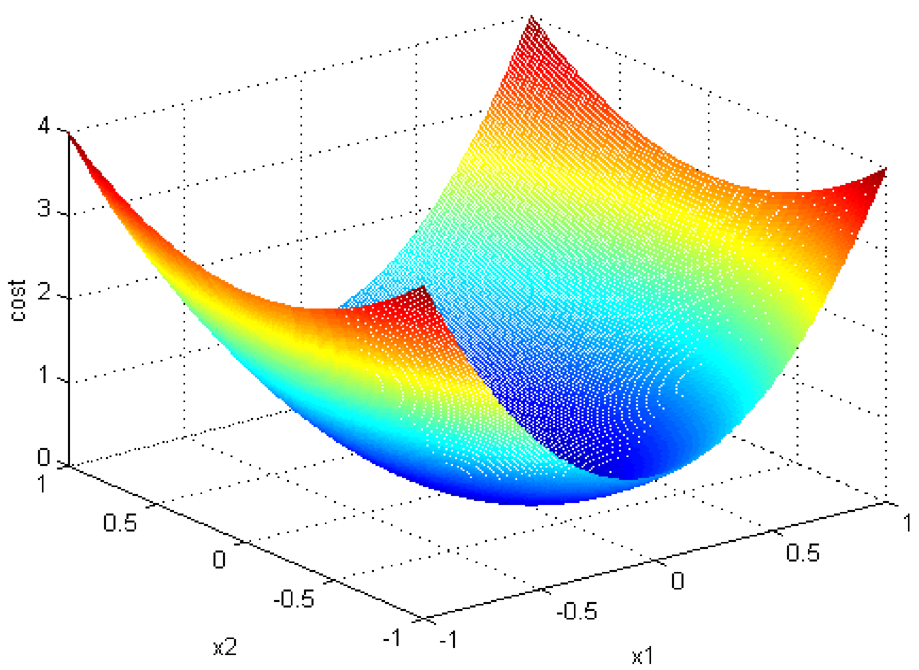


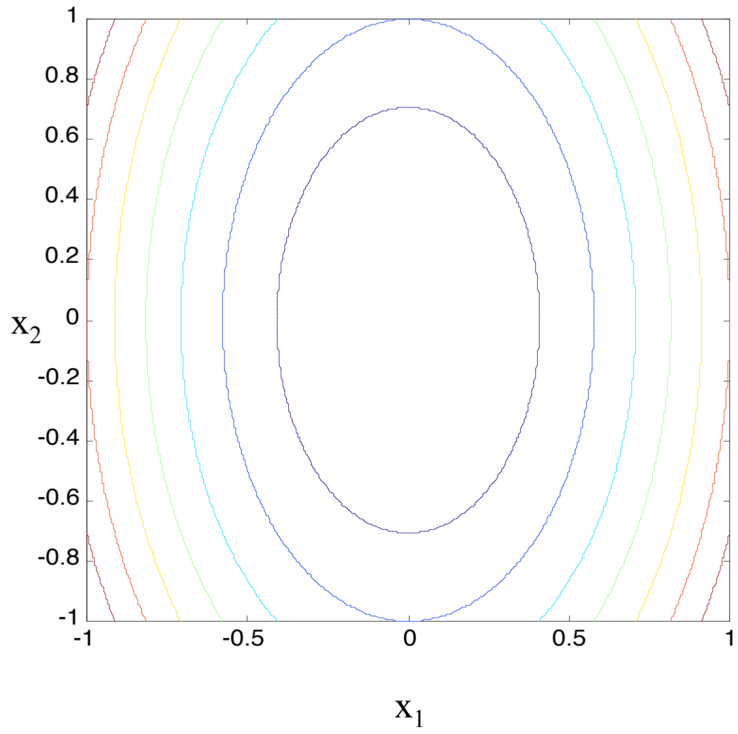
Basic Intro: Gradient Descent



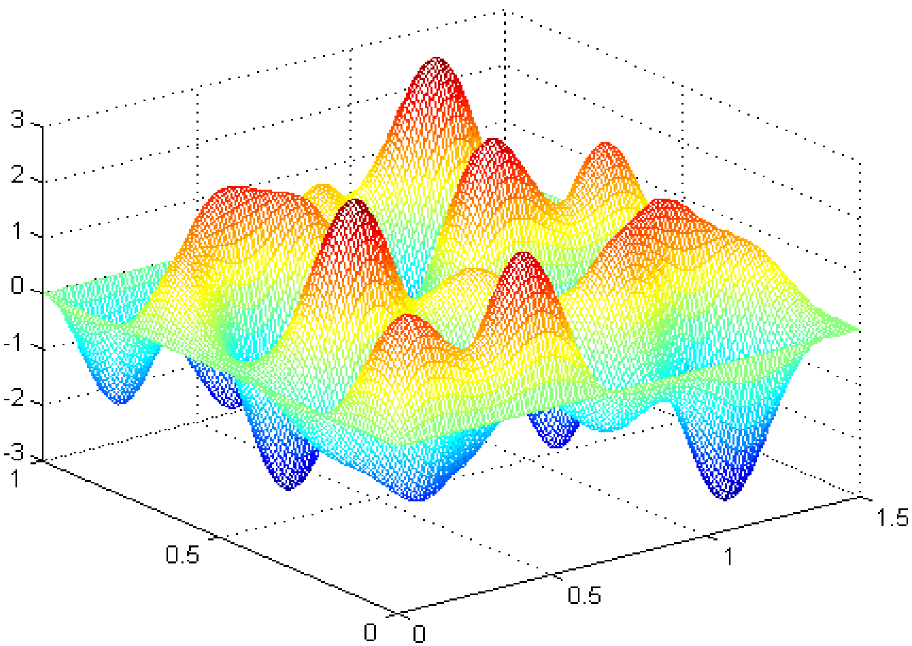
* A Beginners Tutorial for Machine Learning Beginners, Hao

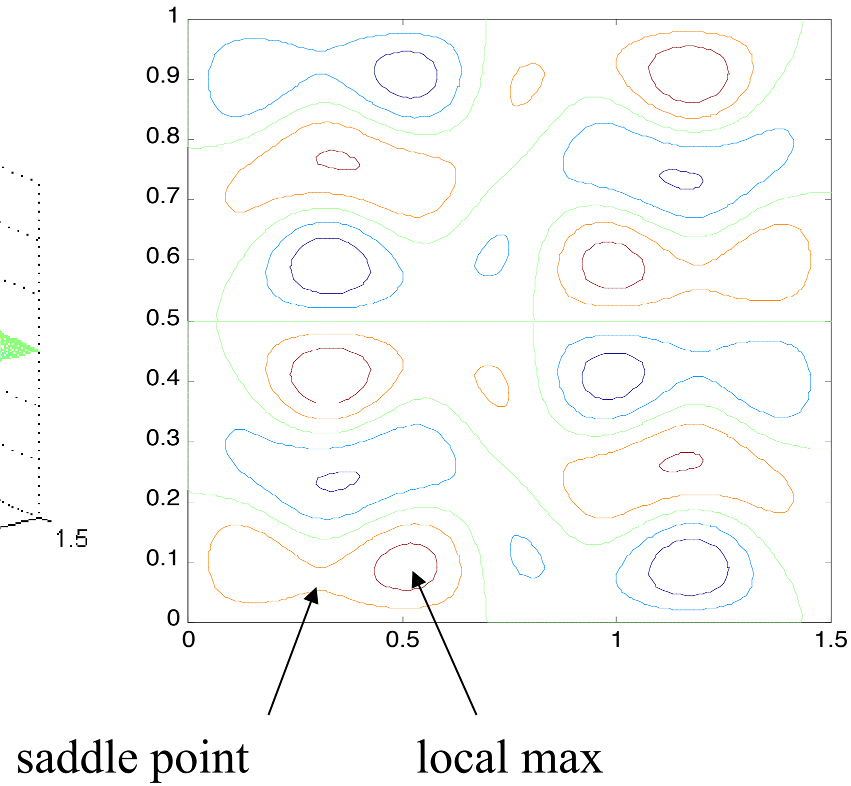
Basic Intro: Gradient Descent





Basic Intro: Gradient Descent





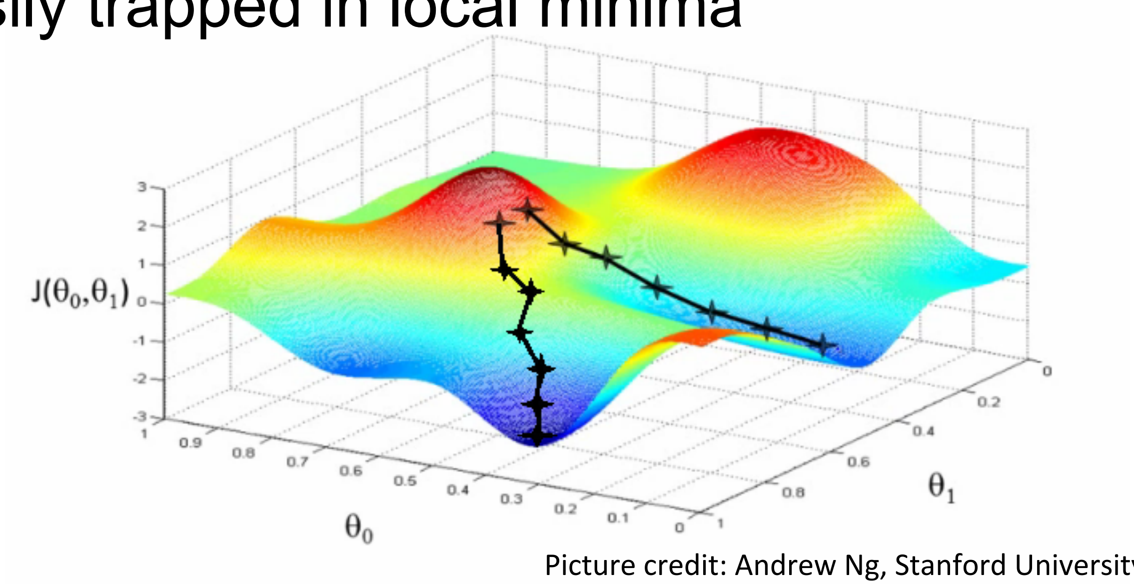
Basic Intro: Gradient Descent

- Simple concept: follow the gradient *downhill*
- Process:
  1. Pick a starting position: $\mathbf{x}^0 = (x_1, x_2, \ldots, x_d)$
  2. Determine the descent direction: $-\nabla f(\mathbf{x}^t)$
  3. Choose a learning rate: η
  4. Update your position: $\mathbf{x}^{t+1} = \mathbf{x}^t - \eta \cdot \nabla f(\mathbf{x}^t)$
  5. Repeat from 2) until stopping criterion is satisfied
- Typical stopping criteria
  - $\nabla f(\mathbf{x}^{t+1}) \sim 0$
  - some validation metric is optimized

* Howbert

Gradient descent optimization

- Problems:
  - Choosing step size
    - too small → convergence is slow and inefficient
    - too large → may not converge
  - Can get stuck on "flat" areas of function
  - Easily trapped in local minima



Picture credit: Andrew Ng, Stanford University, Coursera Machine Learning