# Applied Optimization: Gradients

Alexander G. Ororbia II

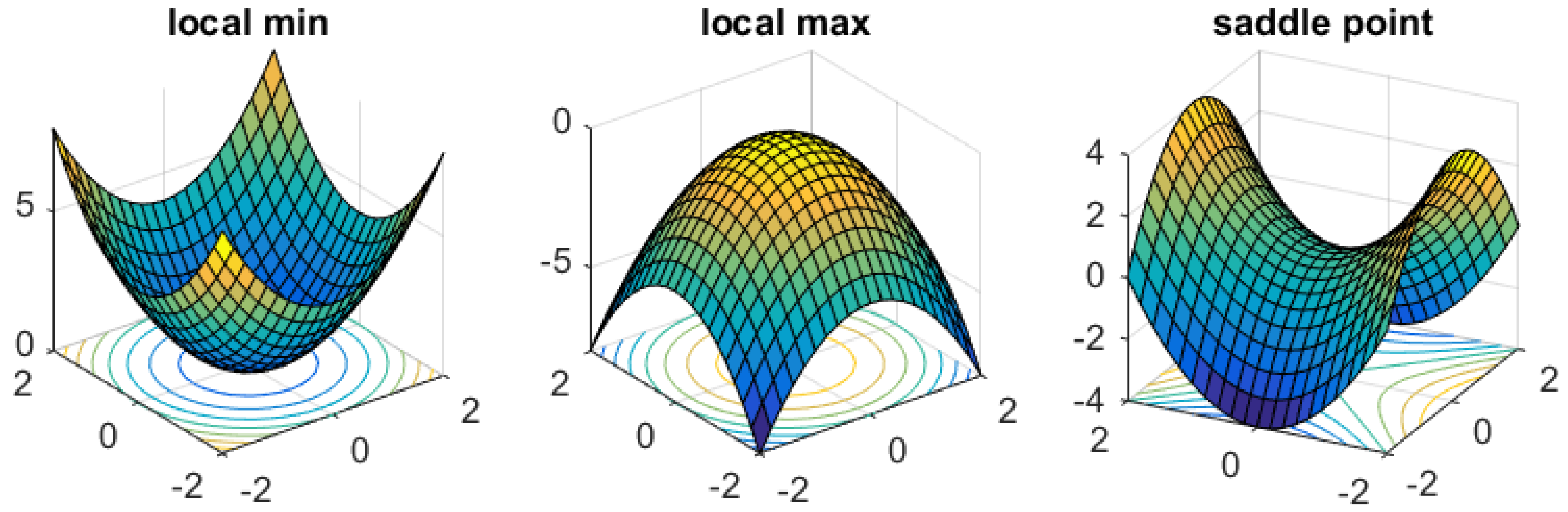
Introduction to Machine Learning

CSCI-335

1/26/2026 (*Mon*) & 1/30/2026 (*Fri*)

# Gradients Means Derivatives; This Means Some Calculus…

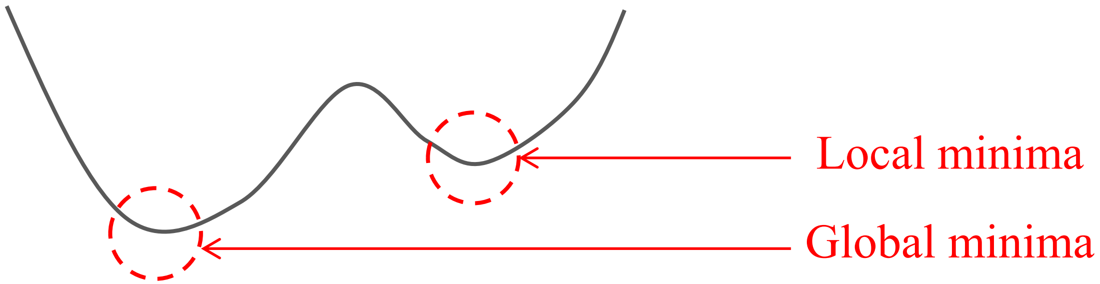
- [See Crash Course on Derivatives]



# The Value of Derivatives

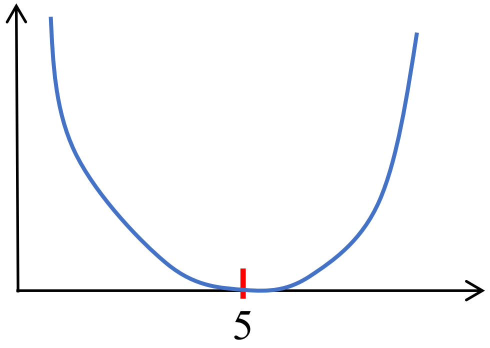
- How do you solve the following problems?

(1)




(2) $\boxed{\min_{w} f(w) = (w-5)^2}$

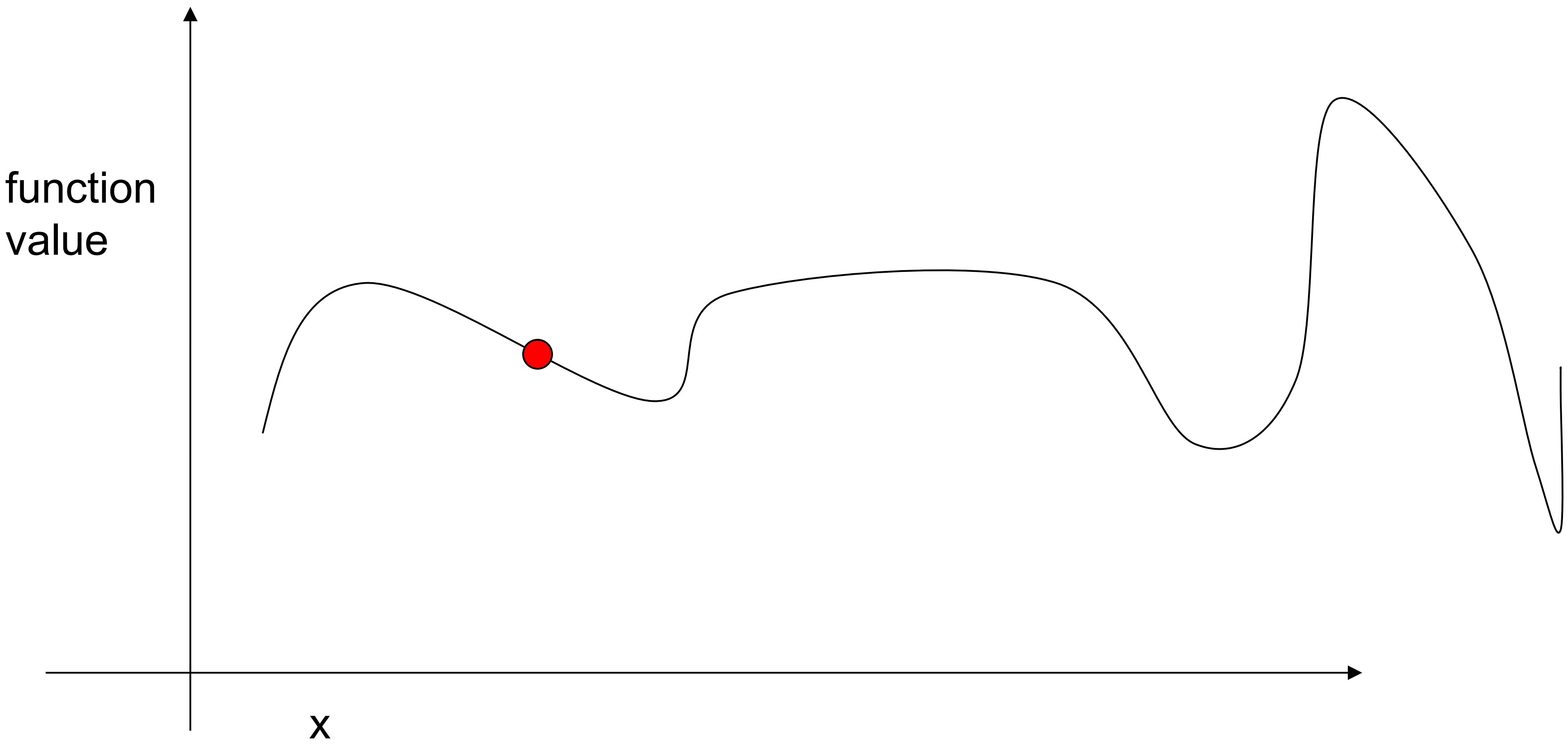
(a) Plot




(b) Take ***derivatives***, **check = 0**

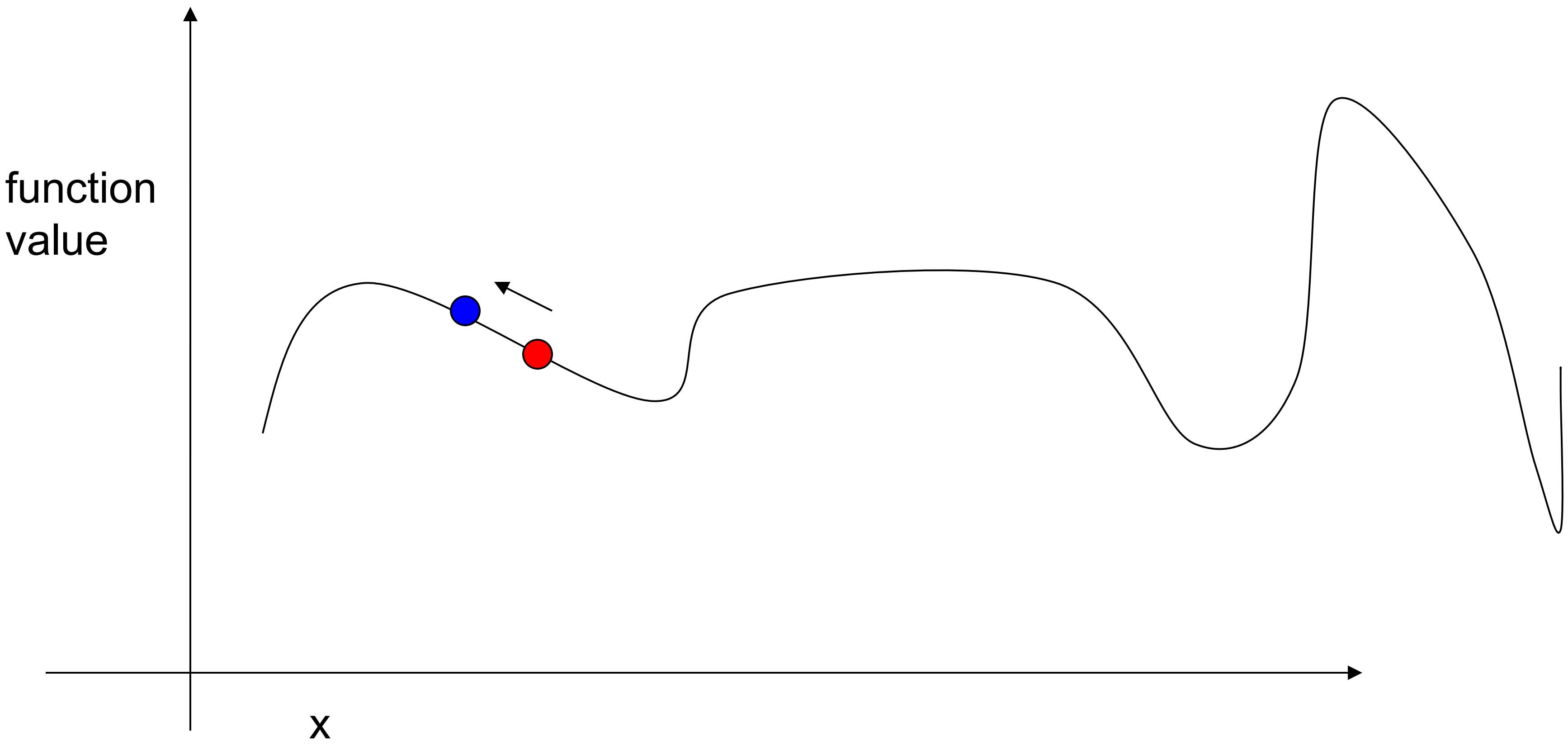
# Gradient **Ascent**

- Random Starting Point



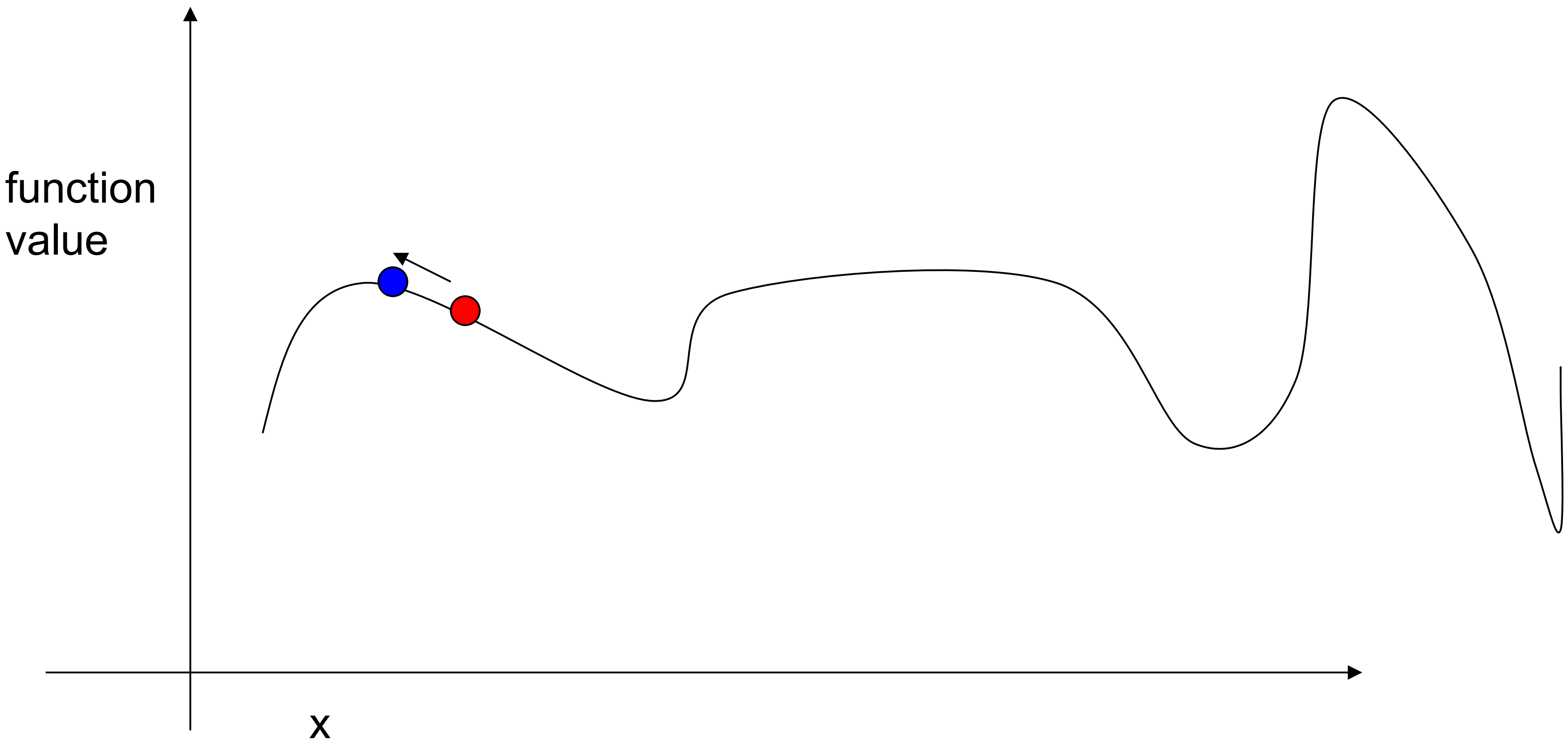
# Gradient Ascent

- Take step in direction of largest increase (obvious in 1D, must be computed in higher dimensions)



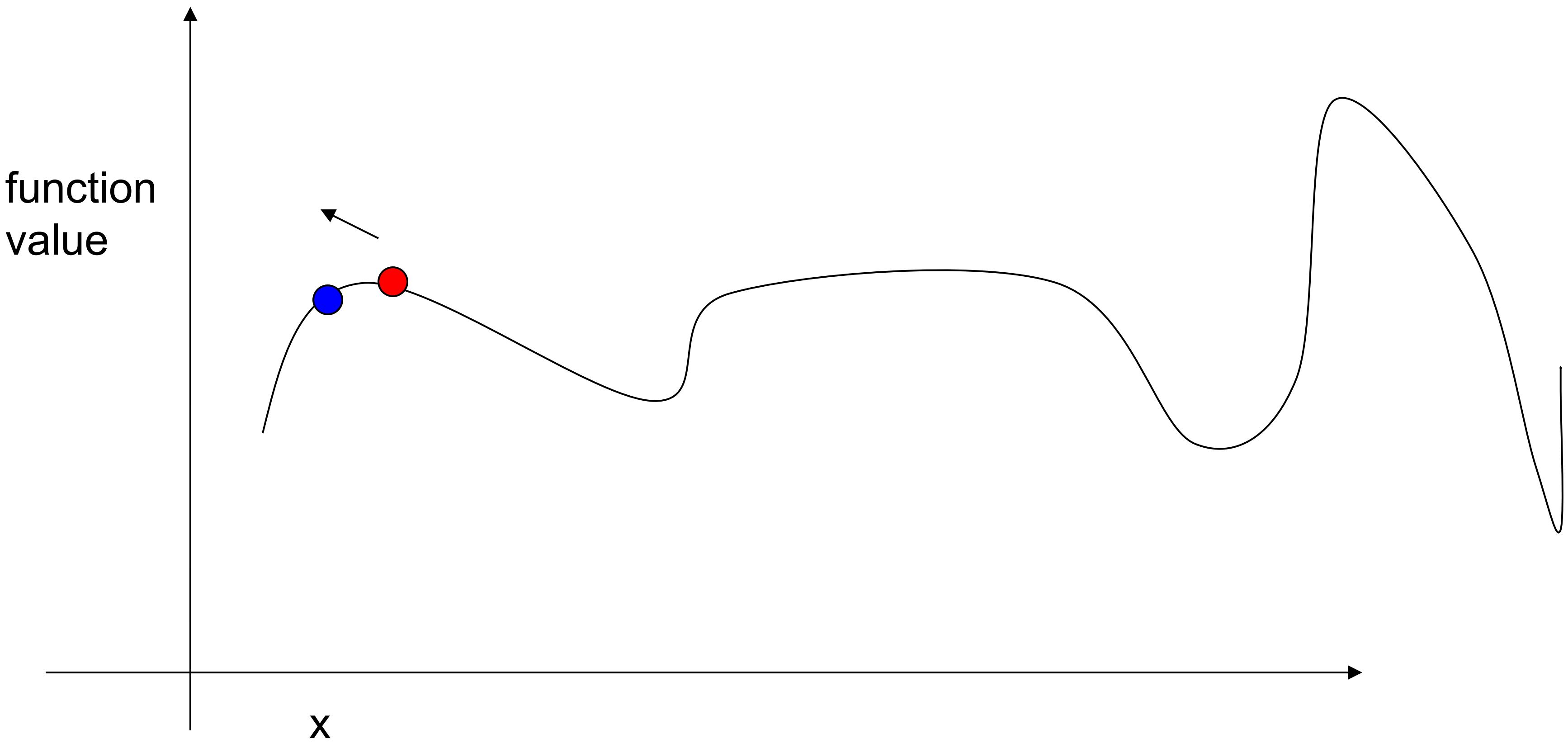
# Gradient Ascent

- Repeat



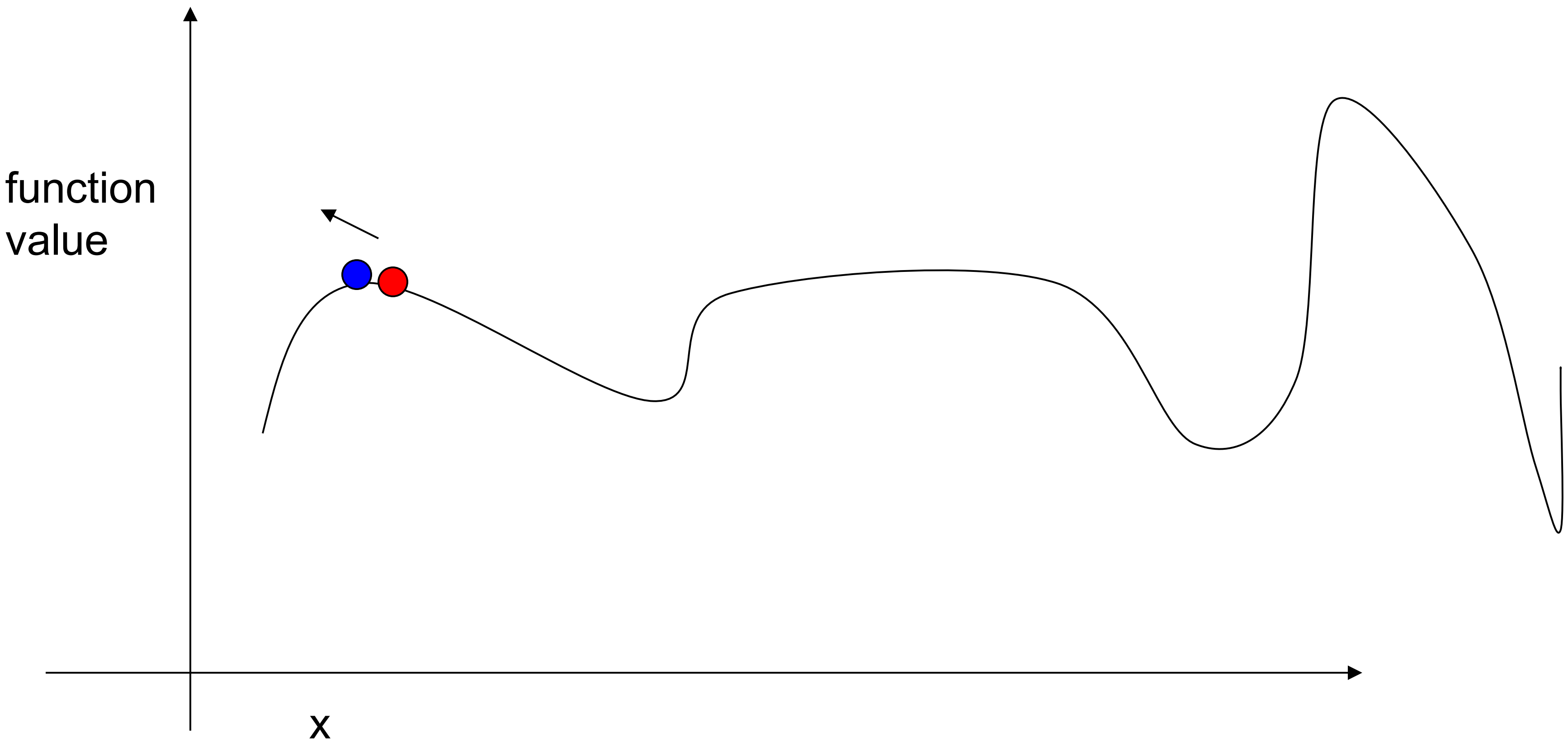
# Gradient Ascent

- Next step is actually lower, so stop



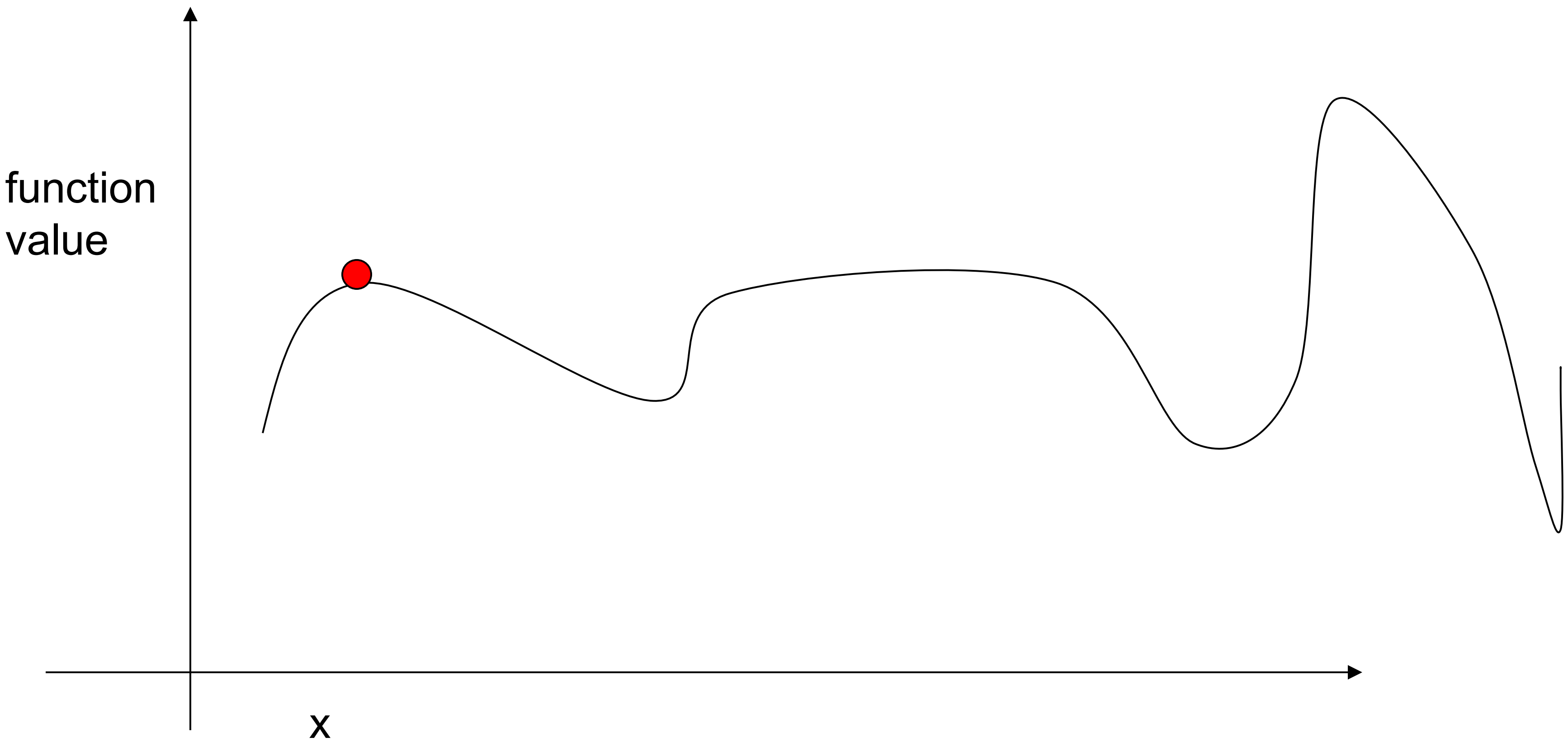
# Gradient Ascent

- Could reduce step size to “hone in”



# Gradient Ascent

- Converge to (local) maximum



# The Finite Difference Method

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

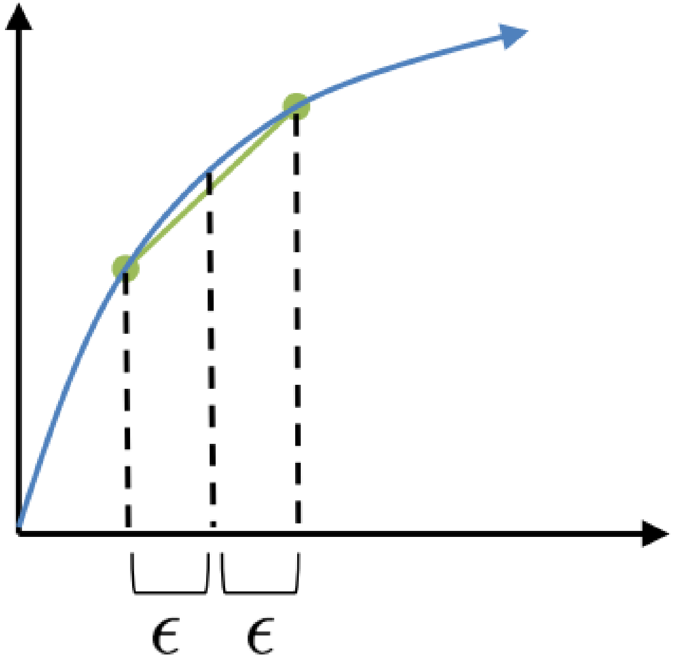
The *centered* finite difference approximation is:

$$\frac{\partial}{\partial \theta_i} J(\boldsymbol{\theta}) \approx \frac{(J(\boldsymbol{\theta} + \epsilon \cdot \boldsymbol{d}_i) - J(\boldsymbol{\theta} - \epsilon \cdot \boldsymbol{d}_i))}{2\epsilon}$$

where $\boldsymbol{d}_i$ is a 1-hot vector consisting of all zeros except for the $i$th entry of $\boldsymbol{d}_i$, which has value 1.

**Notes:**

- Suffers from issues of floating point precision, in practice
- Typically only appropriate to use on small examples with an appropriately chosen epsilon




73

# Ex: Gradient Descent on Least Squares

- Criterion to minimize $f(\mathrm{x}) = \frac{1}{2} \| A\mathrm{x} - \mathrm{b} \|^2$ Evaluation
  - Least squares regression $E_D(\mathrm{w}) = \frac{1}{2} \sum_{n=1}^{N} \left\{ t_n - \mathrm{w}^T \phi(x_n) \right\}^2$
- The gradient is

$$\nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}\right) = A^T\left(A\boldsymbol{x} - b\right) = A^T A\boldsymbol{x} - A^T\boldsymbol{b}$$

Optimization

- Gradient Descent algorithm is

1. Set step size $\varepsilon$, tolerance $\delta$ to small, positive nos.
2. *while* $\| A^T A\boldsymbol{x} - A^T\boldsymbol{b} \|_2 > \delta$ do

   $$\boldsymbol{x} \leftarrow \boldsymbol{x} - \varepsilon\left(A^T A\boldsymbol{x} - A^T\boldsymbol{b}\right)$$
3. *end while*

# Calculus in Optimization

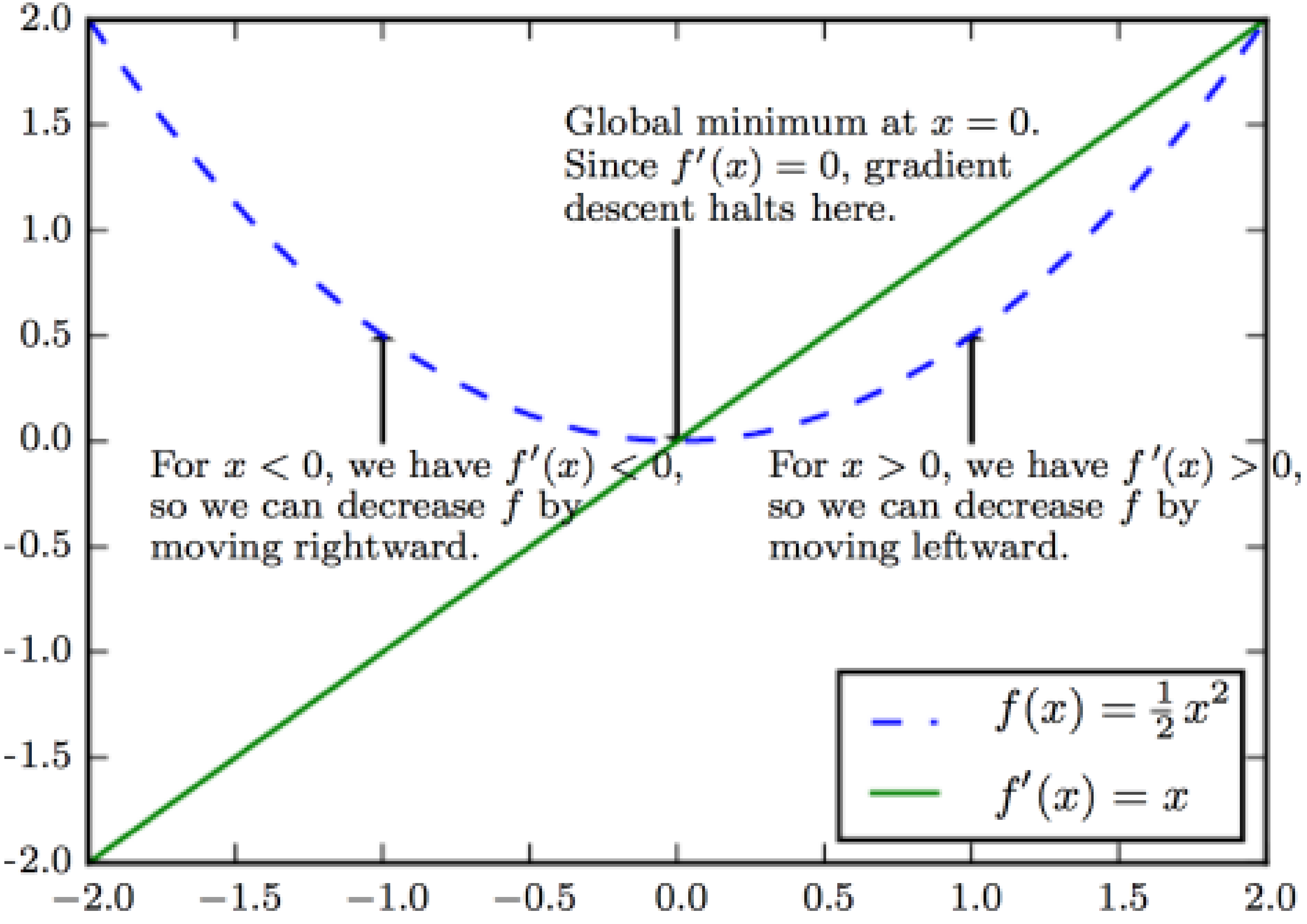
- Suppose we have function $y=f(x)$, $x$, $y$ real nos.
  - Derivative of function denoted: $f'(x)$ or as $dy/dx$
    - Derivative $f'(x)$ gives the slope of $f(x)$ at point $x$
    - It specifies how to scale a small change in input to obtain a corresponding change in the output:

$$f(x+\varepsilon) \approx f(x) + \varepsilon f'(x)$$

  - It tells how you make a small change in input to make a small improvement in $y$
  - We know that $f(x - \varepsilon\ \text{sign}\ (f'(x)))$ is less than $f(x)$ for small $\varepsilon$. Thus we can reduce $f(x)$ by moving $x$ in small steps with opposite sign of derivative
    - This technique is called *gradient descent* (Cauchy 1847)

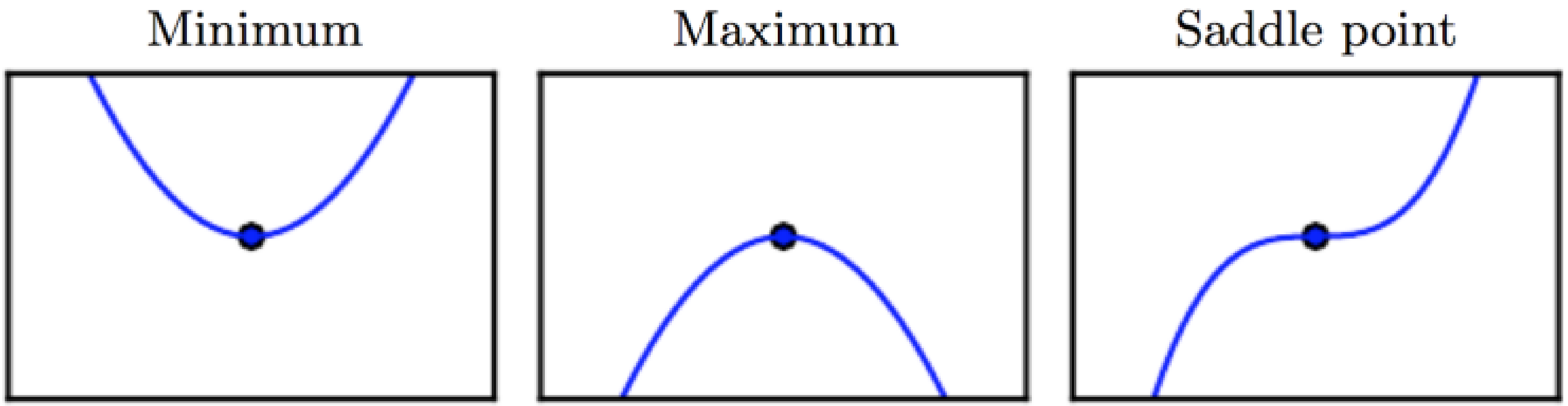
signum(v)

# Gradient Descent Illustrated



# Stationary points, Local Optima

- When $f'(x)=0$ derivative provides no information about direction of move
- Points where $f'(x)=0$ are known as *stationary* or critical points
  - Local minimum/maximum: a point where $f(x)$ lower/ higher than all its neighbors
  - Saddle Points: neither maxima nor minima



# Functions with Multiple Inputs

- Need partial derivatives
- $\frac{\partial}{\partial x_i} f(\boldsymbol{x})$ measures how $f$ changes as only variable $x_i$ increases at point $\boldsymbol{x}$
- Gradient generalizes notion of derivative where derivative is wrt a vector
- Gradient is vector containing all of the partial derivatives denoted $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$
  - Element $i$ of the gradient is the partial derivative of $f$ wrt $x_i$
  - Critical points are where every element of the gradient is equal to zero

w.r.t. (with respect to)

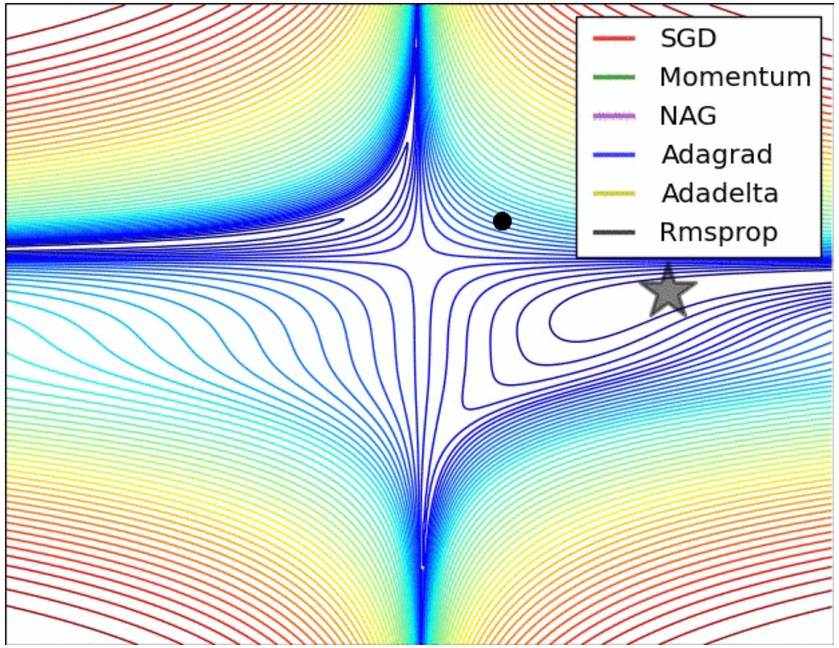
# Method of Gradient Descent

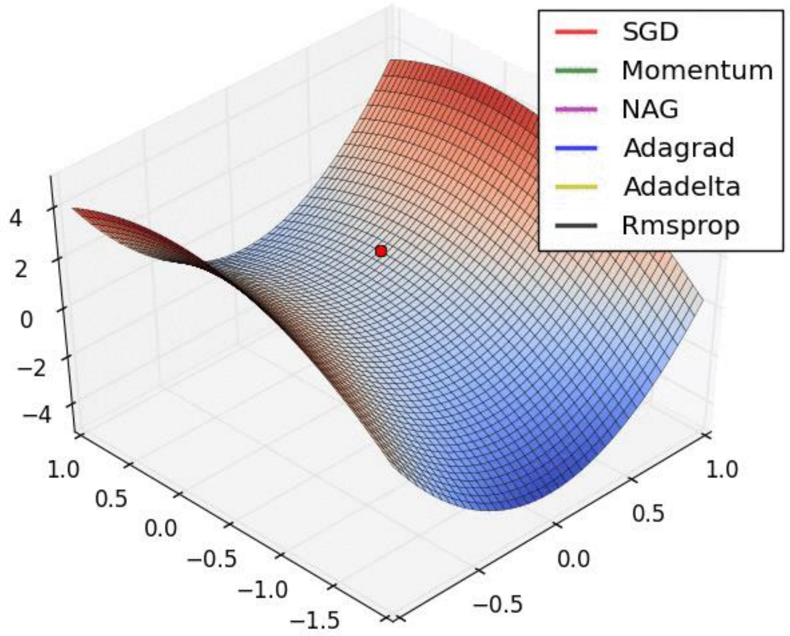
- The gradient points directly uphill, and the negative gradient points directly downhill
- Thus we can decrease $f$ by moving in the direction of the negative gradient
  - This is known as the method of steepest descent or gradient descent
- Steepest descent proposes a new point

$$\boldsymbol{x}' = \boldsymbol{x} - \varepsilon \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}\right)$$

  - where $\varepsilon$ is the learning rate, a positive scalar. Set to a small constant.

# Race of the Optimizers!



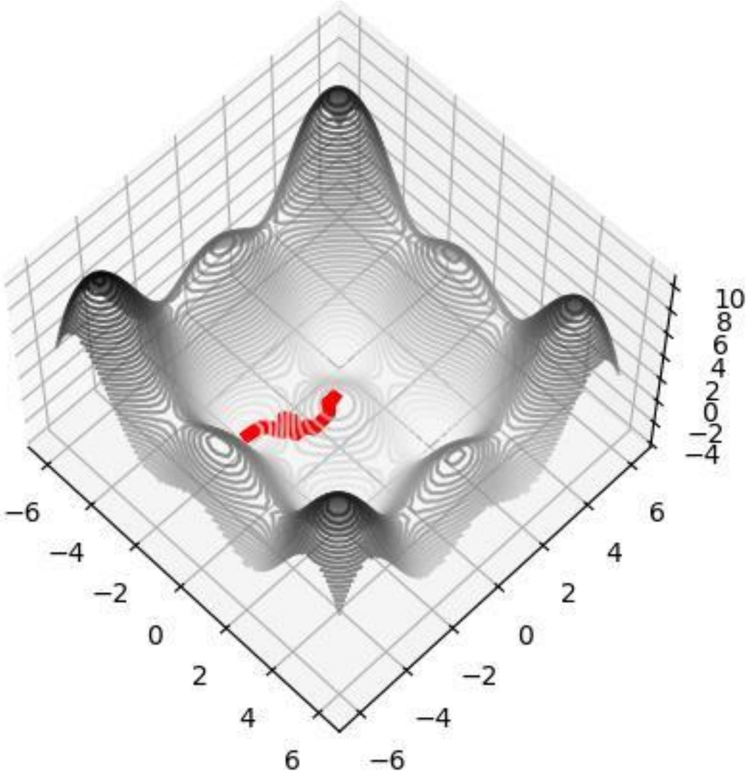



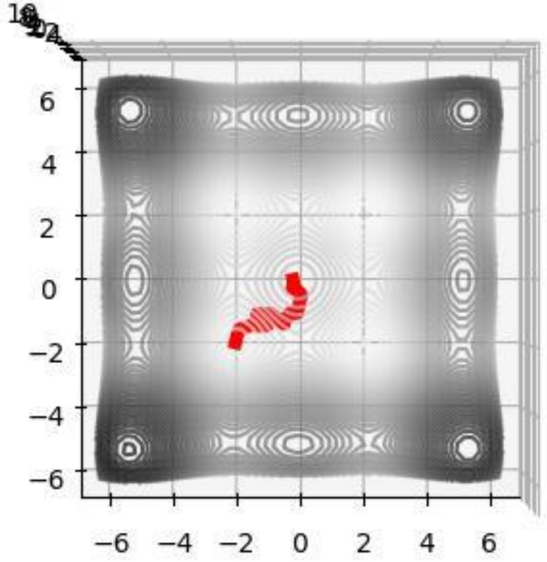

http://cs231n.github.io/neural-networks-3/#hyper

# Stochastic Hill-Climbing

***Negative Alpine Function***

$$f(x) = -\sum_i \left(x_i \sin(x_i) + 0.1\, x_i\right)$$

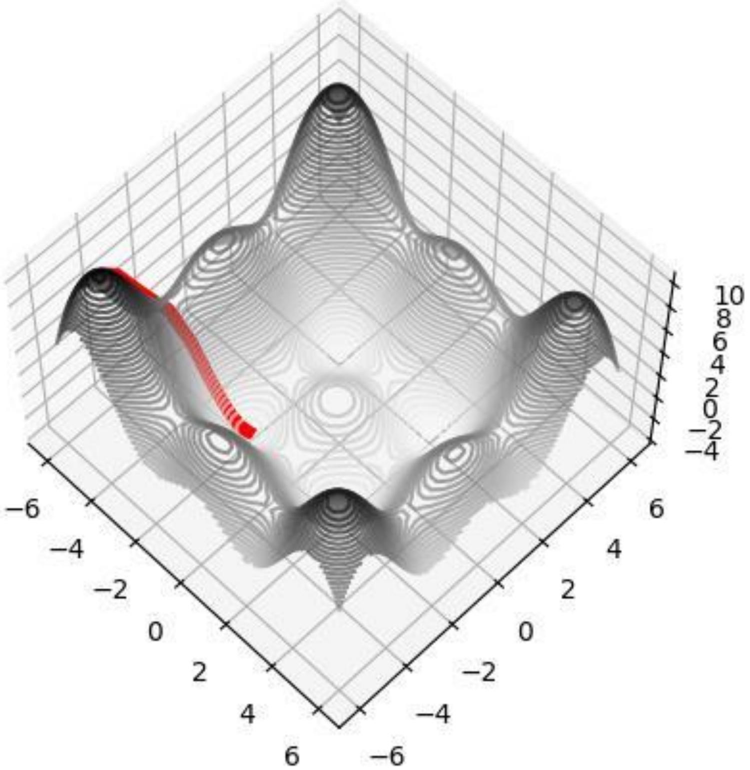


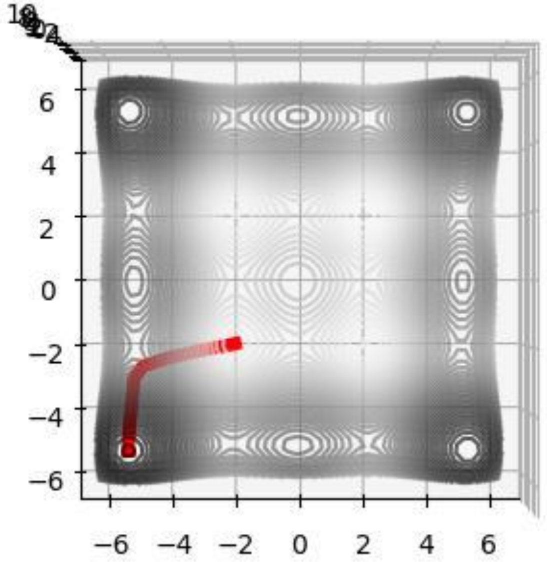



# Stochastic Gradient Ascent

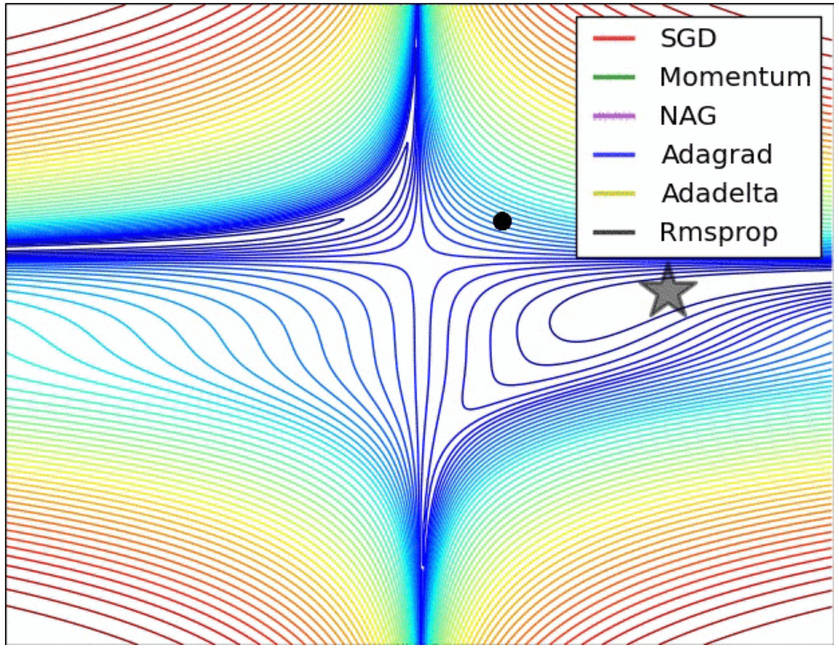
***Negative Alpine Function***

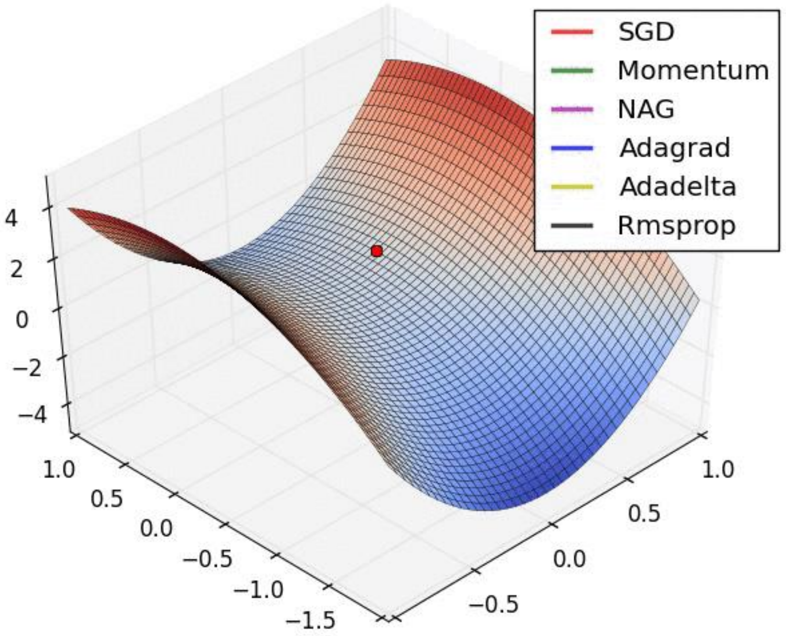
$$f(x) = -\sum_i \left(x_i \sin(x_i) + 0.1\, x_i\right)$$



# Race of the Optimizers!

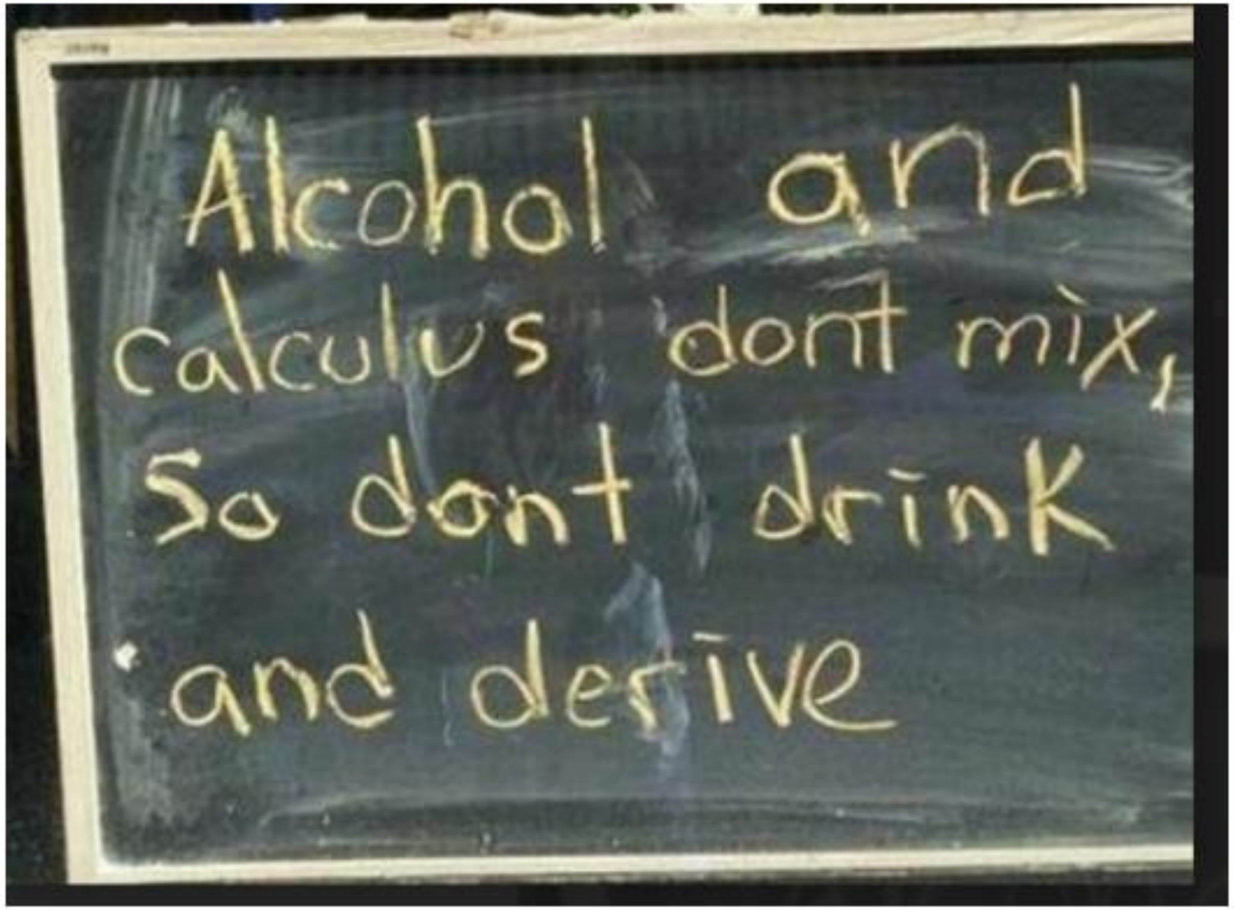



http://cs231n.github.io/neural-networks-3/#hyper

# Gradient

- Essential role of calculus



You want to build a simple univariate regression model for predicting profits $y$ for a food truck. Furthermore, you decide to restrict yourself to a linear hypothesis space and construct a model that adheres to the following form:

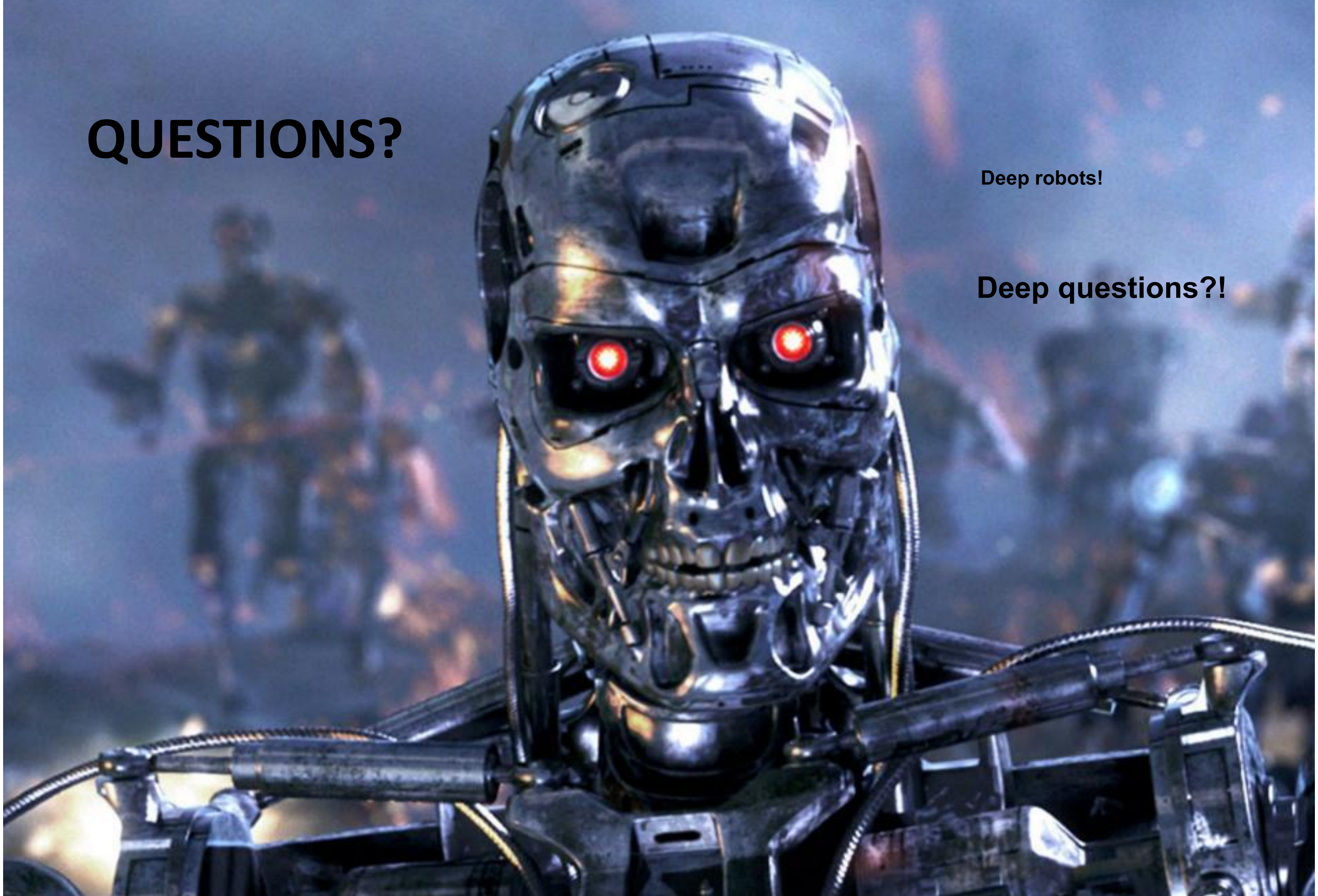
$$f_\Theta(x) = \theta_0 + \theta_1 x$$

Given the data you have collected, represented as a set of $m$ complete $(y, x)$ pairs, your goals will be to estimate the parameters of this model $\Theta = \{\theta_0, \theta_1\}$ (where $\theta_{j>0}$ is the vector of learnable coefficients that weight the observed variables, and $\theta_0$ is a single bias coefficient) using the method of steepest gradient descent. The cost function to minimize is the well-known mean squared error (MSE) defined as follows:

$$\mathcal{J}(\Theta) = \frac{1}{2m} \sum_{i=1}^{m} (f_\Theta(x^i) - y^i)^2$$

**What is a useful constrained version of this cost to get "smaller" coefficients?**
**(*Hint*: Tikhonov regularization)**

**Linear Regression & Gradient Descent**


QUESTIONS?
Deep robots!
Deep questions?!