Gradient Descent

Learning objectives
Know standard, coordinate, stochastic gradient, and minibatch gradient descent

Adagrad: core idea

Lyle Ungar
University of Pennsylvania

In part from slides written jointly with Zack Ives

Gradient Descent

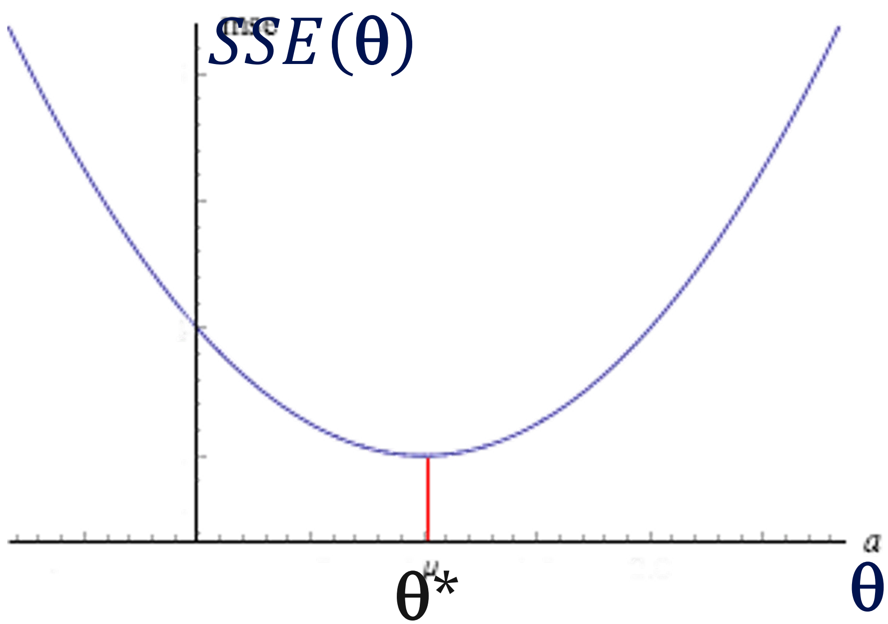
- We almost always want to minimize some loss function
- Example: Sum of Squared Error (SSE):

$$SSE(\theta) = \sum_{i=1}^{n} r_i(\theta)^2$$

$$r_i(\theta) = h_\theta\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}$$

Mean Squared Error

$$SSE(\theta) = \sum_{i=1} r_i(\theta)^2$$



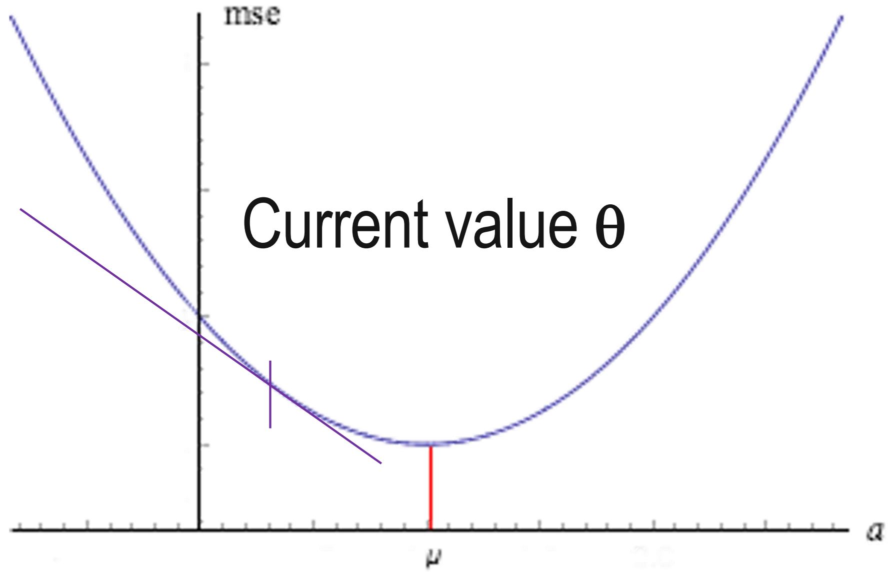
http://www.math.uah.edu/stat/expect/Variance.html

In one dimension, looks like a parabola centered around the optimal value θ*

(Generalizes to *d* dimensions)

Getting Closer

$$SSE(\theta) = \sum_{i=1}^{n} r_i(\theta)^2$$



http://www.math.uah.edu/stat/expect/Variance.html

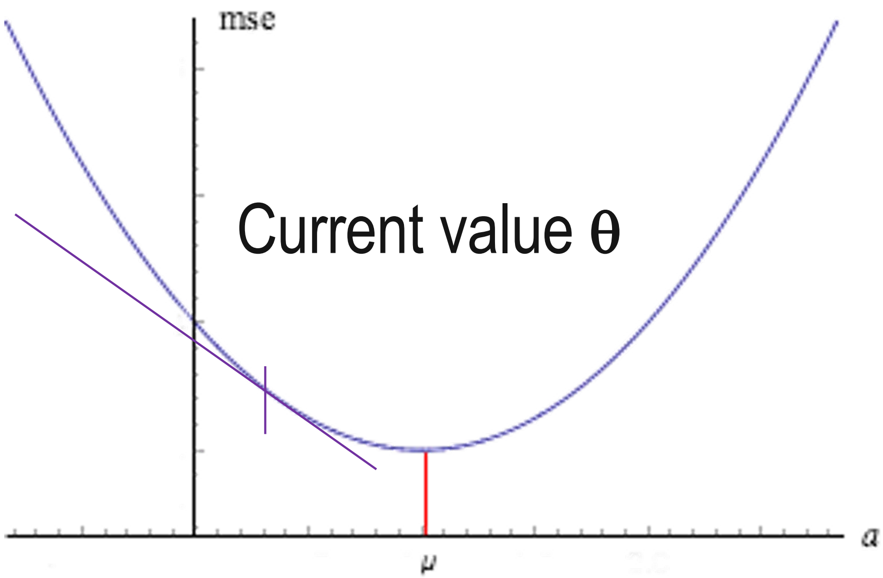
What if we use the slope of the tangent to decide where to "go next"?

$$\theta := \theta - \eta \nabla SSE(\theta)$$

the gradient

$$\nabla SSE(\theta) = \lim_{d \to 0} \frac{\left(\left(h_\theta(\theta + d) - h_\theta(\theta)\right)\right)}{d}$$

Getting Closer



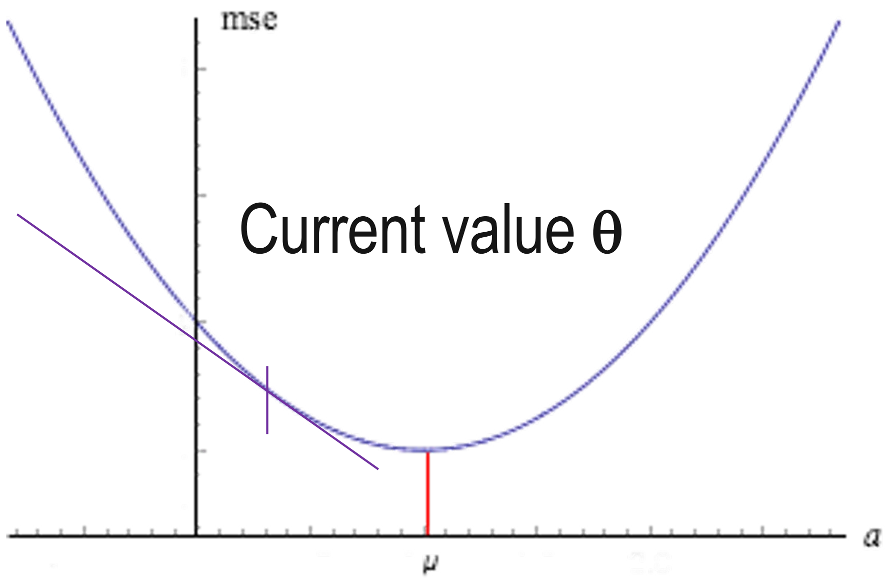
http://www.math.uah.edu/stat/expect/Variance.html

$$SSE(\theta) = \sum_{i=1}^{n} r_i(\theta)^2$$

We can compute the gradient numerically
… But sometimes better to use analytics (calculus)!

$$\nabla SSE(\theta) = \sum_{i=1}^{n} \frac{d}{d\theta} r_i(\theta)^2$$

Getting Closer



http://www.math.uah.edu/stat/expect/Variance.html

$$SSE(\theta) = \sum_{i=1}^{n} r_i(\theta)^2$$

We can compute the gradient numerically
… But sometimes better to use analytics (calculus)!

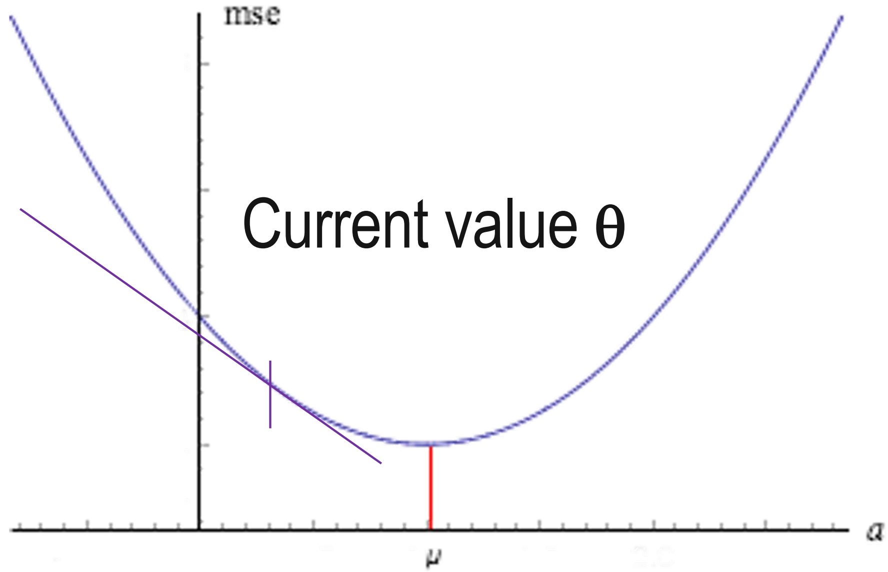
$$\nabla SSE(\theta) = \sum_{i=1}^{n} 2 \cdot \left(r_i(\theta) \cdot \frac{\partial r_i(\theta)}{\partial \theta} \right)$$

$$\frac{\partial r_i}{\partial \theta_j} = x_j^{(i)}$$

Getting Closer

$$SSE(\theta) = \sum_{i=1}^{n} r_i(\theta)^2$$

We can compute the gradient numerically
… But sometimes better to use analytics (calculus)!



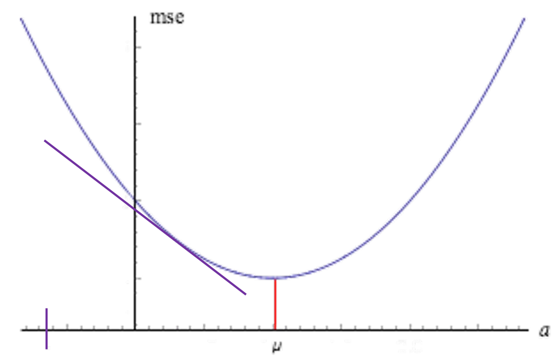
http://www.math.uah.edu/stat/expect/Variance.html

$$\nabla SSE(\theta) = \sum_{i=1}^{n} 2 \cdot \left(r_i(\theta) \cdot x_j^{(i)} \right)$$

$$\frac{\partial r_i}{\partial \theta_j} = x_j^{(i)}$$

$$\theta := \theta - \eta \, \nabla SSE(\theta)$$

Key questions



$$\theta := \theta - \eta \, \nabla SSE(\theta)$$

- **How big a step η to take?**
  - Too small and it takes a long time
  - Too big and it will be unstable
- **"Optimal:" scale η ~ 1/sqrt(iteration)**
  - Or maybe η ~ 1/iteration ???
- **Adaptive (a simple version)**
  - E.g. each time, increase step size by 10%
    - If error ever increases, cut set size in half

Stochastic Gradient Descent

- **If we have a very large data set, update the model after observing *each single* observation**
  - "online" or "streaming" learning

$$SSE(\theta) = \sum_{i=1}^{n} r_i(\theta)^2 \qquad \nabla SSE_i(\theta) = \frac{d}{d\theta} r_i(\theta)^2$$

$$\theta := \theta - \eta \, \nabla SSE_i(\theta)$$

Mini-batch

- **Update the model every k observations**
  - Batch size k (e.g. 50)
- **More efficient than pure stochastic gradient or full gradient descent**

$$MSE(\theta) = \frac{1}{n}\sum_{i=1}^{n} r_i(\theta)^2 \qquad \nabla MSE_k(\theta) = \frac{1}{k}\sum_{i=j}^{j+k} \frac{d}{d\theta} r_i(\theta)^2$$

$$\theta := \theta - \eta \, \nabla \boldsymbol{S}SE_k(\theta)$$

Adagrad

- **Define a *per-feature learning rate* for feature *j* as:**

$$\eta_{t,j} = \frac{\eta}{\sqrt{G_{t,j}}} \qquad G_{t,j} = \sum_{k=1}^{t} \underbrace{g_{k,j}^2}_{} \quad \frac{\partial}{\partial \theta_j} \mathrm{cost}_{\boldsymbol{\theta}}(\mathbf{x}_k, y_k)$$

- ***$G_{t,j}$* is the sum of squares of gradients of feature *j* over time *t***
- Frequently occurring features in the gradients get small learning rates; rare features get higher ones
- Key idea: “learn slowly” from frequent features but “pay attention” to rare but informative feature

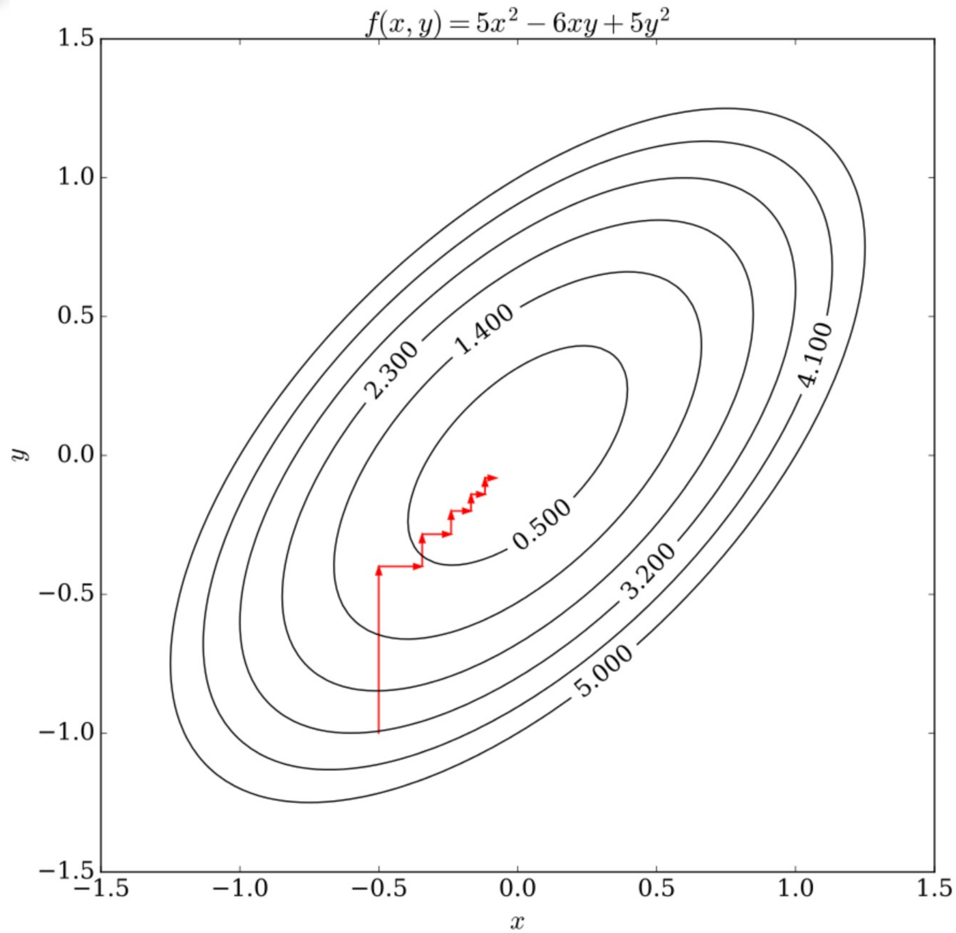
Adagrad

$$\eta_{t,j} = \frac{\eta}{\sqrt{G_{t,j}}} \qquad G_{t,j} = \sum_{k=1}^{t} g_{k,j}^2$$

$$\theta_j \leftarrow \theta_j - \frac{\eta}{\sqrt{G_{t,j}} + \zeta} g_{t,j}$$

In practice, add a small constant $\zeta > 0$ to prevent dividing by zero

For $||w||_1$ or $||y-h_\theta||_1$ use coordinate descent



Repeat:

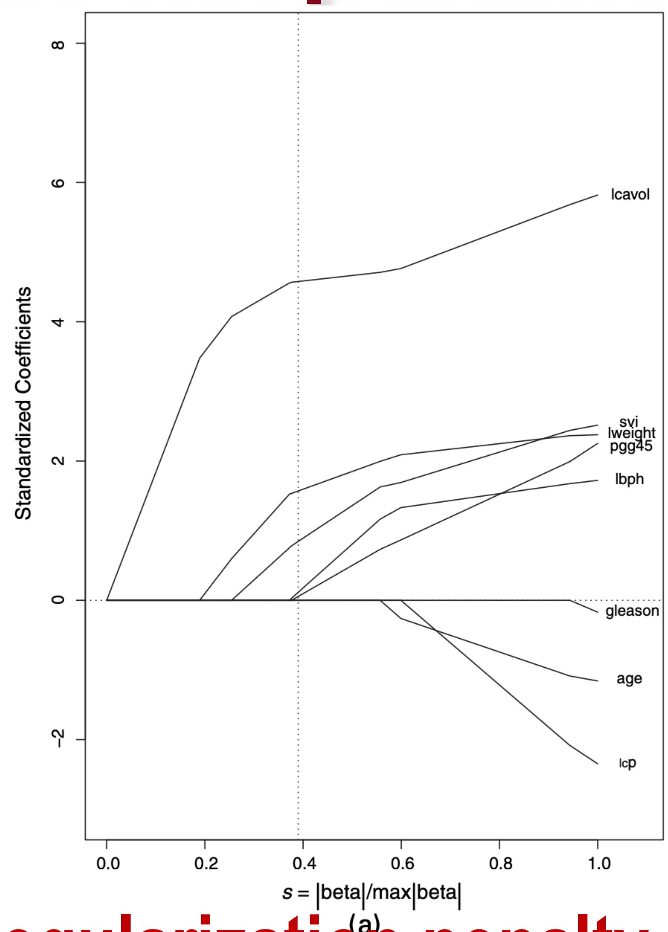
For j=1:p

$\theta_j := \theta_j - \eta dErr/d\theta_j$

https://en.wikipedia.org/wiki/Coordinate_descent

Elastic net parameter search

Size of coefficients



Regularization penalty (inverse)

Zou and Hastie

Recap: Gradient Descent

- **“Follow the slope” towards a minimum**
  - Analytical or numerical derivative
  - Need to pick step size
    - larger = faster convergence but instability
- **Lots of variations**
  - Coordinate descent
  - Stochastic gradient descent or mini-batch
- **Can get caught in local minima**
  - Alternative, *simulated annealing*, uses randomness