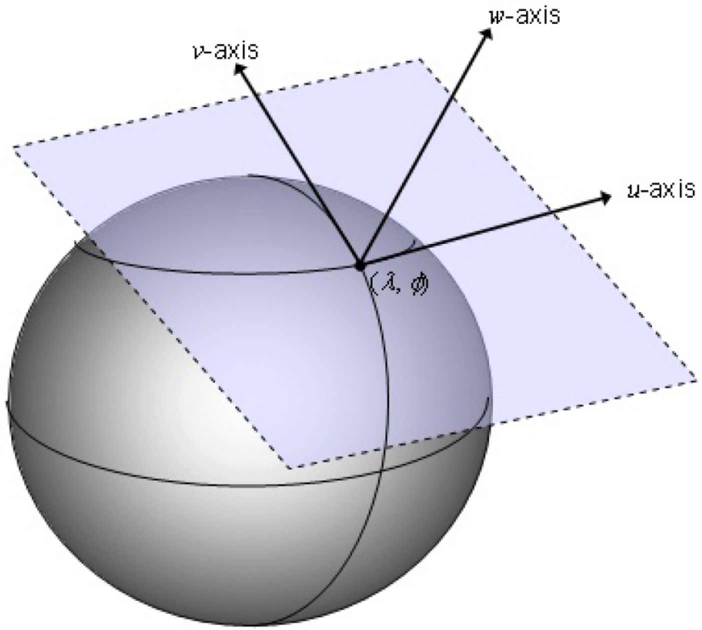
Lecture 9 part 3

Gradient Descent on Manifold

Weiqing Gu

Gradient descent on manifold

Tangent space:

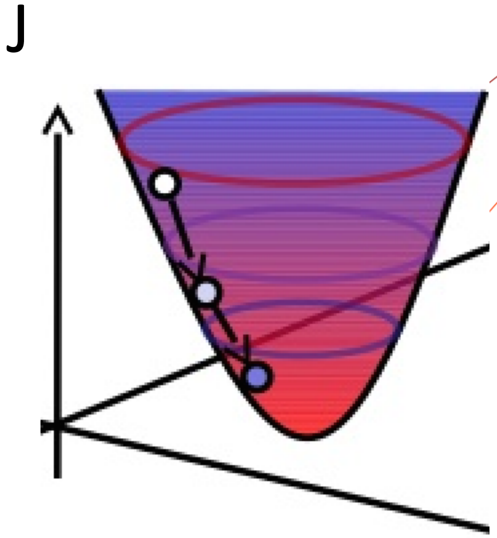


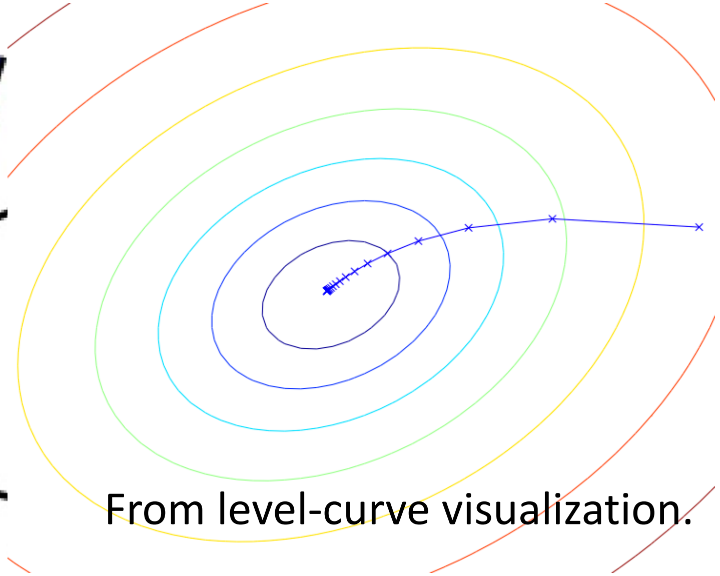
Riemmanian metric: scalar product $\langle u, v \rangle_g$ on the tangent space

First recall gradient descent in R^n

Use the gradient descent algorithm

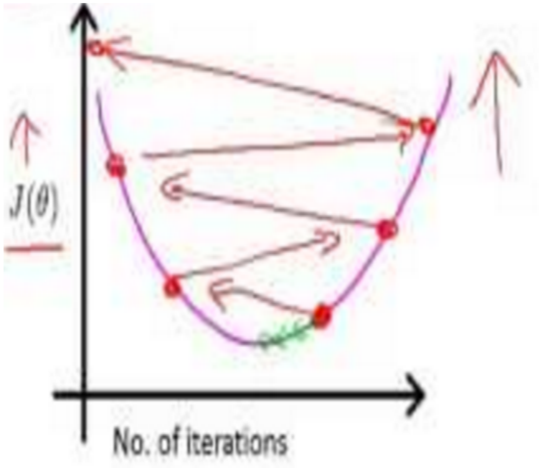
- Which starts with some initial θ, and repeatedly performs the update.
- Here α is called the learning rate.
- Geometrically, it repeatedly takes a step in the direction of steepest decrease of J.





From level-curve visualization.

Make α smaller if necessary.



Batch Gradient Descent (BGD)

Repeat until convergence {

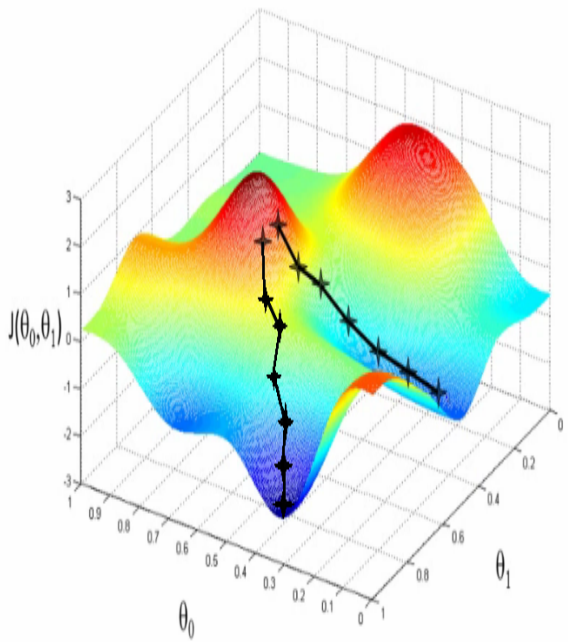
$$\theta_j := \theta_j + \alpha \underbrace{\sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}}_{-\,\partial J(\theta)/\partial \theta_j} \qquad \text{(for every } j\text{)}.$$

}

This is simply gradient descent on the original cost function J.

Remarks:

1) **This method looks at every example in the entire training set on every step**, and is called BGD.
2) It is well know that gradient descent can be susceptible to local minima in general (see the figure on right), **the optimization problem we have** posed here for linear regression **has only one global**, and no other local, **optima**; thus gradient descent always converges (assuming the learning rate α is not too large) to the global minimum.
3) **The key is that our J is a convex quadratic function.**



Stochastic Gradient Descent (SGD)

Loop {

for i=1 to m, {

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{)}.$$

}

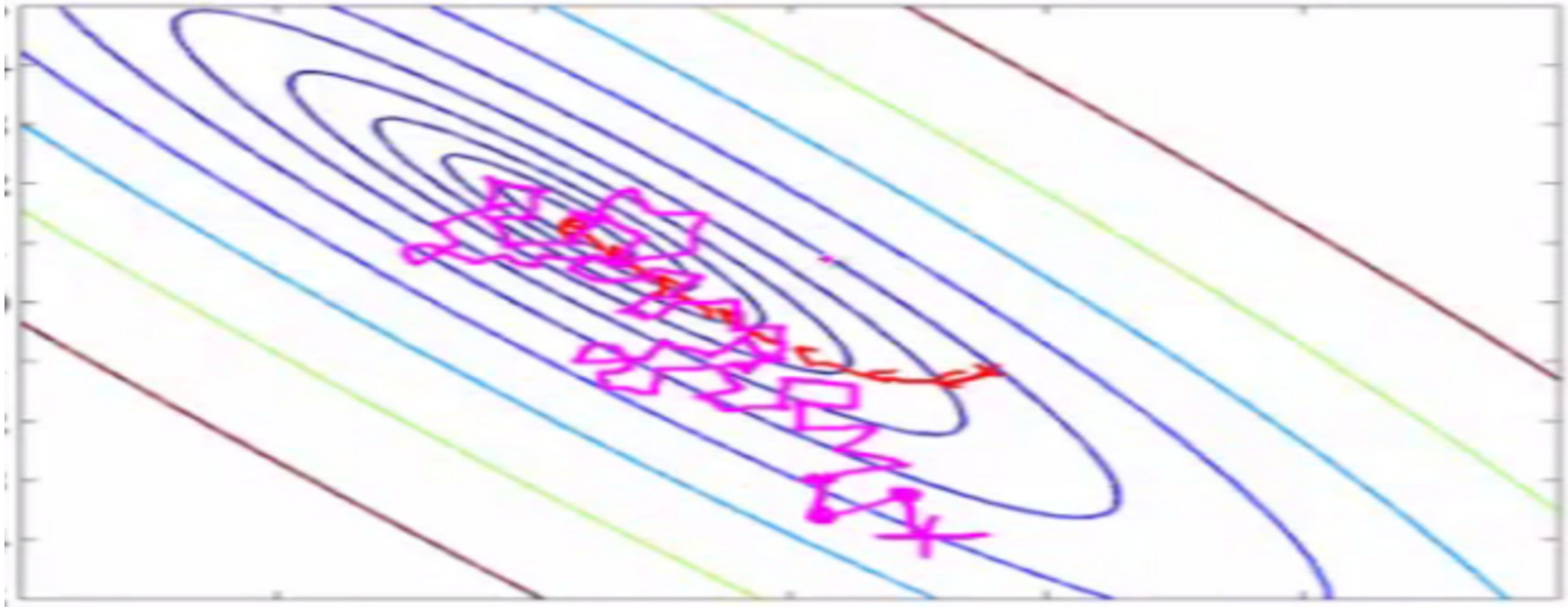
}

Remarks:

1) SGD **repeatedly run through the training set, and each time it encounters a training example, it updates the parameters** according to the gradient of the error with respect to that single training example only.
2) SGD **may never "converge" to the unique minimum**, and the parameters θ will keep oscillating around the minimum of J(θ); but **in practice** most of the values near the minimum will be **reasonably good approximations** to the true minimum.

Comparing Batch gradient descent with Stochastic gradient descent

- ***For big data***, often the training set is large***, people prefer use stochastic gradient descent*** instead of batch gradient descent.
- Since *BGD has to scan thru the entire training set before taking a single step*—a costly operation if m is large—*SGD can start making progress right away*, & continues to make progress with each example it looks at.
- *SGD can run on dynamical data sets*. As data coming, it updates the parameters.
- Often, **SGD gets θ "close" to the minimum much faster than BGD**.
- But SGD gets only approximation solution of θ. This is a **trade off** when dealing with big data.



Now let’s see how to extend SGD to a manifold

Consider $f : \mathcal{M} \to \mathbb{R}$ twice differentiable.

Riemannian gradient: tangent vector at x satisfying

$$\frac{d}{dt}|_{t=0} f(\exp_x(tv)) = \langle v, \nabla f(x) \rangle_g$$

Riemannian Hessian: based on the Taylor expansion

$$f(\exp_x(tv)) = t\langle v, \nabla f(x) \rangle_g + \frac{1}{2} t^2 v^T [\text{Hess } f(x)] v + O(t^3)$$

Second order Taylor expansion:

$$f(\exp_x(tv)) - f(x) \leq t\langle v, \nabla f(x) \rangle_g + \frac{t^2}{2} \|v\|_g^2 k$$

where k is a bound on the hessian along the geodesic.

Stochastic Gradient descent on manifold

Riemannian approximated gradient: $E_z(H(z_t, w_t)) = \nabla C(w_t)$

a tangent vector !

Stochastic gradient descent on $\mathcal{M}$: update

$$w_{t+1} \leftarrow \exp_{w_t}(-\gamma_t H(z_t, w_t))$$

w_{t+1} must remain on $\mathcal{M}$!