Numerical Computation

Yong Li

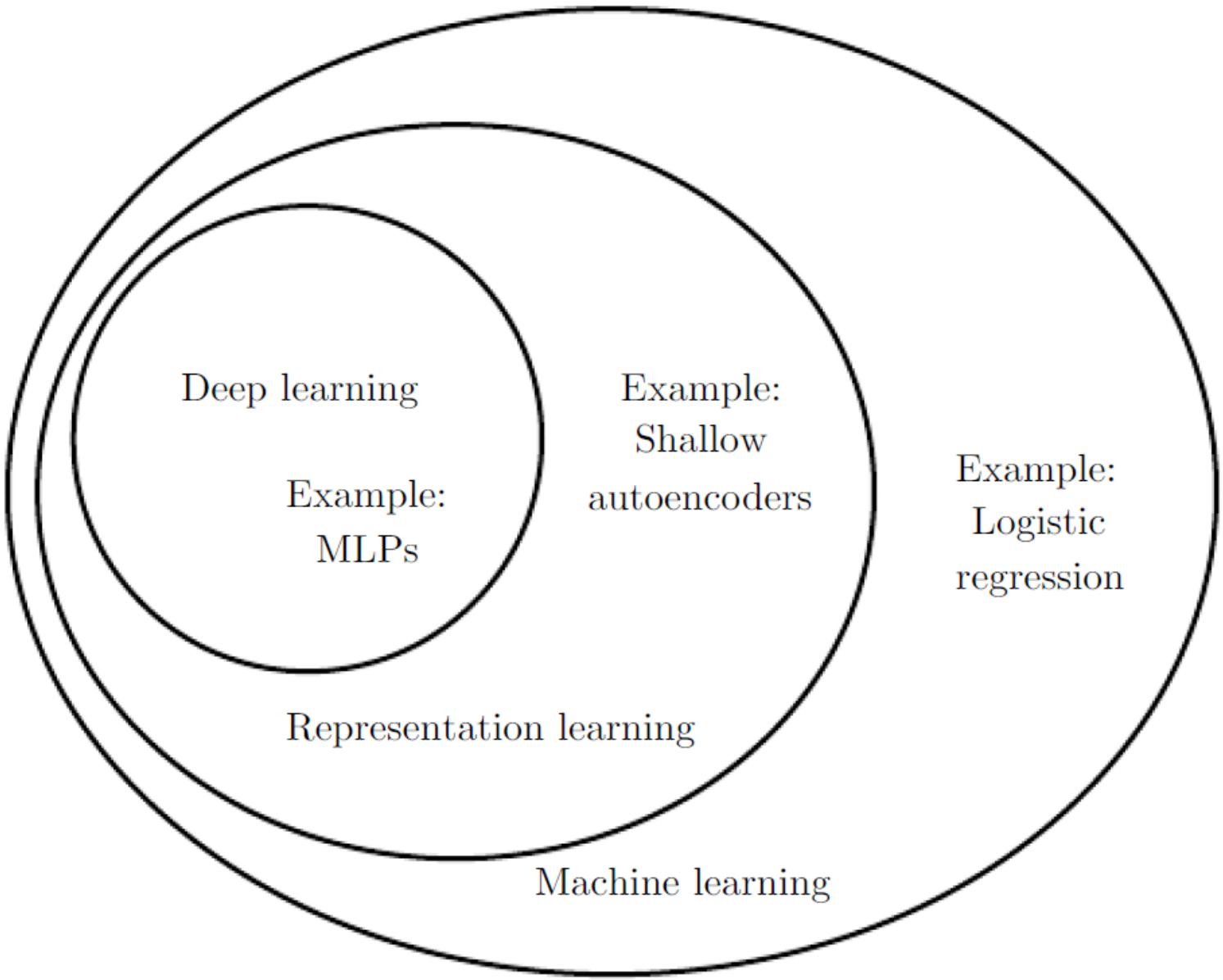
liyongforevercas@163.com
www.foreverlee.net

2016-11-20

Overview

1. Optimization in ML
2. Numerical Stability
   - Overflow and Underflow
   - Poor Conditioning
3. Optimization
   - Gradient descent
   - Jacobian and Hessian Matrices
   - Contrained Optimization
   - Example: LMS
   - More SGD Methods in Deep Learning

Optimization in ML



Optimization in ML

Model

Hypothesis space $F = \{f | Y = f(X)\}$

Strategy

To determine the loss function $L(Y, f(x))$, like 0-1 loss function and quadratic loss function.

Algorithm

The algorithm is to select the best model in the hypothesis space by optimizing the loss function.

Overflow and Underflow

Rounding error can cause algorithms that work in theory to fail in practice if they are not designed to minimize the accumulation of rounding error.

Underflow

- Underflow occurs when numbers near zero are rounded to zero. Zero or not makes difference for operation like division.
- For some operation can be avoided by operation transform like log.

Overflow

- Overflow occurs when numbers with large magnitude are approximated as ∞ or $-\infty$.
- For some function, it can be avoided by funcation transform, like $softmax(x)_i = \frac{exp(x_i)}{\sum_{j=1}^{n} exp(x_j)}$, with $z = x - max_i x_i$.

Luckily, we can rely on libraries that provide stable implementations like Theano.

Poor Conditioning

Conditioning

Conditioning refers to how rapidly a function changes with respect to small changes in its inputs.

When $A \in R^{n \times n}$ has an eigenvalue decomposition, its condition number is

$$\max_{i,j} \left| \frac{\lambda_i}{\lambda_j} \right|$$

, which is the ratio of the magnitude of the largest and smallest eigenvalue

Gradient-Based Optimization

Optimization

Optimization refers to the task of either minimizing or maximizing some function $f(x)$ by altering x

Gradient-Based Optimization

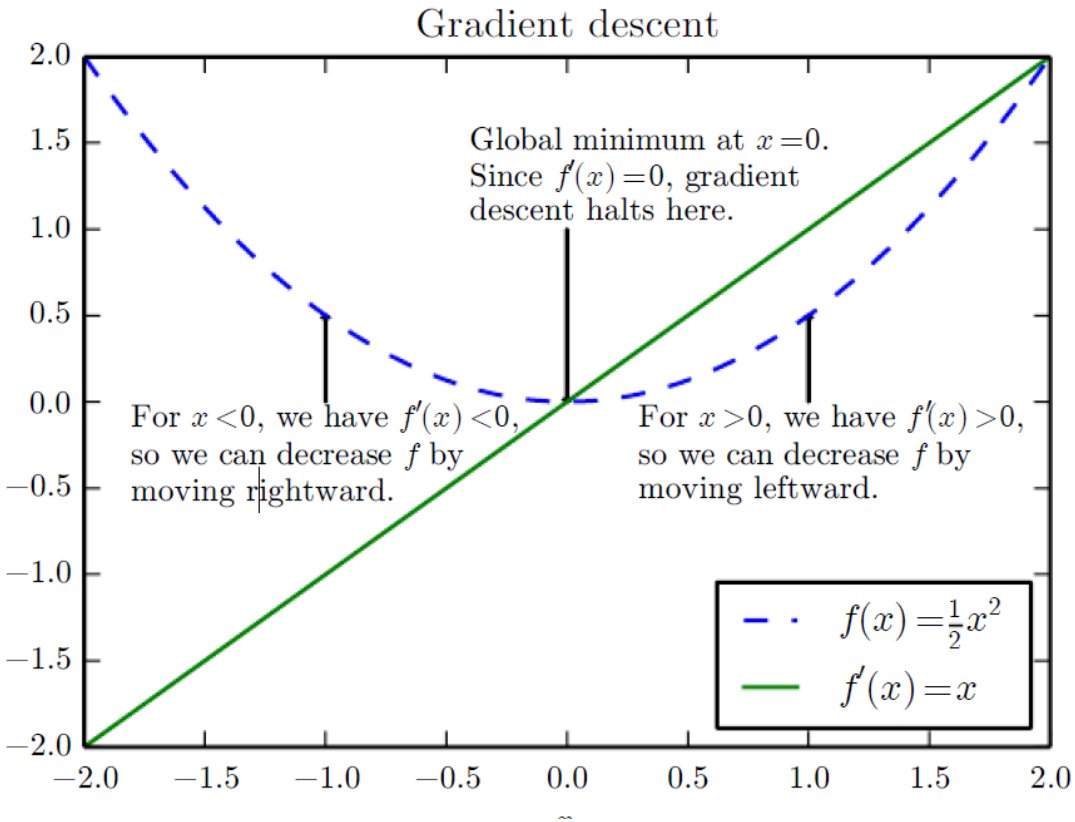
Optimization

Optimization refers to the task of either minimizing or maximizing some function $f(x)$ by altering x

objective function

The function we want to minimize or maximize is called the objective function or criterion. We may also call it the cost function, loss function or error function in the minimization case.

Gradient descent



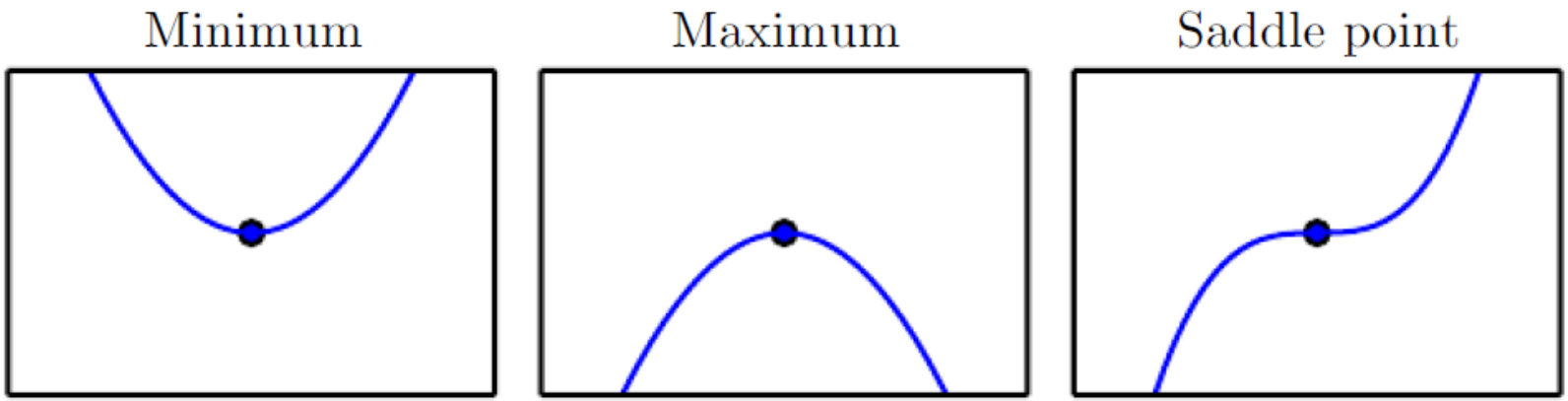
Gradient descent

It is to reduce $f(x)$ by moving x in small steps with opposite sign of the derivative, like $f(x - \epsilon sign(f'(x)))$

Critical Point

Critical Point

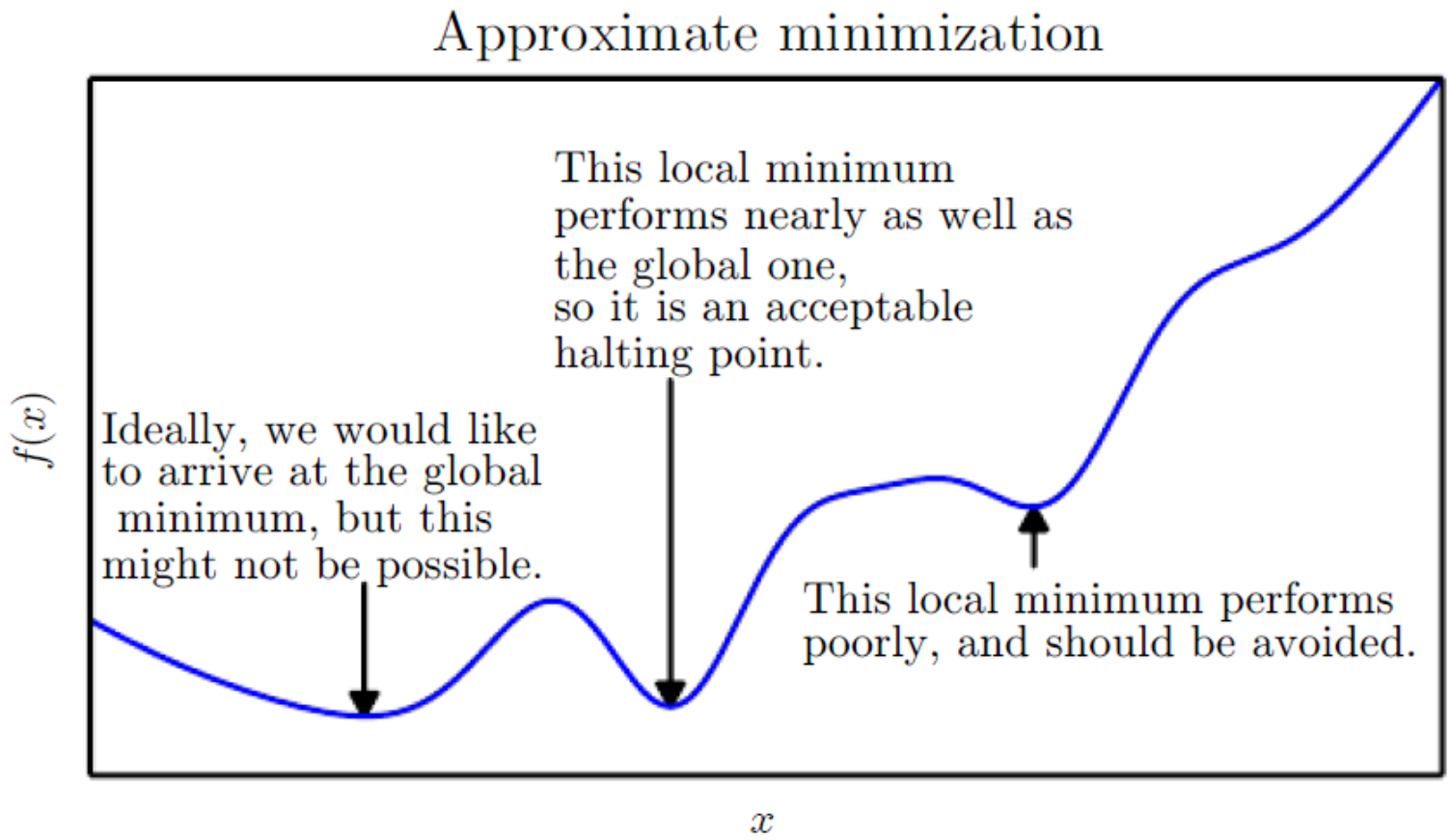
Critical points or stationary points are points with $f'(x) = 0$



Local Minimum and Mimimum

Mimimum

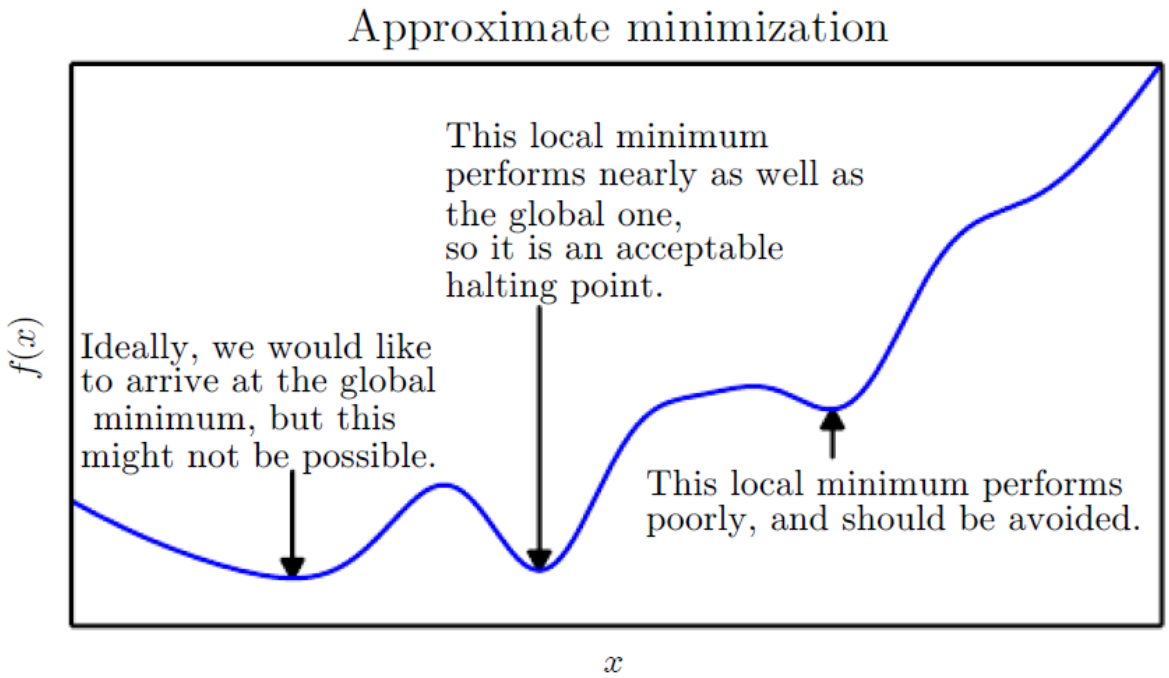
The point that obtains the absolute lowest value of $f(x)$



Derivative of Multidimensional Function

Partial Derivative

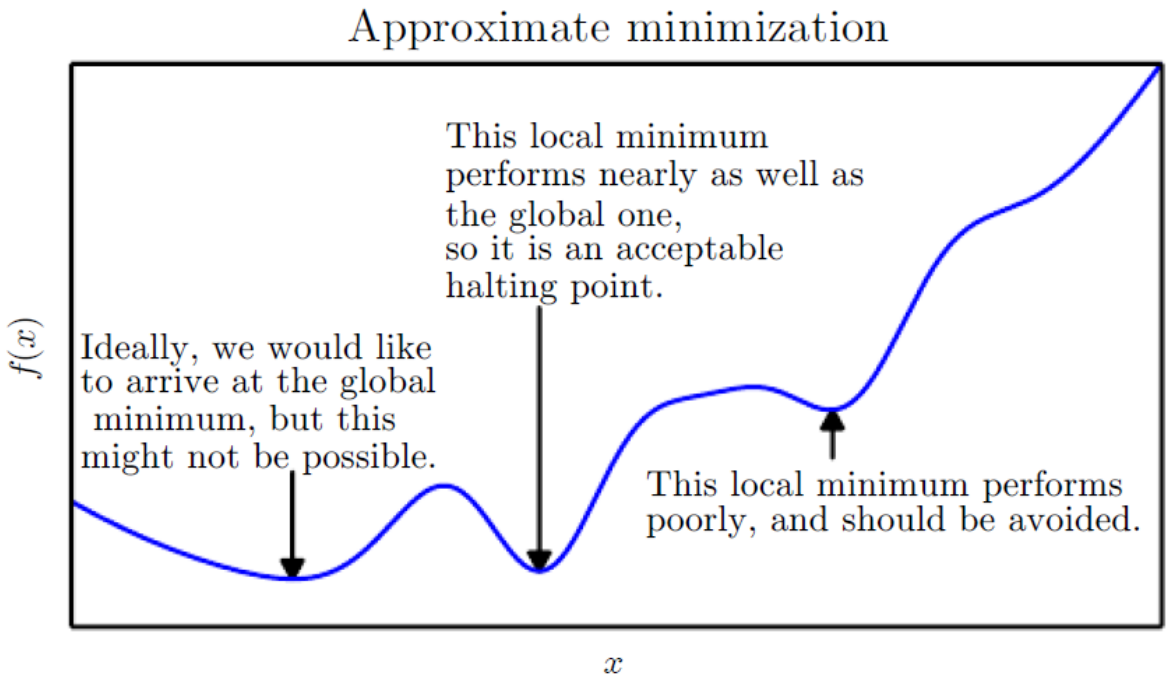
Partial derivative $\frac{\partial}{\partial x_i} f(x)$ measures how $f(x)$ changes as only the variable x_i increases at point x.
The gradient of f at point x is the vector containing all of the partial derivatives, denoted $\nabla_x f(x)$.



Derivative of Multidimensional Function

Derivative

For the minimization problem, the derivative is the steepest downhill direction.



Steepest Downhill Direction

The directional derivative in direction $\mathbf{u}$ is the derivative of the function $f(\mathbf{x}+\alpha\mathbf{u})$ with respect to α, evaluated at $\alpha = 0$, with the chain rule, $\frac{\partial}{\partial\alpha} f(\mathbf{x}+\alpha\mathbf{u}) = \mathbf{u}^T \nabla_x f(x)$.

$$\min_{\mathbf{u},\mathbf{u}^T\mathbf{u}=1} \mathbf{u}^T \nabla_x f(x)$$

$$\min_{\mathbf{u},\mathbf{u}^T\mathbf{u}=1} \|\mathbf{u}\|_2 \|\nabla_x f(x)\|_2 \cos\theta$$

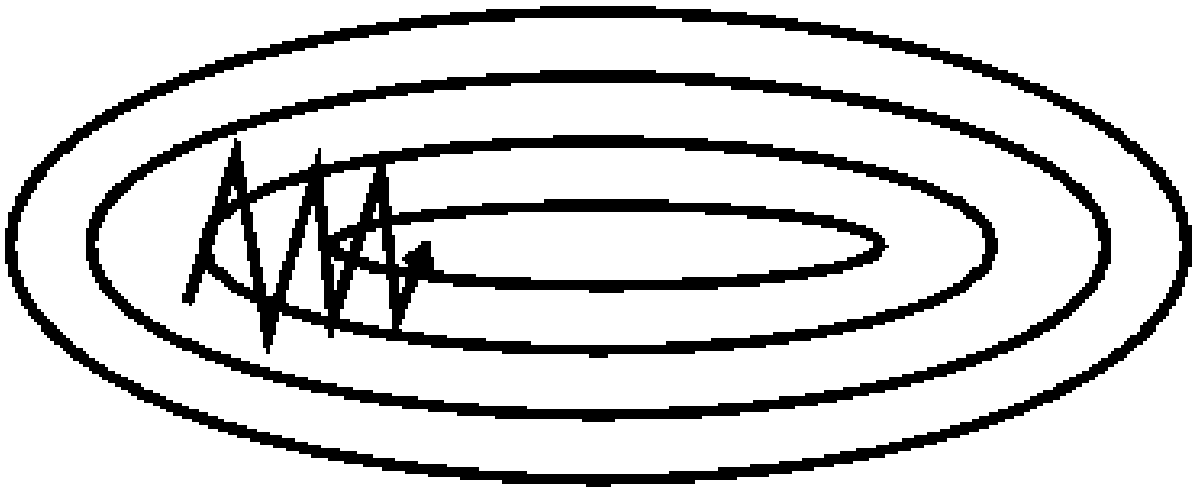
$$\min_{\mathbf{u},\mathbf{u}^T\mathbf{u}=1} \cos\theta$$

Steepest descent proposes a new point,

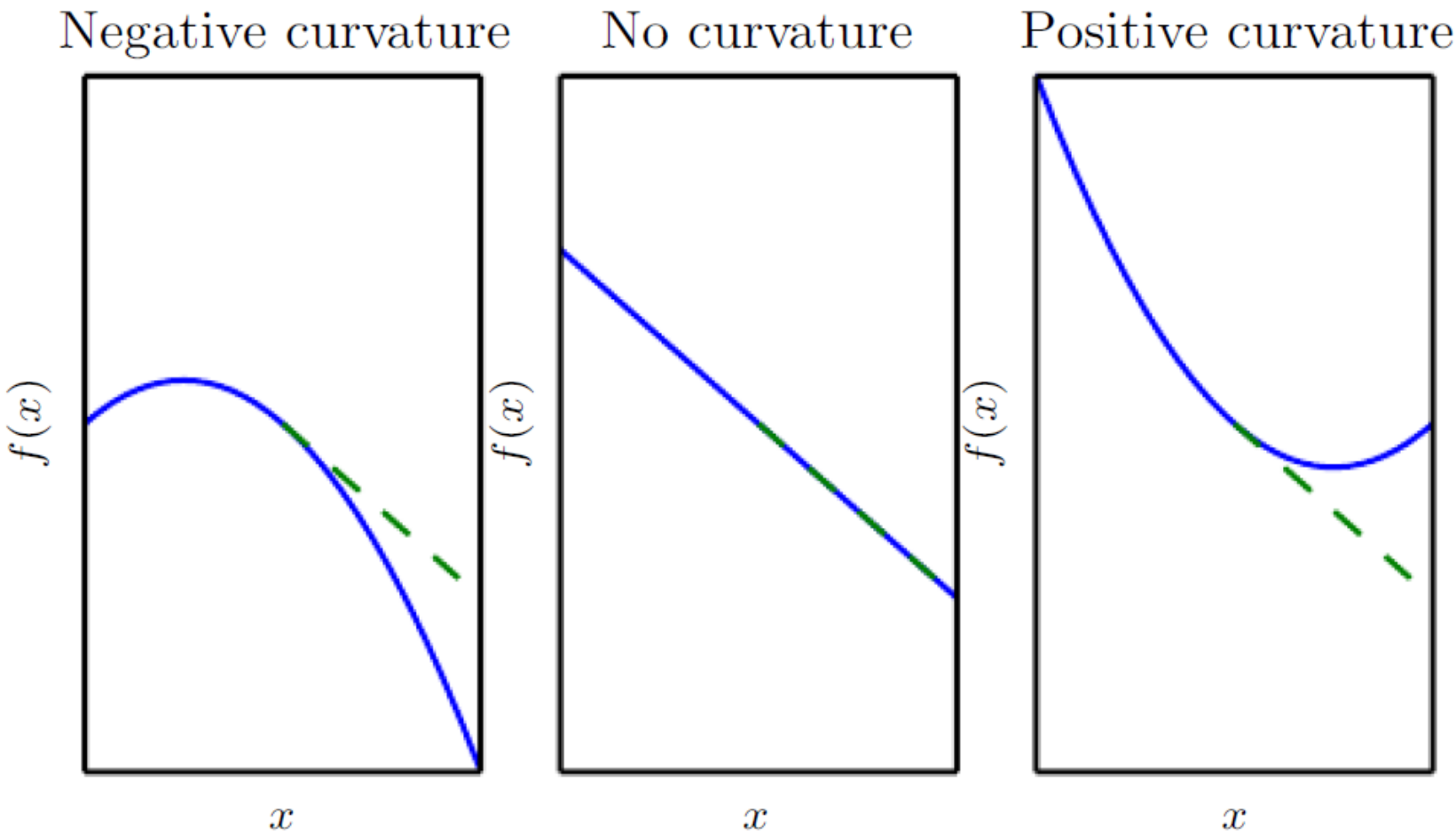
$$\mathbf{x}' = \mathbf{x} - \epsilon \nabla_x f(x)$$

where ϵ is the learning rate.

Weakness of SGD



Curvarure information



Jacobian and Hessian Matrices

Jacobian matrix

The partial derivatives of a function whose input and output are both vectors, Specifically, if we have a function $\mathbf{f} : \Re^m \to \Re^n$, then the Jacobian matrix $J \in \Re^{n \times m}$ of f is defined such that

$$J_{i,j} = \frac{\partial}{\partial \alpha} f(\mathbf{x})_i$$

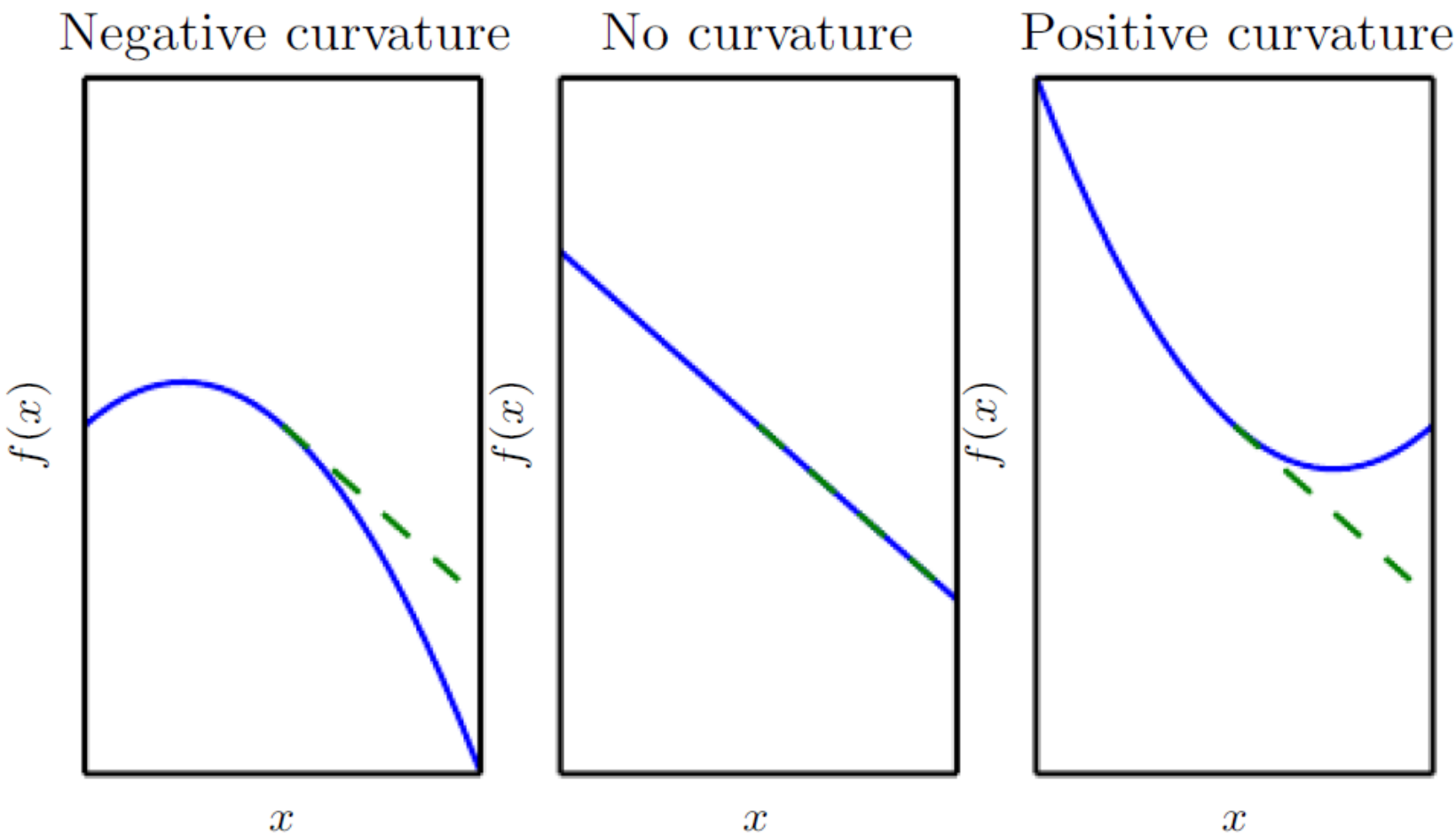
Hessian matrix

Hessian matrix is the second order derivative which is defined such that

$$H_{i,j} = \frac{\partial}{\partial x_i \partial x_j} f(\mathbf{x})$$

Hessian matrix is symmetric $H_{i,j} = H_{j,i}$

Relationship between Second Order Derivative and Curvarure



Optimal Step Size of Gradient Descent

Second-order Taylor series approximation to the function around $f(x)$ the current point $x^{(0)}$

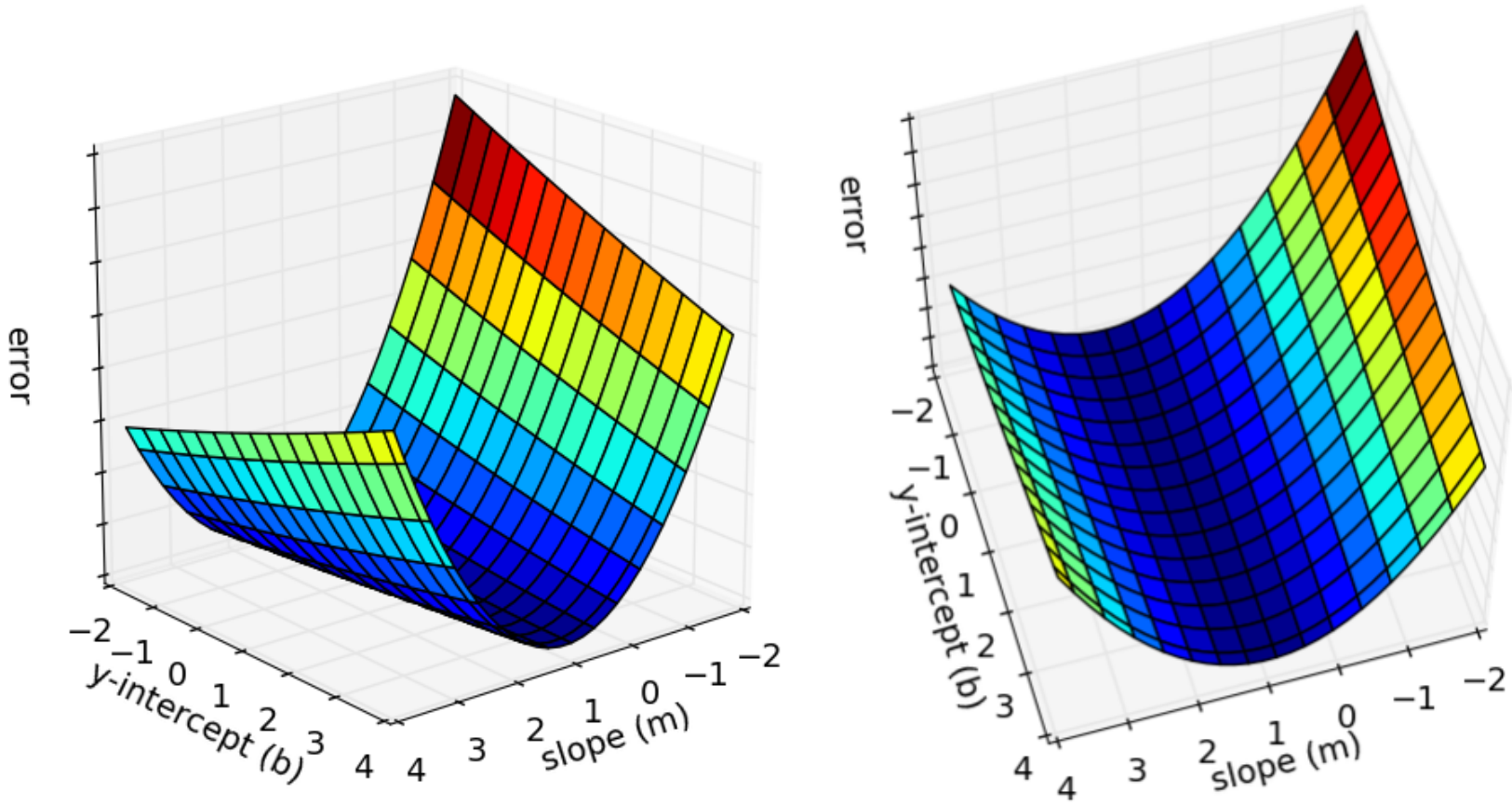
$$f(\mathbf{x}) \approx f(\mathbf{x}^{(0)}) + (\mathbf{x} - \mathbf{x}^{(0)})^T \mathbf{g} + \frac{1}{2}(\mathbf{x} - \mathbf{x}^{(0)})^T \mathbf{H}(\mathbf{x} - \mathbf{x}^{(0)})$$

where $\mathbf{g}$ is the gradient and $\mathbf{H}$ is the Hessian at $x^{(0)}$.
Update with gradient descent $\mathbf{x} \leftarrow x^{(0)} - \epsilon \mathbf{g}$
We can get

$$f(x^{(0)} - \epsilon \mathbf{g}) \approx f(\mathbf{x}^{(0)}) - \epsilon \mathbf{g}^T \mathbf{g} + \frac{1}{2}\epsilon^2 \mathbf{g}^T \mathbf{H} \mathbf{g}$$

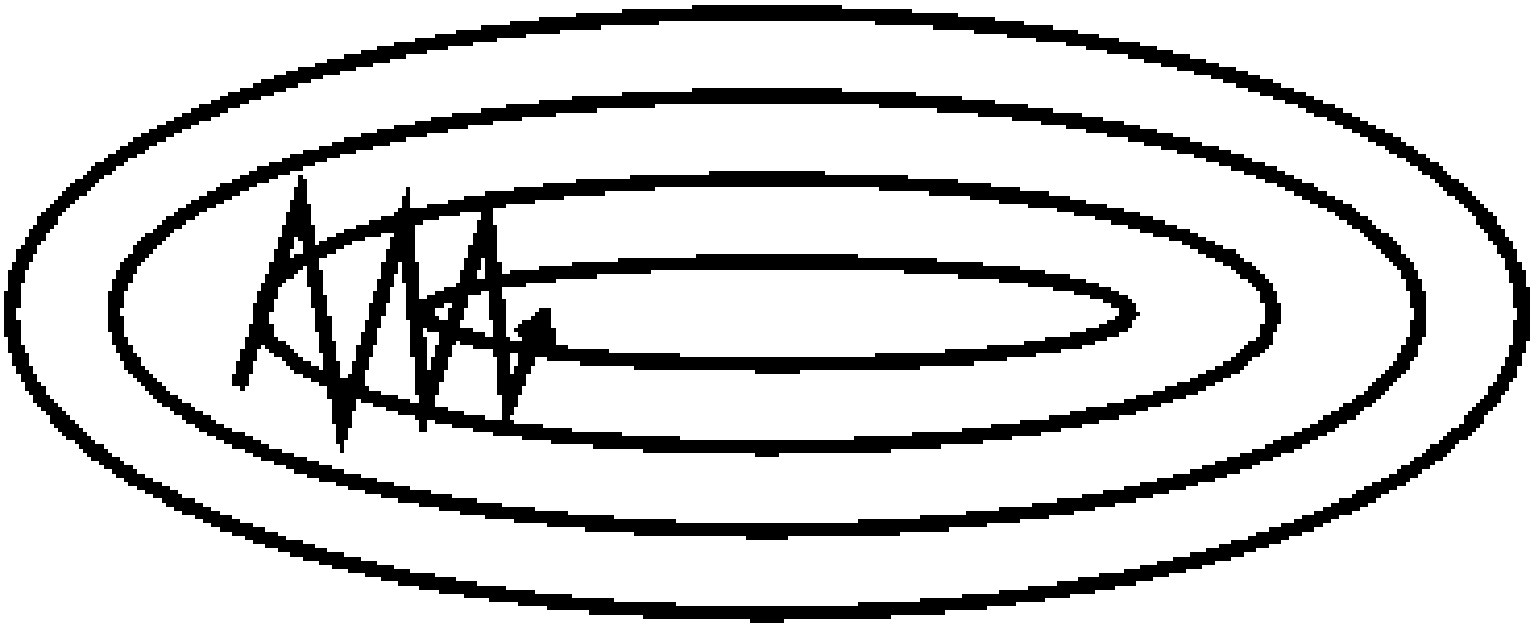
ϵ is impotant and can be optimizaed as follows, $\epsilon^* = \frac{\mathbf{g}^T \mathbf{g}}{\mathbf{g}^T \mathbf{H} \mathbf{g}}$.

Weakness of Gradient Descent



https://spin.atomicobject.com/wp-content/uploads/gradient_descent_error_surface.png

Weakness of Gradient Descent



Newtons method

Second-order Taylor series approximation to the function around $f(x)$ the current point $x^{(0)}$

$$f(\mathbf{x}) \approx f(\mathbf{x}^{(0)}) + (\mathbf{x} - \mathbf{x}^{(0)})^T \mathbf{g} + \frac{1}{2}(\mathbf{x} - \mathbf{x}^{(0)})^T \mathbf{H}(f)(x^{(0)})(\mathbf{x} - \mathbf{x}^{(0)})$$

We can obtain the critical point of this function as follows,

$$x^* = x^{(0)} - \mathbf{H}(f)(x^{(0)})^{-1} \nabla_x f(x^{(0)})$$

Constrained Optimization

Constrained optimization

It is to find the maximal or minimal value of $f(x)$ for values of x in some set S

The Karush-Kuhn-Tucker (KKT) approach provides a general solution to constrained optimization. With KKT approach, we introduce a new function called generalized Lagrangian or generalized Lagrange function.

$$L(\mathbf{x}, \lambda, \alpha) = f(\mathbf{x}) + \sum_i \lambda_i g^{(i)}(\mathbf{x}) + \sum_j \alpha_j h^j(\mathbf{x})$$

With the constrained set

$$S = \mathbf{x} | \forall i, g^{(i)}(\mathbf{x}) = 0 \text{ and } \forall j, h^{(j)}(\mathbf{x}) \leq 0$$

Then

$$\min_x \max_\lambda \max_{\alpha, \alpha \geq 0} L(\mathbf{x}, \lambda, \alpha)$$

has same optimal objective function as $\min_{x \in S} f(x)$

Linear Least Squares

It is to minimize the loss function as follows,

$$\min f(x) = \frac{1}{2}\|\mathbf{Ax} - \mathbf{b}\|_2^2$$

The gradient is as follows,

$$\nabla_x f(\mathbf{x}) = \mathbf{A}^T(\mathbf{Ax} - \mathbf{b}) = \mathbf{A}^T\mathbf{Ax} - \mathbf{A}^T\mathbf{b}$$

Algorithm 4.1 An algorithm to minimize $f(\boldsymbol{x}) = \frac{1}{2}||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||_2^2$ with respect to $\boldsymbol{x}$ using gradient descent.

Set the step size (ϵ) and tolerance (δ) to small, positive numbers.
while $||\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^\top\boldsymbol{b}||_2 > \delta$ **do**
  $\boldsymbol{x} \leftarrow \boldsymbol{x} - \epsilon\left(\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^\top\boldsymbol{b}\right)$
end while

Linear Least Squares with Constraint

It is to minimize the loss function with constraint $\mathbf{x}^T\mathbf{x} \leq 1$ as follows,

$$\min_{\mathbf{x}^T\mathbf{x}\leq 1} f(x) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$$

The Lagrangian is as follows, $L(x, \lambda) = f(x) + \lambda(x^T x - 1)$
Then we can solve the problem

$$\min_{\mathbf{x}} \max_{\lambda, \lambda \geq 0} L(\mathbf{x}, \lambda)$$

- Differentiating with respect to x, $x = (A^(T)A + 2\lambda\mathbf{I})^{(-1)}\mathbf{A}^T\mathbf{b}$
- λ can be performed with gradient ascent $\frac{\partial}{\partial\lambda}L(\mathbf{x}, \lambda) = \mathbf{x}^T\mathbf{x} - 1$

SGD and Mini-batch SGD

Stochastic gradient descent proceeds one example at a time instead of the entire training set,

$$\theta^{k+1} = \theta^k - \eta \frac{\partial L(\theta, x^{(i)}, y^{(i)})}{\partial \theta}$$

Mini-batch SGD use more than one training example to make each estimate of the gradient,

$$\theta^{k+1} = \theta^k - \eta \frac{\partial L(\theta, x^{(i:i+n)}, y^{(i:i+n)})}{\partial \theta}$$

Problem: manually adjust learning rate

Momentum SGD

Momentum SGD adds a fraction of the update vector of the past time step to the current update vector

$$v^k = \gamma v^{k-1} + \eta \frac{\partial L(\theta, x^{(i)}, y^{(i)})}{\partial \theta}$$

$$\theta^{k+1} = \theta^k - v^k$$

AdaGrad

AdaGrad adapts the learning rate to the parameters, performing larger updates for infrequent and smaller updates for frequent parameters

$$\theta^{k+1} = \theta^k - \eta^k \frac{\partial L(\theta, x^{(i)}, y^{(i)})}{\partial \theta}$$

where, $\eta^k = \frac{\eta}{\sqrt{\sum_{i=1}^{k}(g^i)^2 + \epsilon}}$, and ϵ is a smoothing term. Problem: gradually shrunk to an infinitesimally small number.

RMSprop

RMSprop uses a moving average of squared gradients instead,

$$\triangle\theta^k = -\frac{\eta}{RMS[g]_k} g_k$$

$$\theta^{k+1} = \theta^k + \triangle\theta^k$$

where $RMS[g]_k = \sqrt{E[g^2]_k + \epsilon}$, and $E[g^2]_k = \rho E[g^2]_{k-1} + (1-\rho) g_k^2$

AdaDelta

AdaDelta is an extension of Adagrad

$$\triangle\theta^k = -\frac{RMS[\triangle\theta]_{k-1}}{RMS[g]_k} g_k$$

$$\theta^{k+1} = \theta^k + \triangle\theta^k$$

where $RMS[g]_k = \sqrt{E[g^2]_k + \epsilon}$, and $E[g^2]_k = \rho E[g^2]_{k-1} + (1-\rho) g_k^2$

AdaDelta

AdaDelta is an extension of Adagrad

$$\triangle\theta^k = -\frac{RMS[\triangle\theta]_{k-1}}{RMS[g]_k} g_k$$

$$\theta^{k+1} = \theta^k + \triangle\theta^k$$

where $RMS[g]_k = \sqrt{E[g^2]_k + \epsilon}$, and $E[g^2]_k = \rho E[g^2]_{k-1} + (1-\rho) g_k^2$

ADAM

Adaptive Moment Estimation (ADAM)

$$m^k = \beta_1 m^{k-1} + (1-\beta_1) g^k$$

$$v^k = \beta_2 v^{k-1} + (1-\beta_2)(g^k)^2$$

$$\theta^{k+1} = \theta^k - \eta \frac{m^k}{\sqrt{v^k} + \epsilon}$$

Conclusion

1. Optimization in ML
2. Numerical Stability
   - Overflow and Underflow
   - Poor Conditioning
3. Optimization
   - Gradient descent
   - Jacobian and Hessian Matrices
   - Contrained Optimization
   - Example: LMS
   - More SGD Methods in Deep Learning

References

Ian Goodfellow and Yoshua Bengio and Aaron Courville (2016)

Deep Learning

MIT Press Chap4, 80 – 97.

Thanks for your attention!

Q & A

liyongforevercas@163.com
www.foreverlee.net