CSE463: Neural Networks

Bipolar Perceptron Learning & Optimization

by:

Hossam El Din Hassan Abd El Munim

حسام الدين حسن عبد المنعم

Computer & Systems Engineering Dept.,

Ain Shams University,

1 El-Sarayat Street, Abbassia, Cairo 11517

Many slides and images are taken from the internet.

Dataset split

Training Images

- Train classifier

Validation Images

- Measure error
- Tune model hyperparameters

Testing Images

- Secret labels
- Measure error

Random train/validate splits = cross validation

Training

Training Images

Training Labels

Image Features

Training

Learned classifier

Testing

Test Image

Image Features

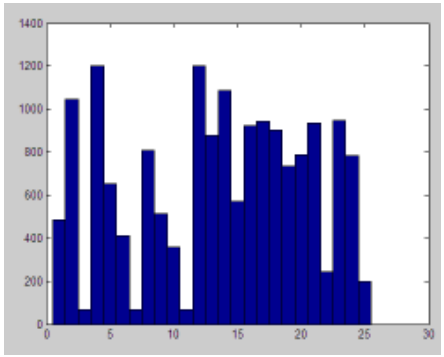
Apply classifier

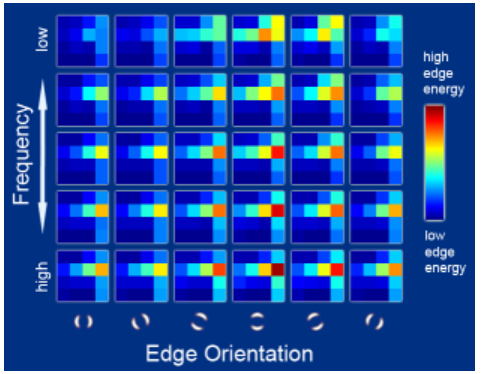
Prediction

Slide credit: D. Hoiem and L. Lazebnik

Features

- Raw pixels
- Histograms
- Templates
- SIFT descriptors
  - GIST
  - ORB
  - HOG….





Training

Training Images

Training Labels

Image Features

Training

Learned classifier

Testing

Test Image

Image Features

Apply classifier

Prediction

Slide credit: D. Hoiem and L. Lazebnik

Training

Training Images

Training Labels

Image Features

Training

Learned classifier

Testing

Test Image

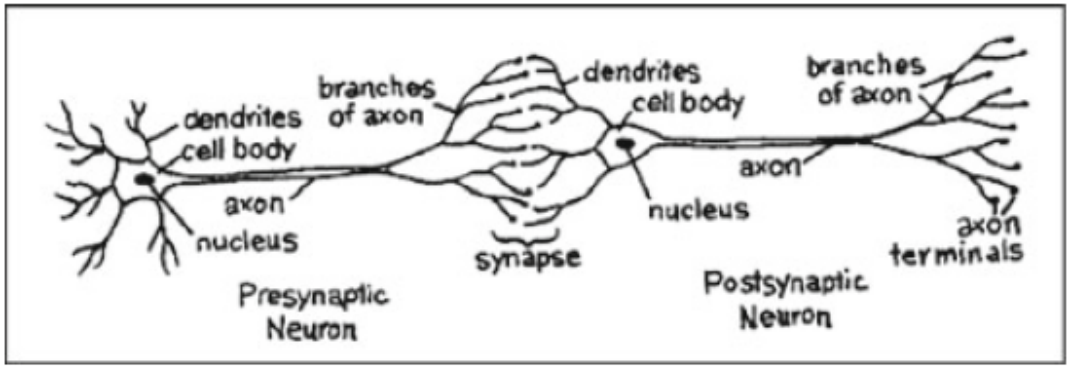
Image Features

Apply classifier

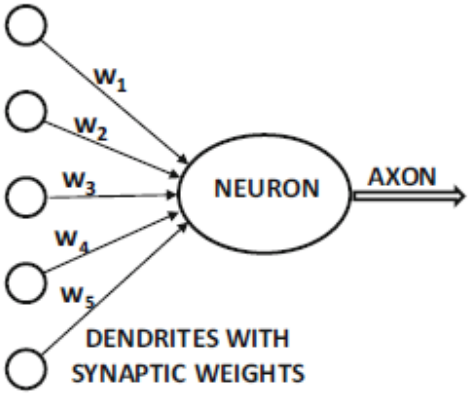
Prediction

Slide credit: D. Hoiem and L. Lazebnik

Simulation of biological mechanism in artificial neural networks



(a) Biological neural network



(b) Artificial neural network

Figure 1.1: The synaptic connections between neurons. The image in (a) is from "*The Brain: Understanding Neurobiology Through the Study of Addiction* [598]." Copyright ©2000 by BSCS & Videodiscovery. All rights reserved. Used with permission.

Deep Learning and Others

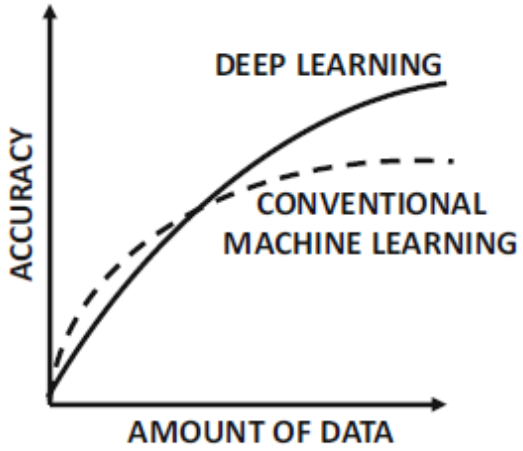
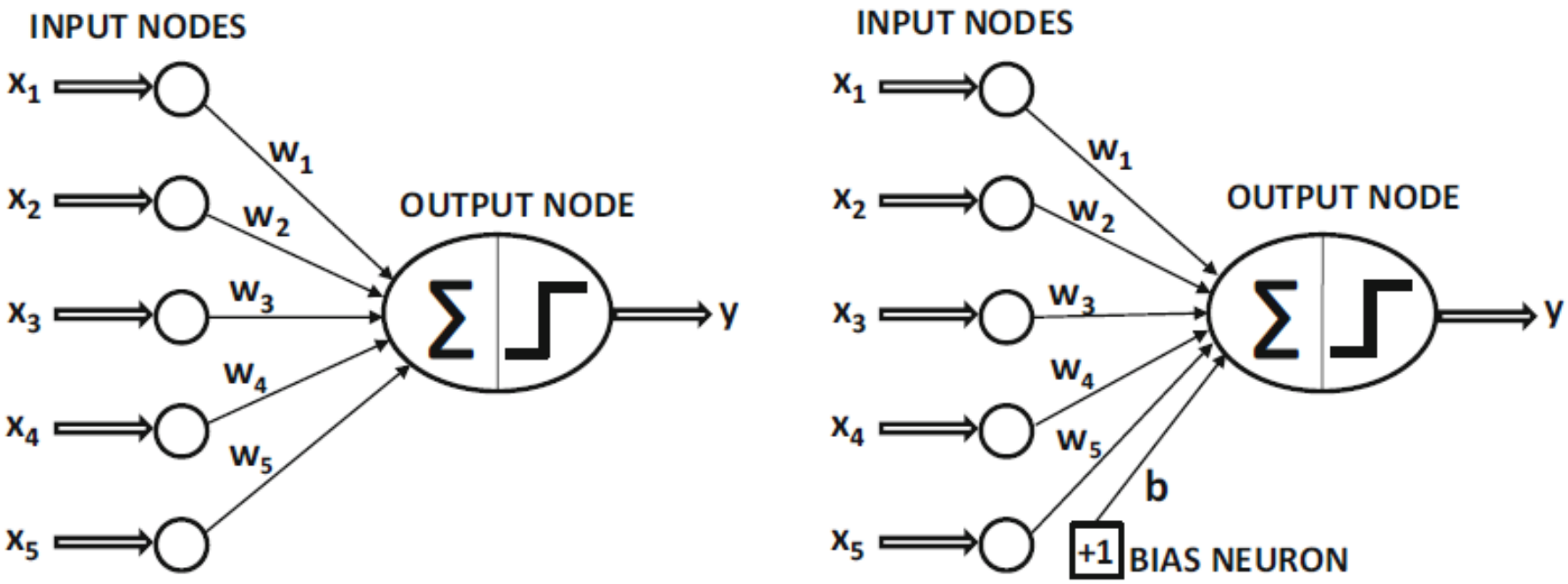


Figure 1.2: An illustrative comparison of the accuracy of a typical machine learning algorithm with that of a large neural network. Deep learners become more attractive than conventional methods primarily when sufficient data/computational power is available. Recent years have seen an increase in data availability and computational power, which has led to a "Cambrian explosion" in deep learning technology.

Humans Versus Computers: Stretching the Limits of Artificial Intelligence

- Humans and computers are inherently suited to different types of tasks.
- For example, computing the cube root of a large number is very easy for a computer, but it is extremely difficult for humans.
- On the other hand, a task such as recognizing the objects in an image is a simple matter for a human, but has traditionally been very difficult for an automated learning algorithm.
- It is only in recent years that deep learning has shown an accuracy on some of these tasks that exceeds that of a human.
- In fact, the recent results by deep learning algorithms that surpass human performance [184] in (some narrow tasks on) image recognition would not have been considered likely by most computer vision experts as recently as 10 years ago.

The Basic Architecture of Neural Networks



(a) Perceptron without bias

(b) Perceptron with bias

Figure 1.3: The basic architecture of the perceptron

Single Computational Layer: The Perceptron

The input layer contains d nodes that transmit the d features $\overline{X} = [x_1 \ldots x_d]$ with edges of weight $\overline{W} = [w_1 \ldots w_d]$ to an output node. The input layer does not perform any computation in its own right. The linear function $\overline{W} \cdot \overline{X} = \sum_{i=1}^{d} w_i x_i$ is computed at the output node. Subsequently, the sign of this real value is used in order to predict the dependent variable of $\overline{X}$. Therefore, the prediction $\hat{y}$ is computed as follows:

$$\hat{y} = \text{sign}\{\overline{W} \cdot \overline{X} + b\} = \text{sign}\{\sum_{j=1}^{d} w_j x_j + b\}$$

Output $\in$ {-1,+1}

Classification boundary is marked by:

$w_1x_1 + w_2x_2 + \ldots + w_dx_d + b = 0$

Perceptron as a Binary Classifier
Only Two Categories

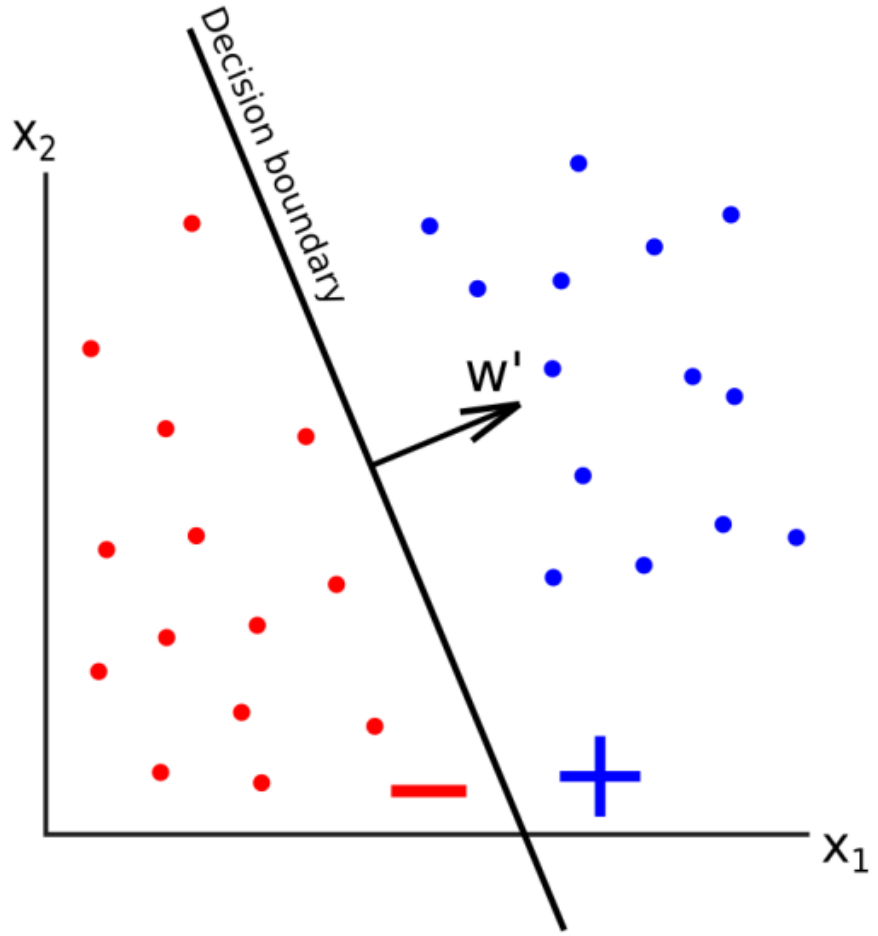
Perceptron as a Binary Classifier (2D Case)

$\mathbf{X} = [x_1\ x_2\ 1]^T$

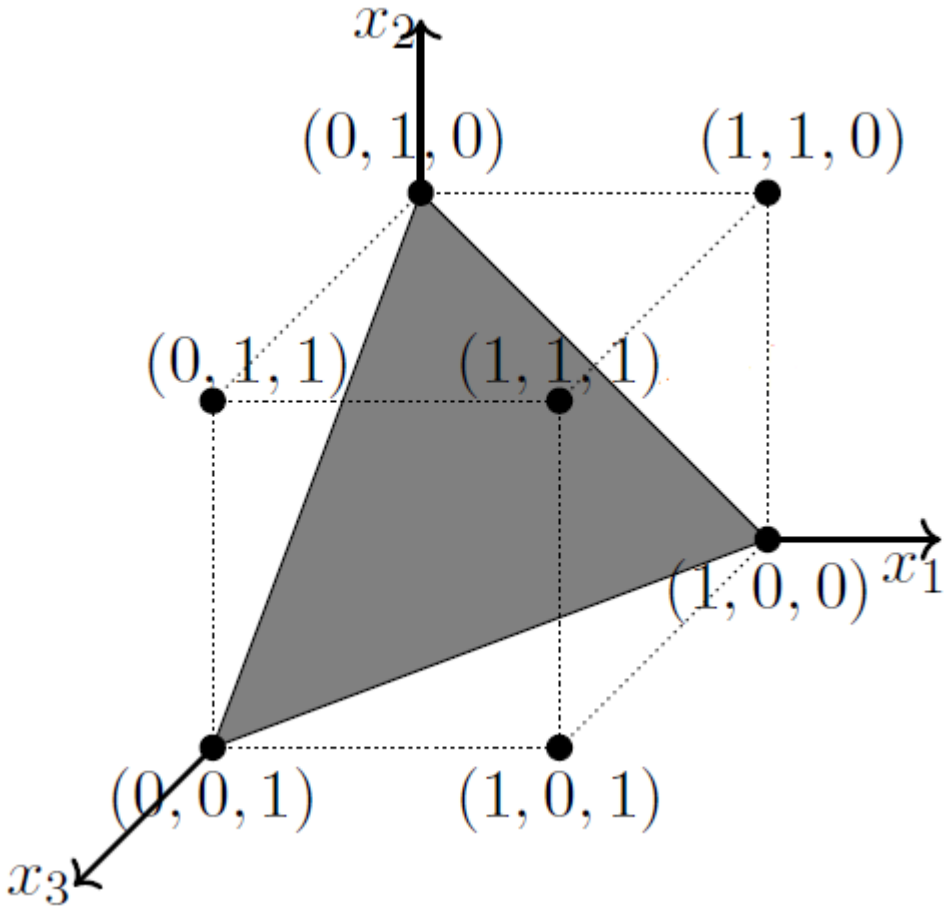
$\mathbf{W} = [w_1\ w_2\ w_0]^T$

$\mathbf{W.X = W^TX = 0}$

$w_1x_1 + w_2x_2 + w_0 = 0$



Perceptron as a Binary Classifier (3D Case)



$$\mathbf{X} = [x_1\ x_2\ x_3\ 1]^T$$

$$\mathbf{W} = [w_1\ w_2\ w_3\ w_0]^T$$

$$\mathbf{W.X} = \mathbf{W}^T\mathbf{X} = 0$$

$$w_1x_1 + w_2x_2 + w_3x_3 + w_0 = 0$$

What do you think about higher dimensions?

AND Function & Parameter Estimation

$(-1)w_1+(-1)w_2+w_0 < 0$

$(-1)w_1+(+1)w_2+w_0 < 0$

$(+1)w_1+(-1)w2+w_0 < 0$

$(+1)w_1+(+1)w2+w_0 \geq 0$

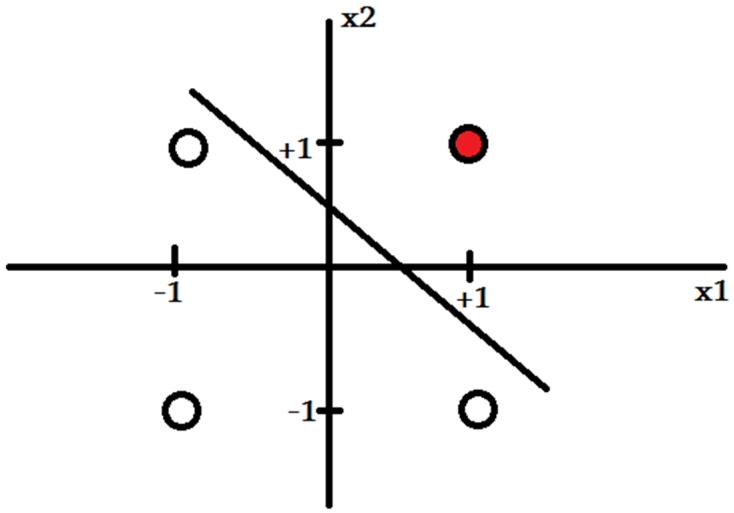
x_1	x_2	y
-1	-1	-1
-1	+1	-1
+1	-1	-1
+1	+1	+1

Second + Third = $2w_0 < 0$

First + Third = $-2w_2 + 2w_0 < 0$

Take $w_0 = -1$ & then $w_2 > -1$…Set $w_2 = 2$

Substitute in the first $w_1 > -3$…Set $w_1 = 2$



Perceptron Training: Learning W

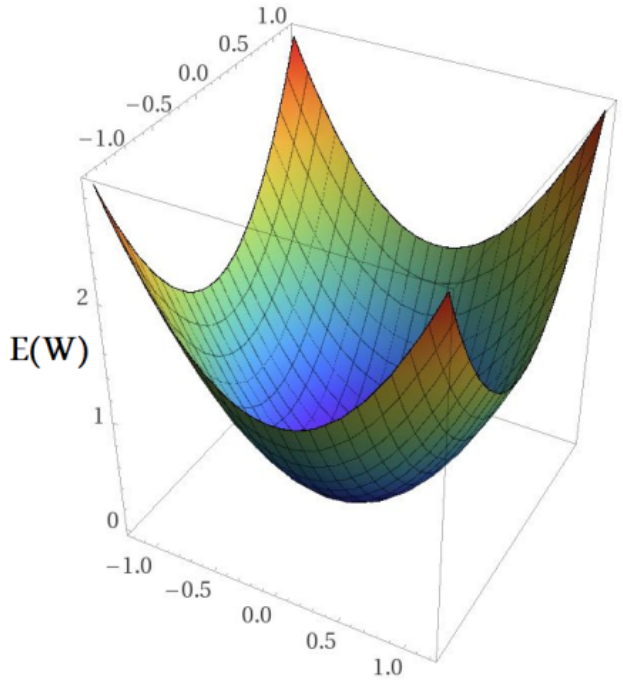
Given a data set, D = {($\mathbf{X}_1$, y_1), ($\mathbf{X}_2$, y_2)... ($\mathbf{X}_N$, y_N)} of labelled feature vectors where y ϵ {-1, +1}, we to estimate the vector **W** that minimizes the following objective function which measure the difference between the actual and desired outputs:-

$$E(\mathbf{W}) = (y_1\text{-sign}(W.\mathbf{X}_1))^2 + (y_2\text{-sign}(W.\mathbf{X}_2))^2 + \ldots + (y_N\text{-sign}(W.\mathbf{X}_N))^2$$

Training & Optimization

Given a data set, D = {($\mathbf{X}_1$, y_1), ($\mathbf{X}_2$, y_2)... ($\mathbf{X}_N$, y_N)} of labelled feature vectors where y ϵ {-1, +1}, we to estimate the vector **W** that minimizes the following objective function which measure the difference between the actual and desired outputs:-

$$E(\mathbf{W}) = (y_1\text{-sign}(W.\mathbf{X}_1))^2 + (y_2\text{-sign}(W.\mathbf{X}_2))^2 + \ldots + (y_N\text{-sign}(W.\mathbf{X}_N))^2$$

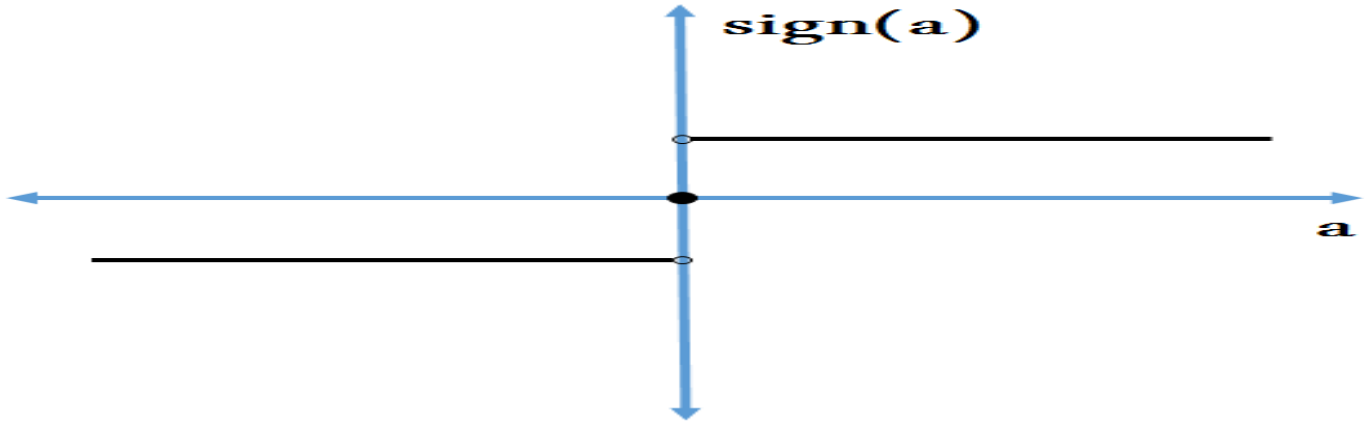


Derivative-based Optimization

- We need to find W that satisfies the following condition:-

$$\frac{\partial E}{\partial W} = 0$$

- First, how can we solve such an equation in multi dimensions?
- Second, the sign function has a derivative of zero every where except at the origin. Can we use a better objective function?



Derivative-based Optimization

- First, how can we solve such an equation in multi dimensions?

- An intelligent solution is made using the gradient descent technique.

- Second, the sign function has a derivative of zero every where except at the origin. Can we use a better objective function?

- Other objective functions cane be used.

Gradient Descent Optimization

Introduction

$\min f(\mathbf{X})$

subject to: $g_i(\mathbf{X}) \leq 0, i=1\ldots m$

$h_j(\mathbf{X})=0, j=1\ldots n$

$\mathbf{X} = [x_1\ x_2 \ldots x_d]^T$

$f: R^d \rightarrow R$

$g: R^d \rightarrow R$

$h: R^d \rightarrow R$

- Minimize means find the parameters vector **X**.
- The number of dimensions = d
- The constraints functions are g and h.

Introduction

- If a point x^* corresponds to the minimum value of the function $f(x)$, the same point also corresponds to the maximum value of the negative of the function, $-f(x)$. Thus optimization can be taken to mean minimization since the maximum of a function can be found by seeking the minimum of the negative of the same function.

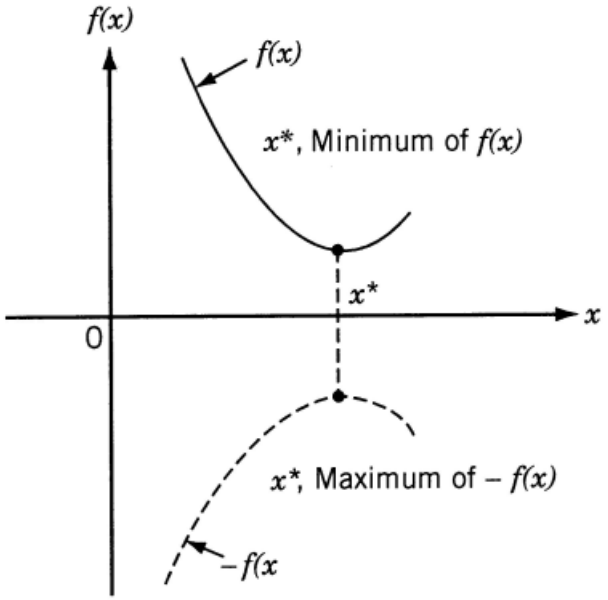
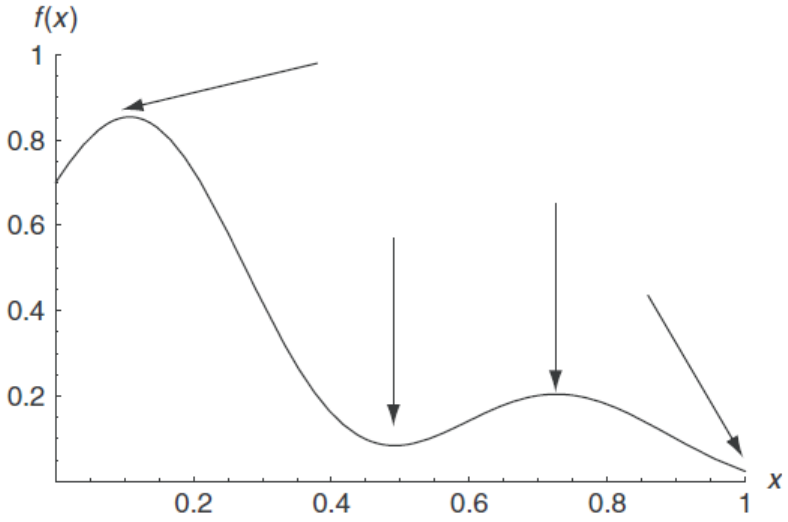


Figure 1.1 Minimum of $f(x)$ is same as maximum of $-f(x)$.

Local vs Global



The arrows indicate the location of local and global optima for a function defined on the closed interval [0, 1]. *The leftmost arrow points to the global optimum, whereas the rightmost arrow indicates the global minimum. Note that the global minimum is not, strictly speaking, a local minimum, since a neighbourhood cannot be defined at* $x = 1$.

In addition to local optima, the concept of *global* optima is essential in optimization: a function $f : \mathbf{D} \rightarrow \mathbf{R}$, has a **global minimum** at a point x^* if $f(\mathbf{x}) \geq f(\mathbf{x}^*)\ \forall x \in \mathbf{D}$.

Challenges

- We need to find X that satisfies the following necessary condition:-

$$\nabla f(\boldsymbol{X}) = 0$$

- First, how can we solve such an equation in multi dimensions? In deep learning, we may have millions of parameters.
- Unfortunately, we need to make another test to make sure it is a true minima. You need to check the Hessian matrix which is a d X d matrix to be a positive definite.

The Hessian Matrix

Specifically, suppose $f\colon \mathbb{R}^n \to \mathbb{R}$ is a function taking as input a vector $\mathbf{x} \in \mathbb{R}^n$ and outputting a scalar $f(\mathbf{x}) \in \mathbb{R}$; if all second partial derivatives of f exist and are continuous over the domain of the function, then the Hessian matrix $\mathbf{H}$ of f is a square $n \times n$ matrix, usually defined and arranged as follows:

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

or, by stating an equation for the coefficients using indices i and j:

$$\mathbf{H}_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}.$$

The determinant of the above matrix is also sometimes referred to as the Hessian.

Necessary condition

- If a function $f(x)$ is defined in the interval $a \leq x \leq b$ and has a relative minimum at $x = x^*$, where $a < x^* < b$, and if the derivative $df(x)/dx = f'(x)$ exists as a finite number at $x = x^*$, then $f'(x^*)=0$

- The theorem does not say that the function necessarily will have a minimum or maximum at every point where the derivative is zero. *e.g.* $f'(x)=0$ at $x= 0$ for the function shown in figure. However, this point is neither a minimum nor a maximum. In general, a point x^* at which $f'(x^*)=0$ is called a *stationary point.*

Necessary condition
Saddle Point

$$f(x) = x^3$$

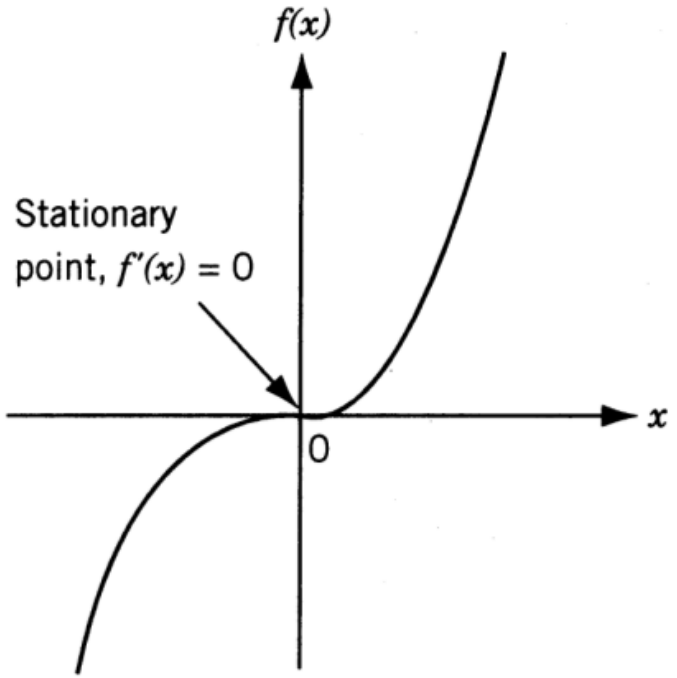
First Derivative = $df/dx = 3x^2 = 0$ then $x = 0$

But

Second derivative = $6x = 0$

Third derivative = $6 \neq 0$

(Odd Derivative)



Necessary condition

$$f(x) = x^4$$

First Derivative = $df/dx = 4x^3 = 0$ then $x = 0$

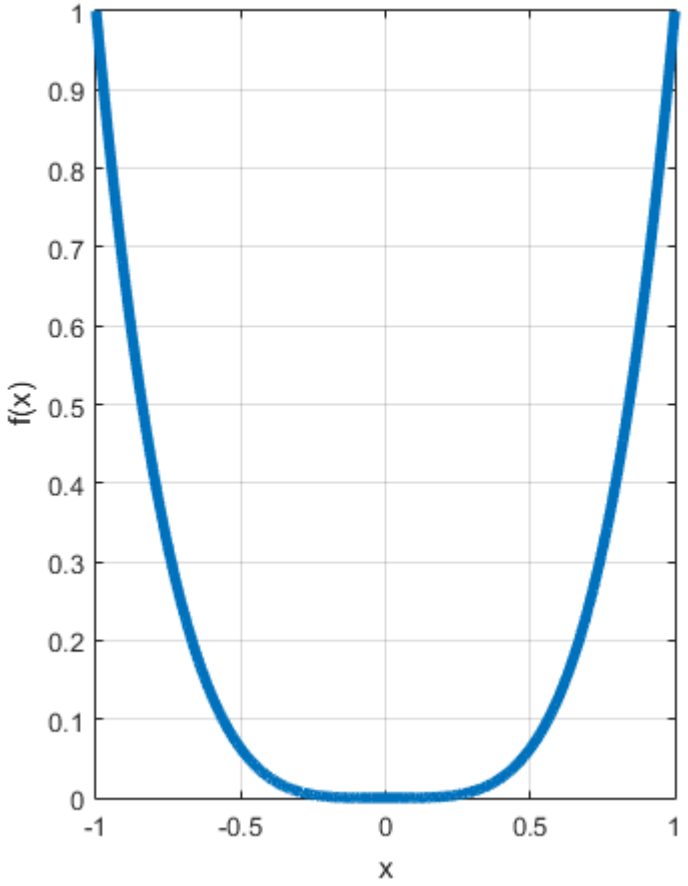
But

Second derivative = $12x^2 = 0$

Third derivative = $24x = 0$

Fourth derivative = $24 \neq 0$

(Even Derivative)



Sufficient condition (1D)

- Let $f'(x^*)=f''(x^*)=\ldots=f^{(n-1)}(x^*)=0$, but $f^{(n)}(x^*) \neq 0$. Then $f(x^*)$ is
  - A **minimum** value of $f(x)$ if $f^{(n)}(x^*) > 0$ and n is **even**
  - A **maximum** value of $f(x)$ if $f^{(n)}(x^*) < 0$ and n is **even**
  - Neither a minimum nor a maximum if n is **odd**

Multivariable optimization with no constraints

- **Necessary condition**

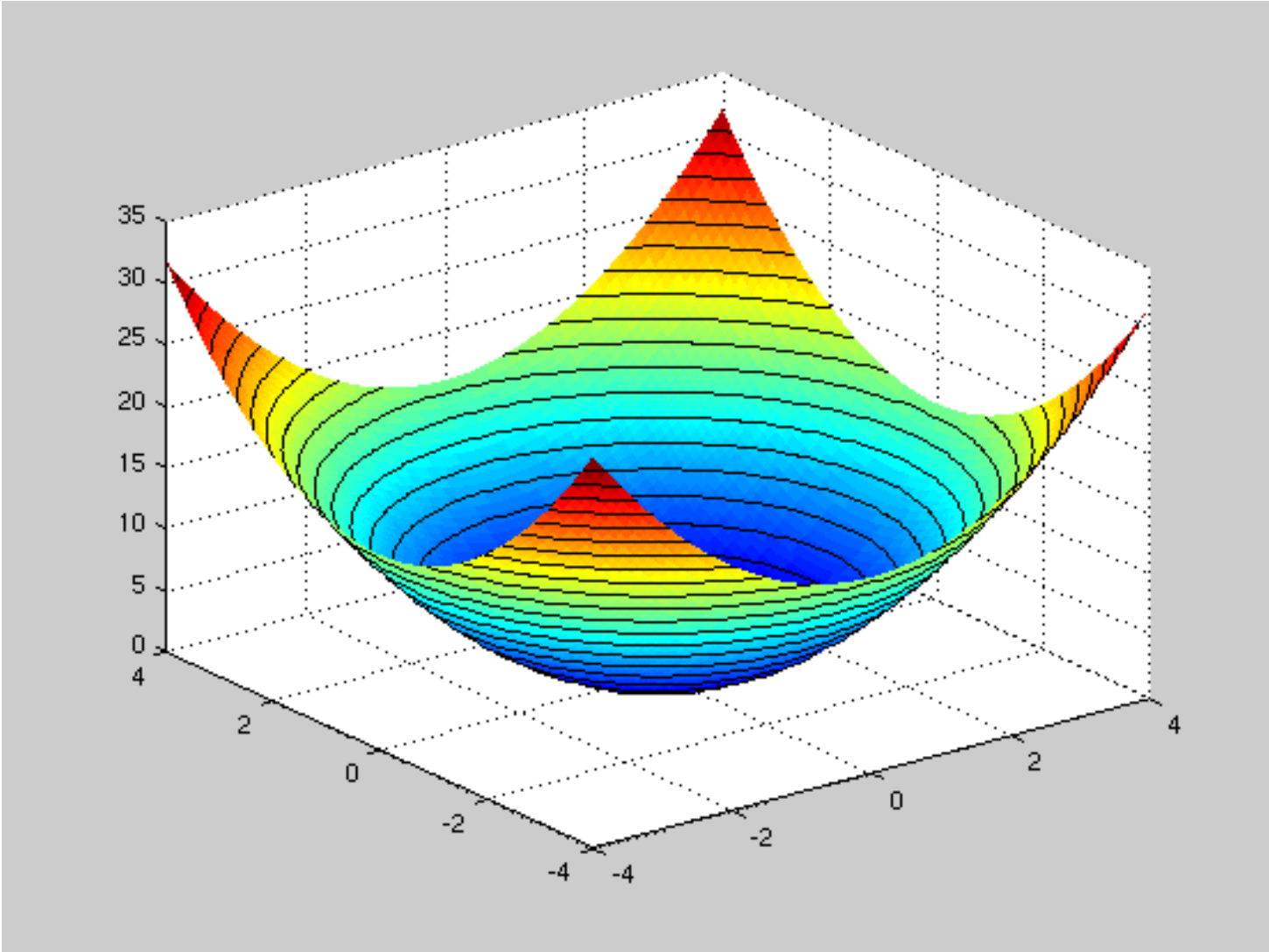
  If $f(\mathbf{X})$ has an extreme point (maximum or minimum) at $\mathbf{X}=\mathbf{X}^*$ and if the first partial derivatives of $f(\mathbf{X})$ exist at $\mathbf{X}^*$, then

$$\frac{\partial f}{\partial x_1}(\mathbf{X}^*) = \frac{\partial f}{\partial x_2}(\mathbf{X}^*) = \cdots = \frac{\partial f}{\partial x_n}(\mathbf{X}^*) = 0$$

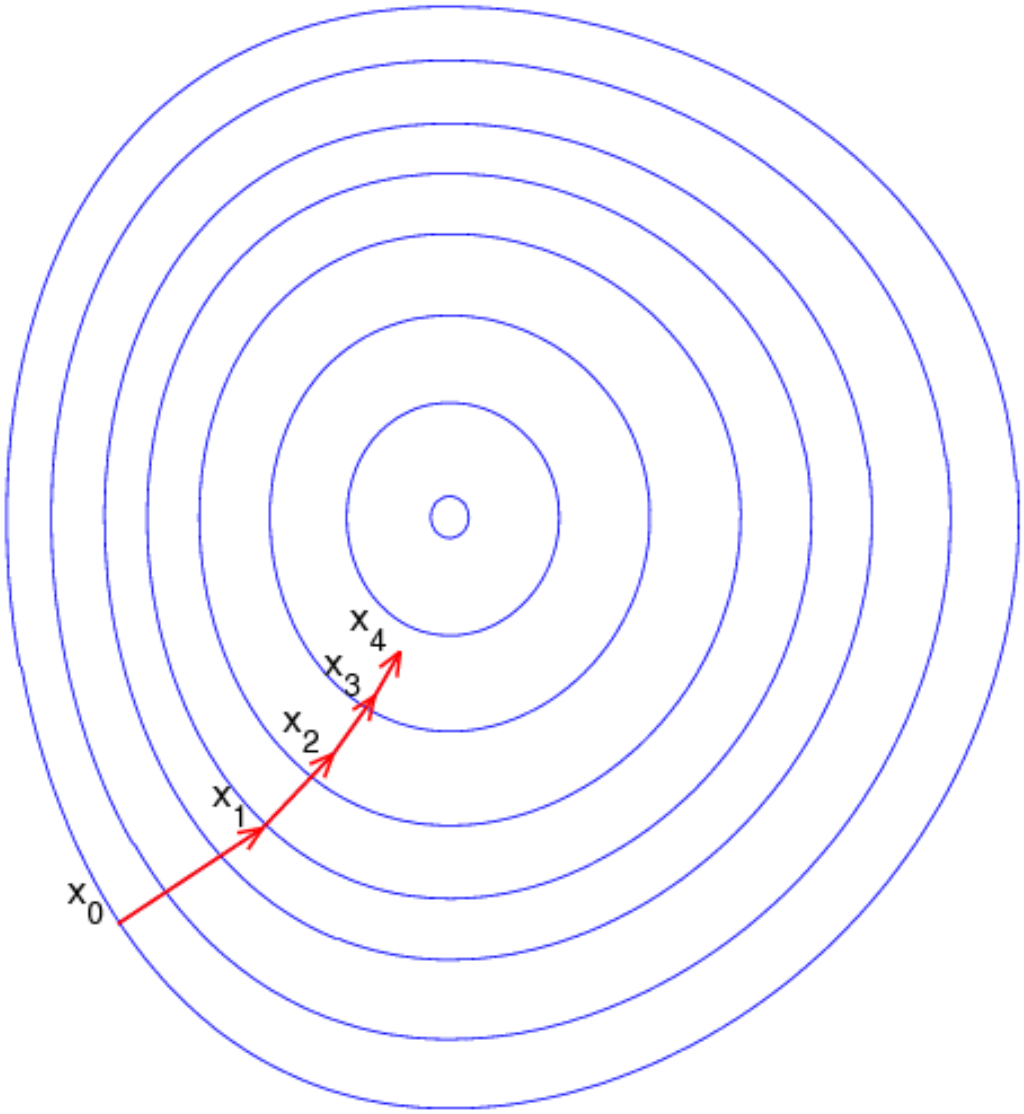
- **Sufficient condition**

  A sufficient condition for a stationary point $\mathbf{X}^*$ to be an extreme point is that the matrix of second partial derivatives (Hessian matrix) of $f(\mathbf{X}^*)$ evaluated at $\mathbf{X}^*$ is
  - **Positive definite** when $\mathbf{X}^*$ is a **relative minimum point**
  - **Negative definite** when $\mathbf{X}^*$ is a **relative maximum point**

Assume the following 2D Function



Contour plot



Gradient descent: head downhill

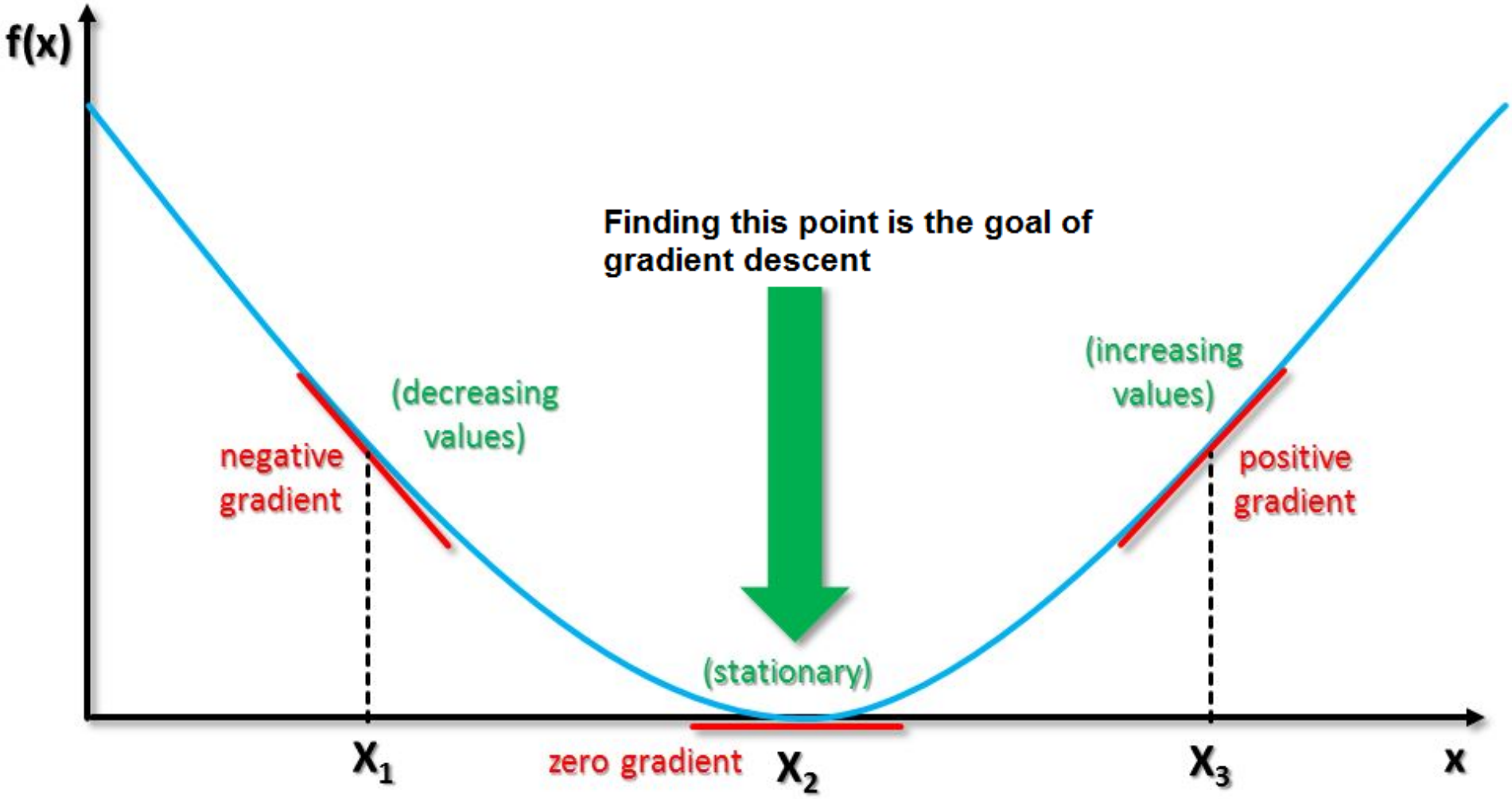
http://en.wikipedia.org/wiki/Gradient_descent

The Gradient Descent

**We select moving in the gradient direction such that:-

$$f(X_0) \geq f(X_1) \geq f(X_2) .... \geq f(X_{n-1}) \geq f(X_n)$$

The Gradient Descent Algorithm



The Gradient Descent Algorithm

Step 0: $Select\ X_0 \in R^n,\ Set\ \alpha, and\ i = 0$

Step 1: Compute $\nabla f(X_i)$

Step 2: if $||\nabla f(X_i)|| < \varepsilon, Stop$
Otherwise Go To Step 3

Step 3: Compute $X_{i+1} = X_i - \alpha \nabla f(X_i)$

Step 4: Update i=i+1

Step 5: Go To Step 1

Computing the Gradient

$$f : R^n \rightarrow R$$

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

Step Size Selection (α)

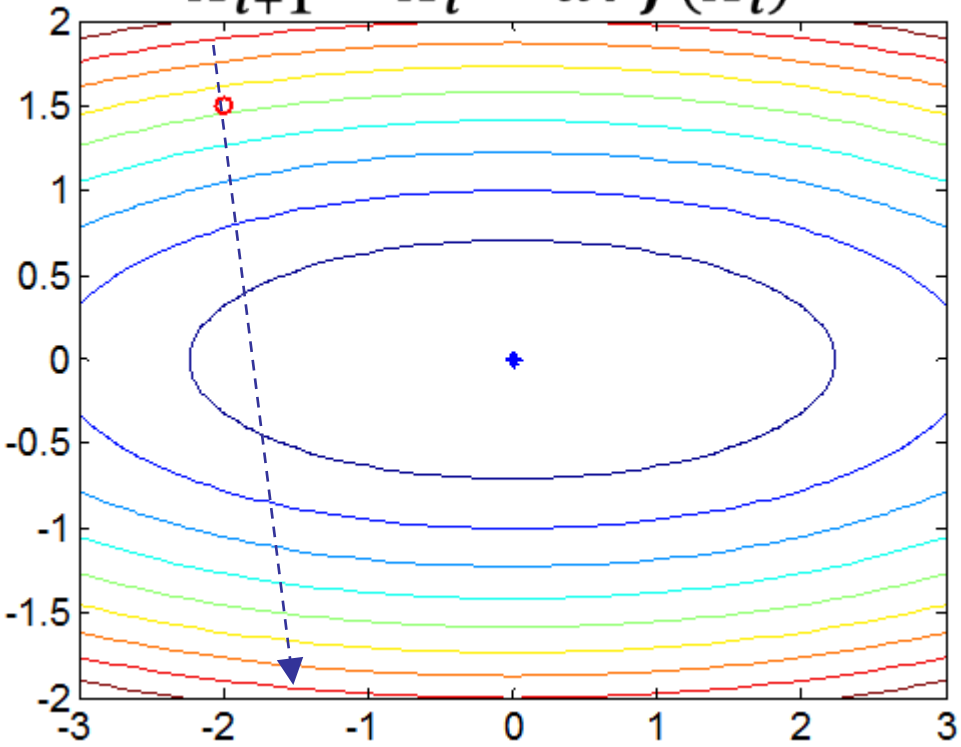
How should we select the step size?

- α too small: convergence takes long time
- α too large: overshoot minimum

Line minimization:

$$\alpha = \arg\min_{\alpha} f(X_i - \alpha \nabla f(X_i))$$

$$X_{i+1} = X_i - \alpha \nabla f(X_i)$$



The Steepest Gradient Descent Algorithm (Line Search)

Step 0: $Select\ X_0 \in R^n,\ Set\ \alpha, and\ i = 0$

Step 1: Compute $\nabla f(X_i)$

Step 2: if $||\nabla f(X_i)|| < \varepsilon, Stop$

Otherwise Go To Step 3

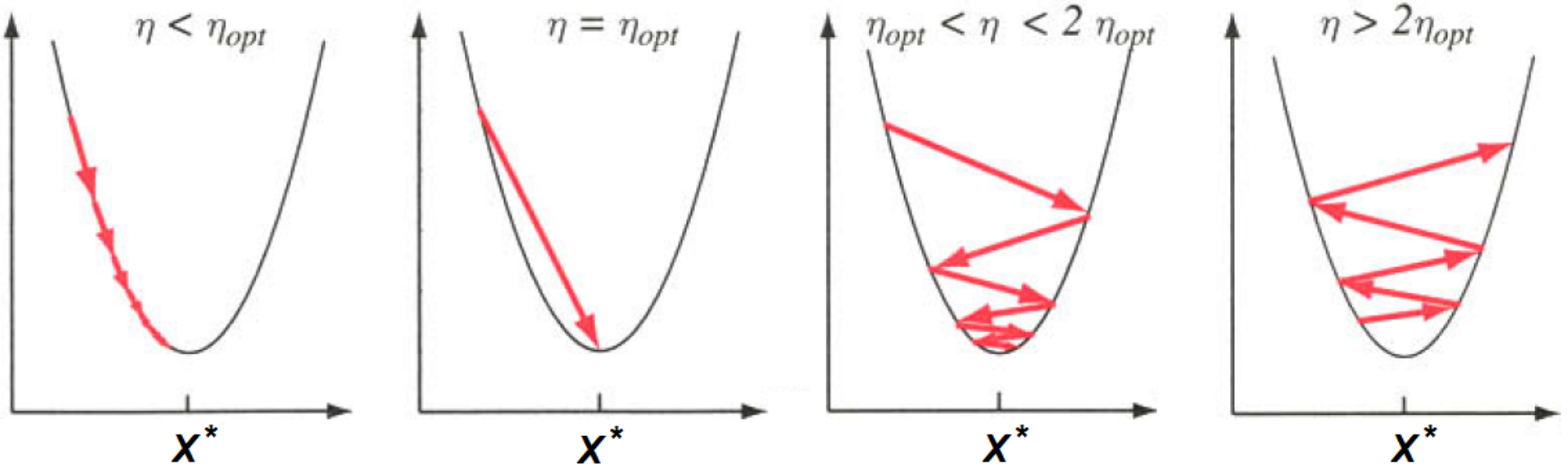
Step 3: Update $\alpha^* = \underset{\alpha}{\operatorname{argmin}} f(X_i - \alpha \nabla f(X_i))$

Step 4: Compute $X_{i+1} = X_i - \alpha^* \nabla f(X_i)$

Step 5: Update i=i+1

Step 6: Go To Step 1

The Steepest Gradient Descent Algorithm



Gradient descent in a one-dimensional quadratic criterion with different learning rates. If $\eta < \eta_{opt}$, convergence is assured, but training can be needlessly slow. If $\eta = \eta_{opt}$, a single learning step suffices to find the error minimum. If $\eta_{opt} < \eta < 2\eta_{opt}$, the system will oscillate but nevertheless converge, but training is needlessly slow. If $\eta > 2\eta_{opt}$, the system diverges.

Step Size Automatic Selection: The Newton-Raphson Algorithm

Step 0: $Select\ X_0 \in R^n,\ and\ i = 0$

Step 1: Compute $\nabla f(X_i)\ and\ H$

Step 2: if $||\nabla f(X_i)|| < \varepsilon, Stop$
Otherwise Go To Step 3

Step 3: Compute $\alpha = H^{-1}$

Step 4: Compute $X_{i+1} = X_i - \alpha \nabla f(X_i)$

Step 5: Update i=i+1

Step 6: Go To Step 1

Ex1: Gradient Descent

$\alpha = 0.1$

$f(x)=x^4 - x^3 + x^2 - x + 1$ $\quad\quad$ $df/dx = 4x^3 - 3x^3 + 2x - 1$

	Xn	f(Xn)	df/dx
1	1	1	2
2	0.8000	0.7376	0.7280
3	0.7272	0.6967	0.4062
4	0.6866	0.6834	0.2536
5	0.6612	0.6781	0.1672
6	0.6445	0.6757	0.1137
7	0.6331	0.6746	0.0789
8	0.6252	0.6741	0.0554
9	0.6197	0.6738	0.0393
10	0.6158	0.6737	0.0280
11	0.6130	0.6736	0.0200
12	0.6110	0.6736	0.0144
13	0.6095	0.6736	0.0103
14	0.6085	0.6736	0.0074
15	0.6078	0.6736	0.0054
16	0.6072	0.6736	0.0039
17	0.6068	0.6736	0.0028
18	0.6066	0.6736	0.0020
19	0.6064	0.6736	0.0015
20	0.6062	0.6736	0.0011
21	0.6061	0.6736	7.6266e-04

Ex2: Newton Raphson

$f(x)=x^4 -x^3 +x^2 -x+1$ $df/dx=4x^3 -3x^3 +2x -1$

$d^2f/dx^2=12x^2 -9x^2 +2$

X	f(X)	df/dx	d2f/dx2
10	9091	3719	1142
6.7434	1.8010e+03	1.1027e+03	507.2260
4.5695	357.8926	327.1530	225.1489
3.1165	71.6578	97.1690	99.8496
2.1433	14.7075	28.8891	44.2657
1.4907	3.3569	8.5650	19.7216
1.0564	1.1260	2.4805	9.0532
0.7824	0.7255	0.6441	4.6514
0.6439	0.6756	0.1119	3.1121
0.6080	0.6736	0.0059	2.7876
0.6058	0.6736	1.9371e-05	2.7694
0.6058	0.6736	2.0890e-10	2.7694
0.6058	0.6736	-1.1102e-16	2.7694
0.6058	0.6736	-1.1102e-16	2.7694
0.6058	0.6736	-1.1102e-16	2.7694

Ex3: Line Search

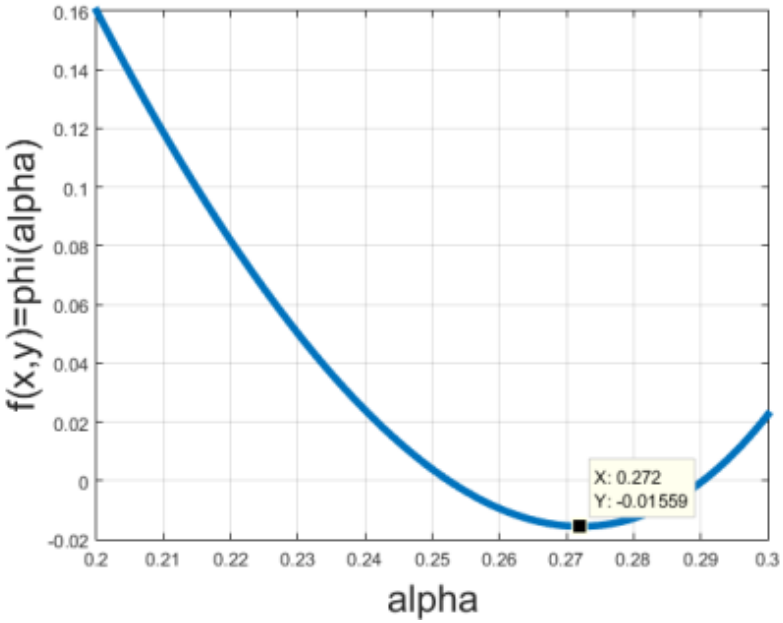
$X_0=(1,1)^T$

$f(x,y) = x^4+xy+y^2$ $\quad f_x = 4x^3+y$ $\quad f_y = x+2y$

Gradient at $X_0 = \begin{pmatrix}5\\3\end{pmatrix}$

New Position $X_1 = \begin{pmatrix}1\\1\end{pmatrix} - \alpha\begin{pmatrix}5\\3\end{pmatrix} = \begin{pmatrix}1-5\alpha\\1-3\alpha\end{pmatrix}$

$f(X_1) = (1-5\alpha)^4+(1-5\alpha)(1-3\alpha)+(1-3\alpha)^2$



Ex3: Line Search

$X_0=(1,1)^T$

$f(x,y) = x^4+xy+y^2$

$f_x = 4x^3+y$

$f_y = x+2y$

	fx	fy	α	x	y
1	5	3	0.2721	-0.3606	0.1836
2	-0.0040	0.0066	1.0032	-0.3566	0.1770
3	-0.0045	-0.0027	0.3955	-0.3549	0.1780
4	-7.2371e-04	0.0012	1.0128	-0.3541	0.1768
5	-8.3974e-04	-5.0392e-04	0.3972	-0.3538	0.1770
6	-1.3802e-04	2.3000e-04	1.0146	-0.3537	0.1768
7	-1.6110e-04	-9.6683e-05	0.3976	-0.3536	0.1768
8	-2.6553e-05	4.4238e-05	1.0151	-0.3536	0.1768
9	-3.1020e-05	-1.8622e-05	0.3976	-0.3536	0.1768
10	-5.1118e-06	8.5225e-06	1.0148	-0.3536	0.1768

Ex4: Newton Raphson

$X_0=(-2,-2)^T$

$f(x,y) = x^4+xy+y^2$

$f_x = 4x^3+y$

$f_y = x+2y$

x	y	f(x,y)	fx	fy	fxx	fxy	fyx	fyy
-1.3474	0.6737	2.8418	-9.1104	6.6613e-16	21.7848	1	1	2
-0.9193	0.4597	0.5031	-2.6484	-1.1102e-16	10.1424	1	1	2
-0.6447	0.3223	0.0688	-0.7494	0	4.9873	1	1	2
-0.4777	0.2388	-0.0050	-0.1971	0	2.7381	1	1	2
-0.3896	0.1948	-0.0149	-0.0417	0	1.8214	1	1	2
-0.3580	0.1790	-0.0156	-0.0045	0	1.5380	1	1	2
-0.3536	0.1768	-0.0156	-8.1783e-05	0	1.5007	1	1	2
-0.3536	0.1768	-0.0156	-2.8342e-08	0	1.5000	1	1	2
-0.3536	0.1768	-0.0156	-3.4139e-15	0	1.5000	1	1	2
-0.3536	0.1768	-0.0156	-2.7756e-17	0	1.5000	1	1	2