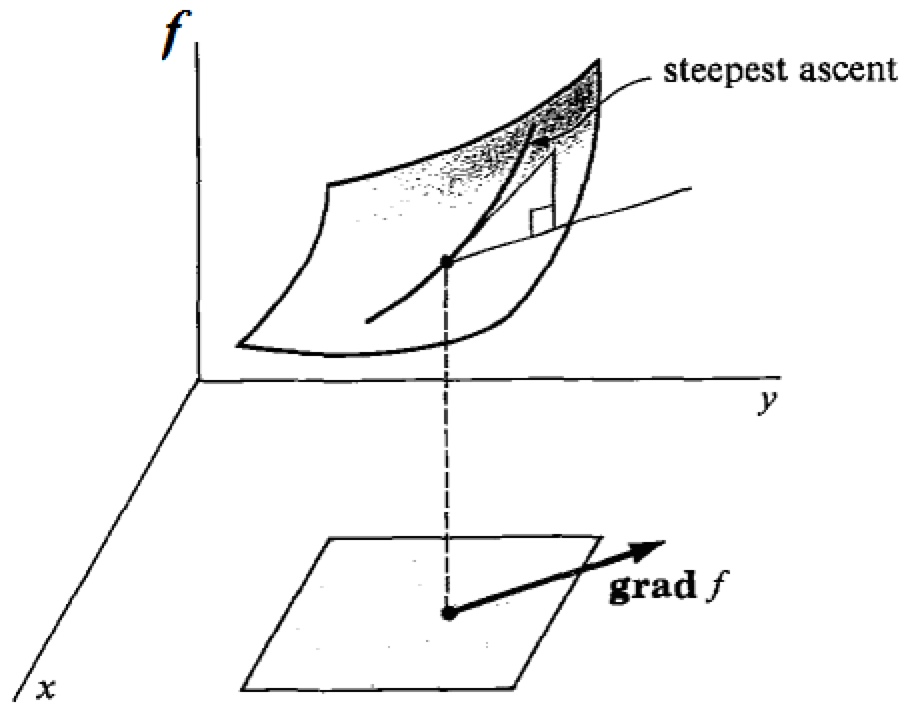
LOCAL SEARCH ALGORITHMS AND OPTIMIZATION PROBLEMS

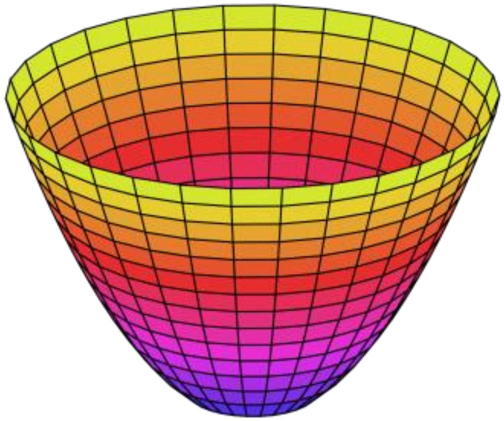
Gradient is a vector in the state space. (State space is continuous)

- **2D example**

$$grad \quad f = \nabla f = \begin{pmatrix} \partial f / \partial x \\ \partial f / \partial y \end{pmatrix}$$



An example: Gradient Descent



Let $X = \binom{x_1}{x_2} = (x_1 \ \ x_2)^T$.

Let $f(X) = (x_1 - 1)^2 + (x_2 - 2)^2 + 4$

$\nabla f(X) = \binom{2x_1 - 2}{2x_2 - 4}$

Let $X_0 = \binom{0}{0}$

Let the step size be $\eta = 0.1$

Then, $X_1 = X_0 - \eta \nabla f(X_0) = \binom{0}{0} - 0.1\binom{-2}{-4} = \binom{0.2}{0.4}$

$X_2 = X_1 - \eta \nabla f(X_1) = \binom{0.2}{0.4} - 0.1\binom{-1.6}{-3.2} = \binom{0.36}{0.72}$

Gradient Descent is from first order Taylor series approximation

Let us consider 1D case to begin with,

In one dimensional case $\nabla f = f'$

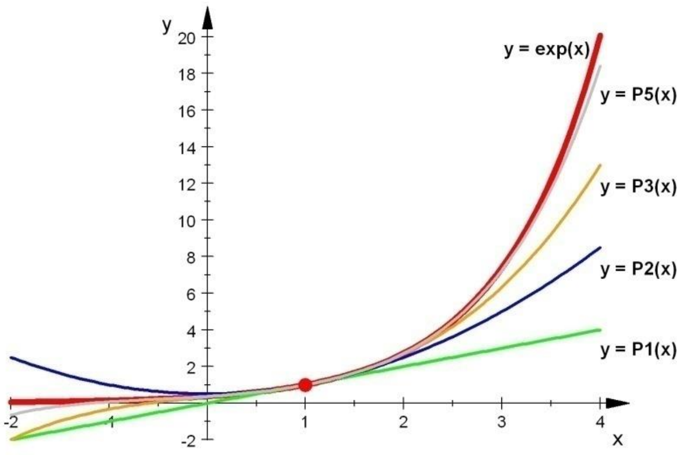
- $f(a_{k+1}) = f(a_k) + f'(a_k)(a_{k+1} - a_k) + \frac{f''(a_k)(a_{k+1} - a_k)^2}{2!} + \cdots$
- $\hat{f}(a_{k+1}) = f(a_k) + f'(a_k)(a_{k+1} - a_k)$

Taylor Series

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots.$$

In mathematics, a **Taylor series** is a representation of a function as an infinite sum of terms that are calculated from the values of the function's derivatives at a single point.

If the Taylor series is centered at zero, then that series is also called a **Maclaurin series**, named after the Scottish mathematician Colin Maclaurin, who made extensive use of this special case of Taylor series in the 18th century.



The Taylor series for the exponential function e^x at $a = 0$ is

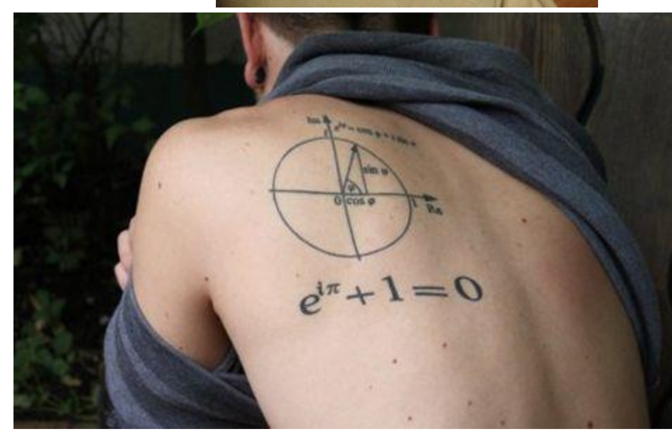
$$\frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \cdots = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \cdots = \sum_{n=0}^{\infty} \frac{x^n}{n!}.$$

Euler's great formula

$$\boxed{e^x \quad = \quad 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots}$$

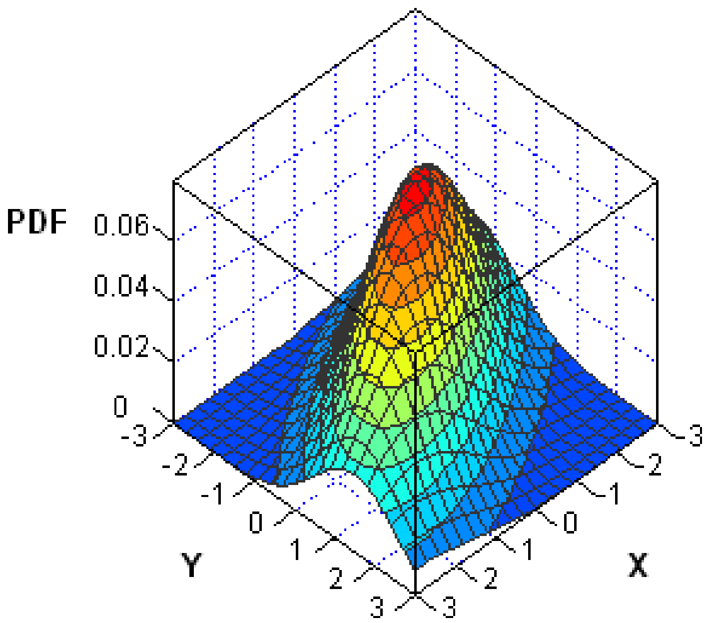
$$\begin{aligned}
e^{ix} &= \sum_{n=0}^{\infty} \frac{(ix)^n}{n!} \\
&= 1 + ix - \frac{(ix)^2}{2!} - \frac{(ix)^3}{3!} + \frac{(ix)^4}{4!} + \frac{(ix)^5}{5!} + \cdots \\
&= \left(1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots\right) + \left(ix - \frac{(ix)^3}{3!} + \frac{(ix)^5}{5!} + \cdots\right) \\
&= \left(1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots\right) + i\left(x - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots\right) \\
&= \cos x + i \sin x
\end{aligned}$$

$$\begin{aligned}
e^{i\pi} &= \cos \pi + i \sin \pi \\
&= -1
\end{aligned}$$



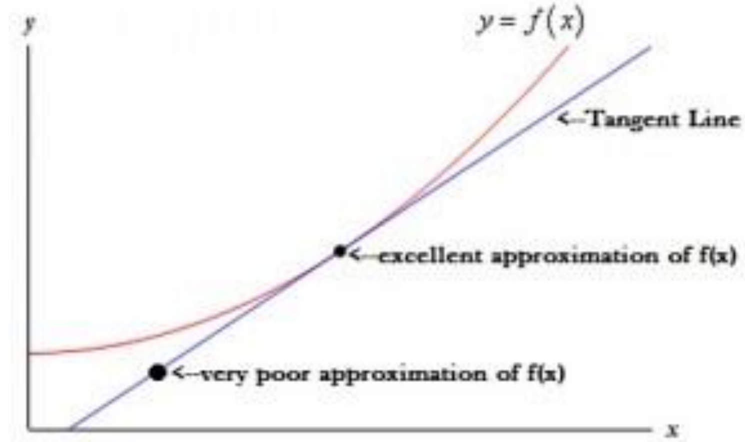
Multivariate

$$y = f(\mathbf{x} + \Delta\mathbf{x}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^{\mathrm{T}} \Delta\mathbf{x} + \frac{1}{2} \Delta\mathbf{x}^{\mathrm{T}} \mathbf{H}(\mathbf{x}) \Delta\mathbf{x}$$



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

Gradient Descent works with linear approximation



- $f(a_{k+1}) = f(a_k) + f'(a_k)(a_{k+1} - a_k) + \frac{f''(a_k)(a_{k+1} - a_k)^2}{2!} + \cdots$
- $\hat{f}(a_{k+1}) = f(a_k) + f'(a_k)(a_{k+1} - a_k)$

What is the best a_{k+1}

Considering upto 2nd order approximation

$$f(a_{k+1}) \approx f(a_k) + f'(a_k)(a_{k+1} - a_k) + \frac{f''(a_k)}{2}(a_{k+1} - a_k)^2$$

Differentiating w.r.t. a_{k+1}

$$f'(a_{k+1}) \approx 0 + f'(a_k) + \frac{f''(a_k)}{2} 2(a_{k+1} - a_k)$$

At best a_{k+1}, $f'(a_{k+1}) = 0$. So,

$$a_{k+1} \approx a_k - \frac{f'(a_k)}{f''(a_k)}$$

if f is quadratic then $a_{k+1} = a_k - \dfrac{f'(a_k)}{f''(a_k)}$

That is, in a single step we get the solution.

Gradient Descent Vs Newton's Descent

- $a_{k+1=}a_k - \frac{1}{f''(a_k)} f'(a_k)$

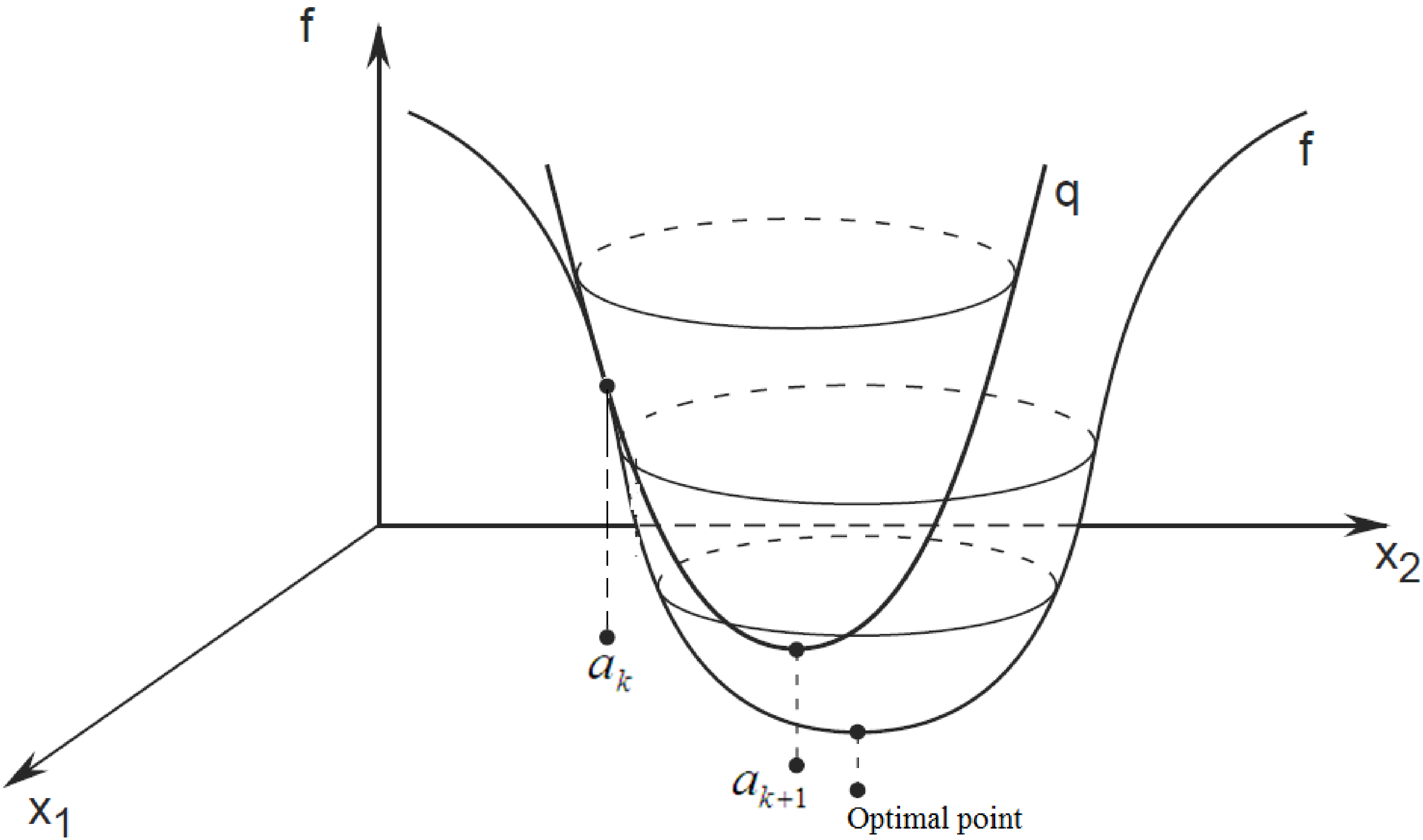
- Since second order information is not used,
  $a_{k+1=}a_k - \eta f'(a_k)$ is going to give a better solution by moving a small step in the negative of the gradient direction.

- Newton's method will say, if you can use the second order information then use $\frac{1}{f''(a_k)}$ instead of η

Newton's Descent

- $a_{k+1=}a_k - \frac{1}{f''(a_k)} f'(a_k)$
- For multivariate case $\frac{1}{f''(a_k)} = \mathrm{H}^{-1}$
- Where H is the Hessian matrix.
- Newton's method converges at a faster rate to local minima.
- If the objective is quadratic, then Newton's method gives solution in a single step.
  - Closed form solution ….

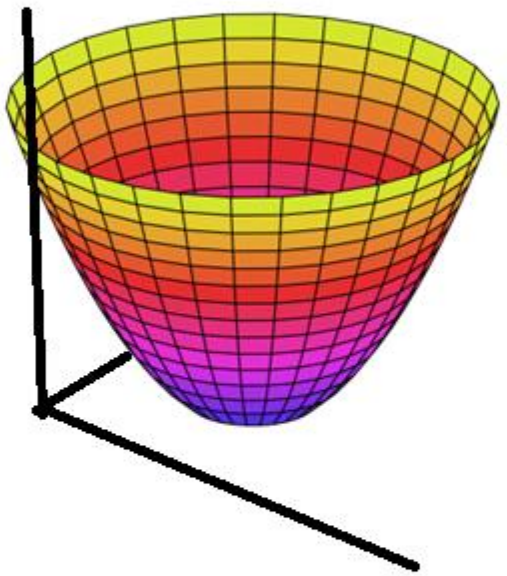
Newton's Descent



Can you apply the Newton's descent

Let $X = \binom{x_1}{x_2} = (x_1 \;\; x_2)^T$.

Let $f(X) = (x_1 - 1)^2 + (x_2 - 2)^2 + 4$

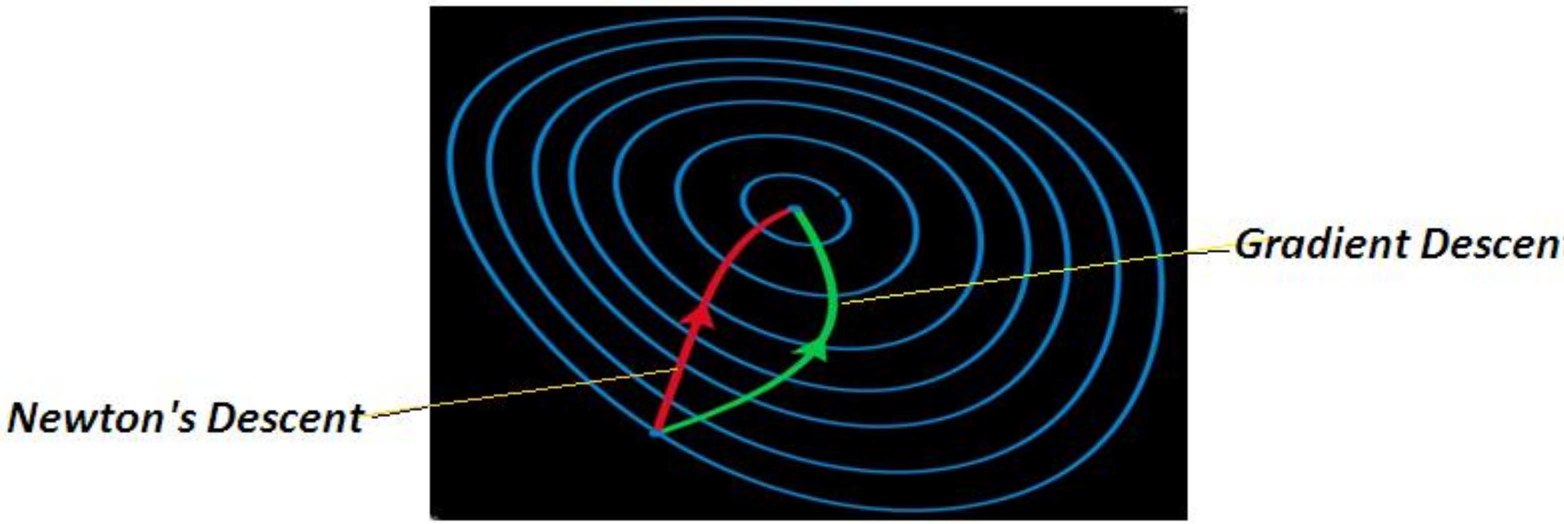


$\nabla f(X) = \binom{2x_1 - 2}{2x_2 - 4}$

Let $X_0 = \binom{0}{0}$

- You should get solution in a single step (why?)

Gradient Descent Vs Newton’s Descent



Contours of function values

Step size

Step size $\eta(k)$ should be carefully chosen.

- If $\eta(k)$ is too small then the convengence process will be needlessly slow. That is, number of iterations will be large.
- On the otherhand, if $\eta(k)$ is too large, the correction process will overshoot and can even diverge.

- There exist some systematic procedures which guide us in choosing the step size, at the given time.

An application of Gradient Descent

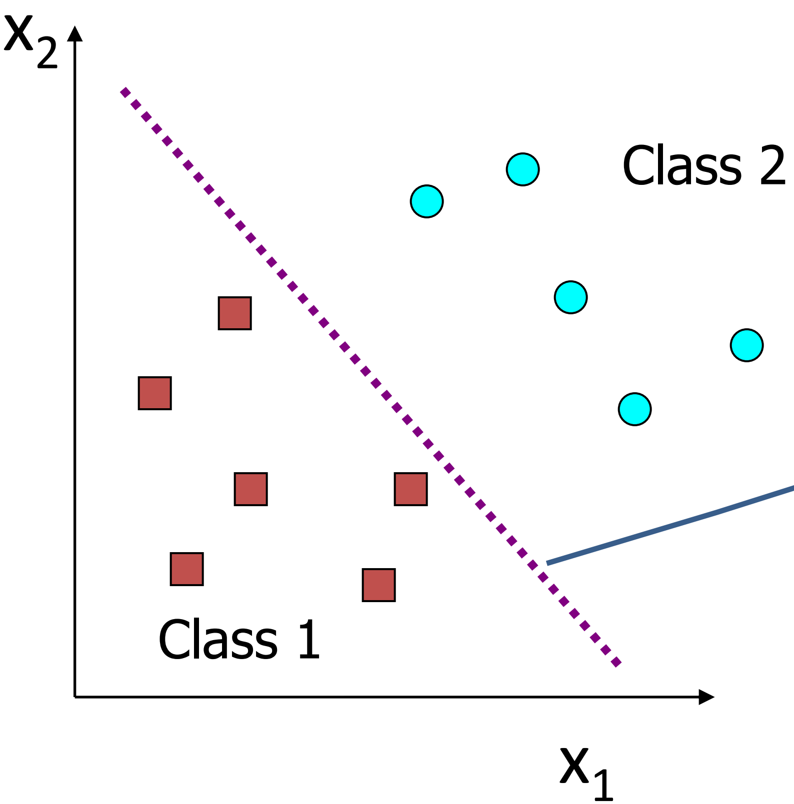
- We try to build a simple classifier called the **perceptron**.
- Recall what is the classification problem !

Linear Classifier

<u>Classifier:</u>

If $g(x_1,x_2) < 0$ assign Class 1;

If $g(x_1,x_2) > 0$ assign Class 2;



$$\boxed{g(x_1,x_2) = w_1x_1+w_2x_2+b = 0}$$

Perceptron

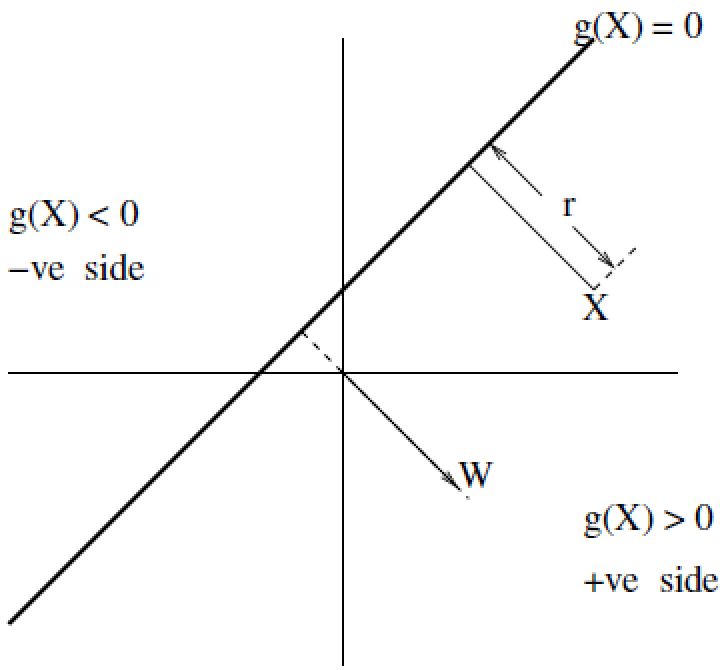
- Perceptron is the name given to the linear classifier with a threshold delimiter.
- If there exists a Perceptron that correctly classifies all training examples, then we say that the training set is linearly separable.
- In 1960s Rosenblatt gave an algorithm for Perceptron learning for linearly separable data.

In general, the linear discriminant

- Consider a two class problem, $\Omega = \{\omega_1, \omega_2\}$
  A pattern $X = (x_1, \ldots, x_d)^t$
- The discriminant function
  $g(X) = w_1 x_1 + \cdots + w_d x_d + w_0 \; = \; W^t X + w_0$
- $g(X) = 0$ defines a hyperplane in the feature space.
- Classification rule:

$$
\begin{aligned}
g(X) > 0 \quad &\Rightarrow \quad \text{class is } \omega_1 \\
g(X) < 0 \quad &\Rightarrow \quad \text{class is } \omega_2 \\
g(X) = 0 \quad &\Rightarrow \quad \text{class is decided arbitrarily}
\end{aligned}
$$

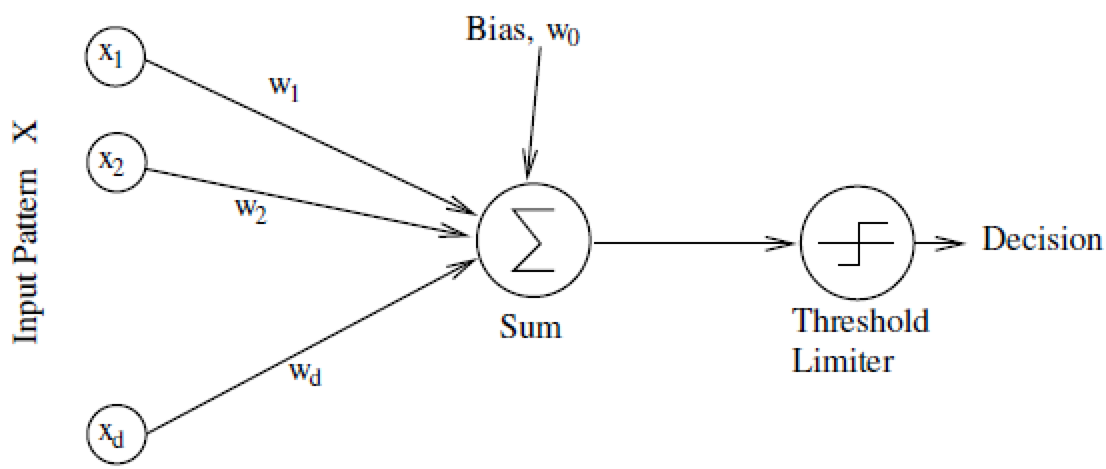
Perceptron is a linear discriminant



- $r = g(X)/||W||$ is the perpendicular distance of a point X with the hyperplane.

Perceptron: some historical remarks ...

- Its inventor is Frank Rosenblatt (1958), a psychologist.
- It is one of the early artificial neural network models.



•This is that which is conventionally known as the Perceptron.
•See the difference, *g(X)* as defined in linear discriminant does not use the threshold limiter.
•Actually when it comes to learning the weights, we ignore the threshold!

Perceptron: some historical remarks ...

- Rosenblatt gave an algorithm which can be implemented with a machine.
- He gave a convergence theorem to establish certain important properties of his method.
- It raised lot of interest among many researchers.

Let us first simplify the problem

Augmented Feature Space

- X-Space: $X = (x_1, \ldots, x_d)^t$, $W = (w_1, \ldots, w_d)^t$ and $g(X) = w_0 + W \cdot X$
  - $g(X) = 0$ may not pass through the origin.
- Y-Space: $Y = (1, x_1, \ldots, x_d)^t$, $a = (w_0, w_1, \ldots, w_d)^t$ and $\hat{g}(Y) = a \cdot Y$
  - $\hat{g}(Y) = 0$ passes through the origin.
  - That is, $\hat{g}(Y) = 0$ is in homogeneous form.
  - This Y-Space is called as *augmented space* and Y is called as *augmented vector* of X.
- It is easy to work with the augmented space.
- We want to find the vector a which is called *separating vector* or more generally *solution vector*. It need not be unique.

Further simplifying…

Two category case

- Augmented feature space is used.

$$g(Y) = a^t Y = \begin{cases} > 0 & \text{for } Y \in \omega_1 \\ < 0 & \text{for } Y \in \omega_2 \\ = 0 & \text{we do not consider this.} \end{cases}$$

- Normalization
  - For all $Y_i \in \omega_2$ replace Y_i by $-Y_i$
  - Then $a^t Y > 0$ for all patterns irrespective of their class.
- If a linear discriminant functin can correctly classify the given dataset, then the dataset is called *linearly separable*.

How to find a ?

- We should find a solution to the set of linear inequalities $a^t Y_i > 0$ for all i.
- A easy way is to define a criterion function $J(a)$ that is minimized if a is a solution vector.
- Directly solving $\nabla J(a) = 0$ may not be always possible.
- An iterative method for finding a solution is to apply *gradient descent (Hill climbing)* methods.
- Gradient descent procedures are popular in many engineering applications.

Gradient Descent Procedures

- Basic gradient descent is very simple.
- We call the a in i th iteration as $a(i)$.
- We start with some arbitrarily chosen weight vector $a(1)$ and compute the gradient vector $\nabla J(a(1))$.
- The next value $a(2)$ is obtained by moving some distance from $a(1)$ in the direction of steepest descent, i.e., along the negative of the gradient.

Perceptron: Gradient descent

- Perceptron criterion:

$$J_p(a) = \sum_{Y \in \mathcal{Y}} -a^t Y$$

where $\mathcal{Y}$ is the set of misclassified patterns by the discriminant defined by a.

$$\nabla J_p(a) = \sum_{Y \in \mathcal{Y}} -Y$$

- Hence the update rule is, $a_{k+1} = a_k + \eta_k \sum_{Y \in \mathcal{Y}_k} Y$ where $\mathcal{Y}_k$ is the set of misclassified patterns by a_k.

Batch Perceptron

- $$a_{k+1} = a_k + \eta_k \sum_{Y \in \mathcal{Y}_k} Y$$

  where $\mathcal{Y}_k$ is the set of misclassified patterns by a_k.
- Normally, η_k is taken as 1. This is called *fixed increment* rule.
- For linearly separable datasets, it is proved that the learning converges to a solution.

Perceptron: Single Sample Correction

- This is a variant of the Batch Perceptron.
- Start with a arbitrary a_0.
- Whenever a pattern Y is misclassified, i.e., $a_k^t Y < 0$ then $a_{k+1} = a_k + Y$.
- The above step needs to be done repeatedly (training set needs to be scanned again and again) until all the training patterns are correctly classified.
- These kind of learning procedures are called *error correcting procedures* because a is updated only when error occurs.

Perceptron: Single Sample Correction

- It is easy to see geometrically what is happening.
- If Y is misclassified by a_k then $a_k^t Y < 0$.
- $a_{k+1} = a_k + Y \;\Rightarrow\; a_{k+1}^t Y = a_k^t Y + ||Y||^2$.
- Hence, the correction is to move the weight vector in a good direction.

Example: Single sample correction

Given Data

X	Y	Class
1	2	+1
2	3	+1
3	2	-1

Augmented Data

	X	Y	Class
1	1	2	+1
1	2	3	+1
1	3	2	-1

Normalized augmented data

1	1	2
1	2	3
-1	-3	-2

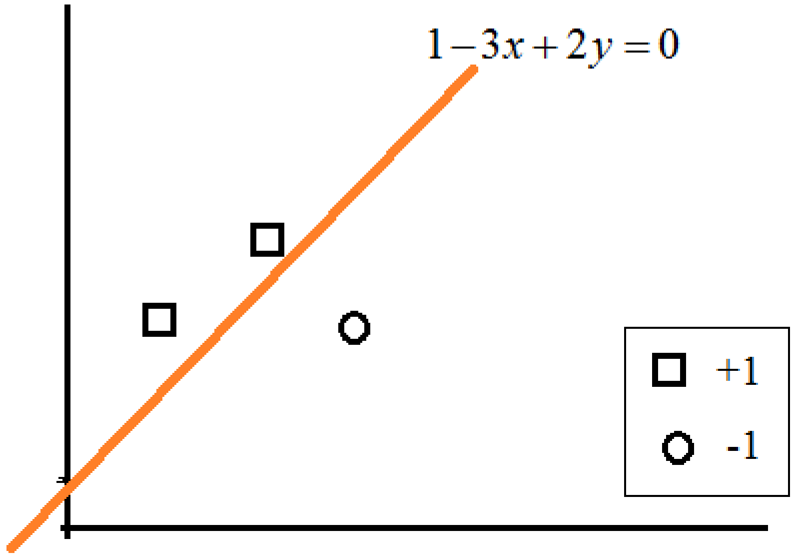
1	1	2
1	2	3
-1	-3	-2

- Initial solution (0,0,0).
- 1st misclassified, so next sol. = (1,1,2)
- 3rd … ,so next sol. = (0,-2,0)
- 1st = (1,-1,2)
- 3rd … = (0,-4,0)
- 1st … = (1,-3,2) <---seems okay

X	Y	Class
1	2	+1
2	3	+1
3	2	-1

- (1,-3,2) <--- seems okay

$$1 - 3x + 2y = 0 \ \textit{is the solution}$$



Name confusion …

- Whatever we saw, we call it the Perceptron (Rosenblatt).
  - We specify the batch method saying the Perceptron (Rosenblatt) Batch Method, and
  - the single sample method, we call, the Perceptron (Rosenblatt) Single Sample Correction Method.
- Note, these methods work only with linearly separable data.
  - With linearly not separable data, these may not even converge!!

The Perceptron (General)

- The Perceptron which works for any data, whether linearly separable or not, is called the Perceptron (general).
- This defines a criterion which is proportionate to the sum of squared errors with the training data, and
  - minimizes this error, to find the classifier.
  - this is guaranteed to converge, irrespective of the data being linearly separable or not.

THE PERCEPTRON (GENERAL)

(WORKS EVEN FOR LINEARLY NOT SEPARABLE DATA)

Labels

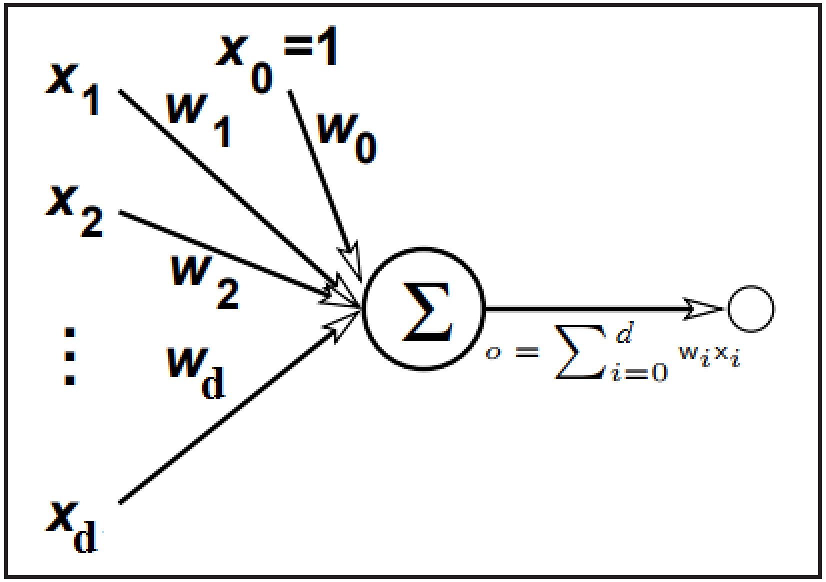
- Let us call the positive class label is $+1$,
- and the negative class label is -1.

- In the training set each example, say X is given with a target label. Let us call this t.
- The classifier outputs its decision. Let us call this o.

Error

- Error in classifying the pattern X is $\frac{1}{2}(t-o)^2$
- Why $\frac{1}{2}$ is there?
- Why we are doing squaring?

Direct attempt, in learning the linear discriminant



$$o(X) = w_0 + w_1 x_1 + \cdots + w_d x_d$$

Let's learn W that minimize the ***squared error***

$$E(W) = \frac{1}{2} \sum_{X_j \in D} (t_j - o_j)^2$$

where D is set of training examples.

The notation used,

$$X = (1, x_1, x_2, \dots, x_d)^t$$

$$W = (w_0, w_1, w_2, \dots, w_d)^t$$

$D = \{X_1, X_2, \dots, X_n\}$ is the training set.

Training Procedure

$$\nabla_W(E) = \nabla_W\left(\frac{1}{2}\sum_{j=1}^{n}(t_j - W\cdot X_j)^2\right) = \sum_{j=1}^{n}(t_j - W\cdot X_j)(-X_j)$$

$$W_{new} = W + \eta\sum_{j=1}^{n}(t_j - W\cdot X_j)(X_j)$$

Single sample or stochastic correction is

$$W_{new} = W + \eta(t_j - W\cdot X_j)(X_j)$$

Stop when the gradient, *i.e.*, $\nabla_W(E)$ is sufficiently small.

Convergence

[Hertz et al., 1991]

The gradient descent training rule used by the linear unit is guaranteed to converge to a hypothesis with minimum squared error

- given a sufficiently small learning rate η
- even when the training data contains noise
- even when the training data is not linearly seperable (it finds least error linear seperator).

Note: If η is too large, the gradient descent search runs the risk of overstepping the minimum in the error surface rather than settling into it. For this reason, one common modification of the algorithm is to gradually reduce the value of η as the number of gradient descent steps grows.

Closed Form Solution

- Since the criterion (Sum of squared errors) is quadratic, we can solve $\nabla_W E(W) = 0$ directly to get the solution.
- This is nothing but employing the Newton's descent.

Notation

- Let $X_i = (1, x_{i1}, x_{i2}, \dots, x_{id})^t$
- Let the data matrix be $D = \begin{bmatrix} 1\ x_{11}\ x_{12} \dots x_{1d} \\ 1\ x_{21}\ x_{22} \dots x_{2d} \\ \vdots \\ 1\ x_{n1}\ x_{n2} \dots x_{nd} \end{bmatrix}$
- Let $W = (w_0\ w_1\ w_2 \dots w_d)^t$
- Let the target vector be $\mathrm{T} = (t_1 t_2 \dots t_n)^t$

Closed form solution

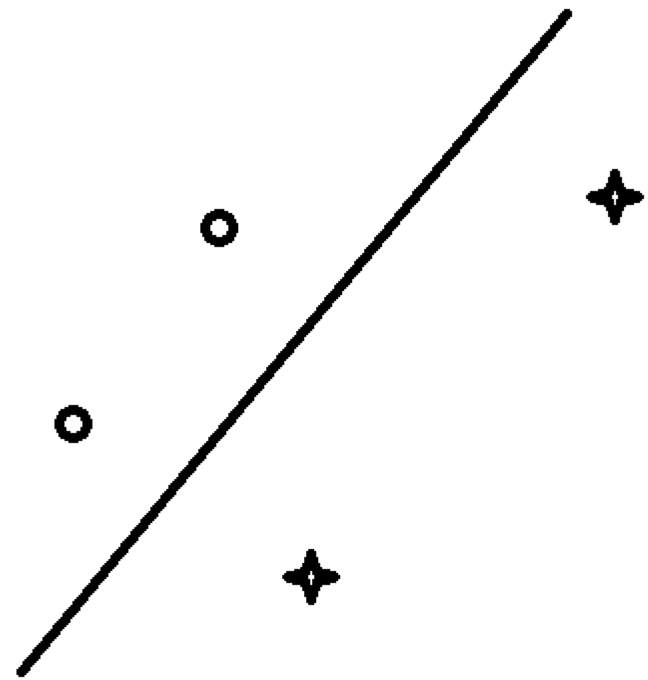
- Then, the error $E(W) = \frac{1}{2}\|T - DW\|^2$

- $\nabla_W E(W) = -D^t T + D^t D W$
- Equating the gradient to zero, we get,

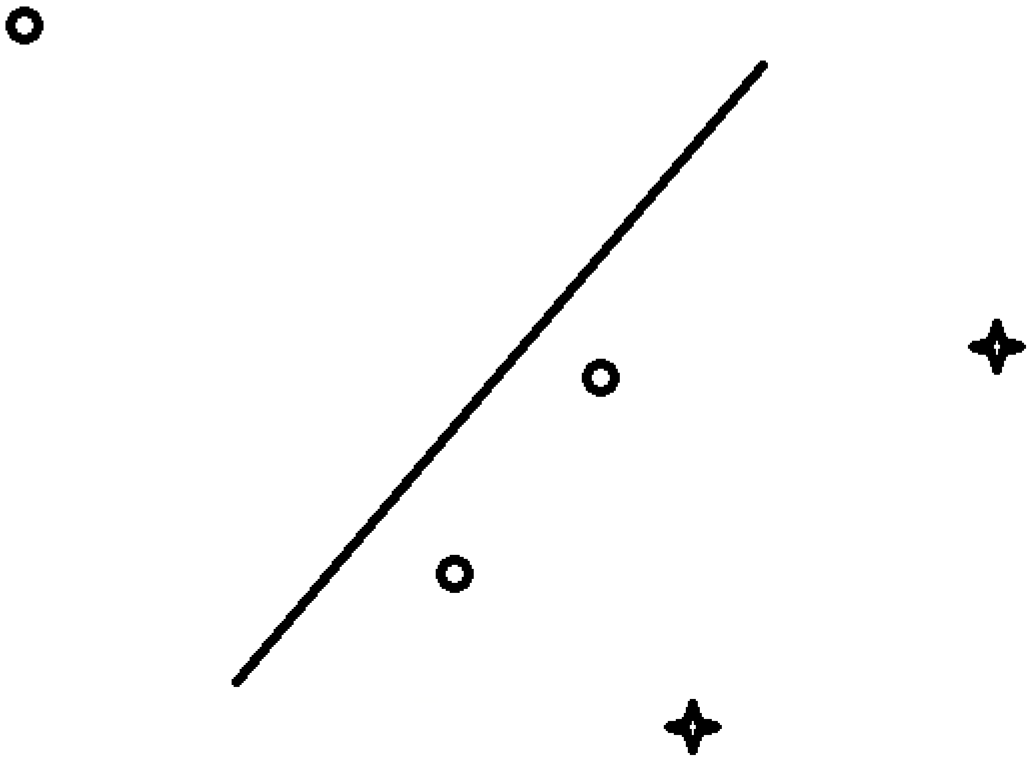
- $W = (D^t D)^{-1} D^t T$

Big drawback…

- We expect like



But with outliers, we may end like



- Have you noticed, in the previous slide the data is indeed linearly separable.
- Even then, Perceptron (general) can get like that.
  – The error because of the outlier is overshadowing all other errors…