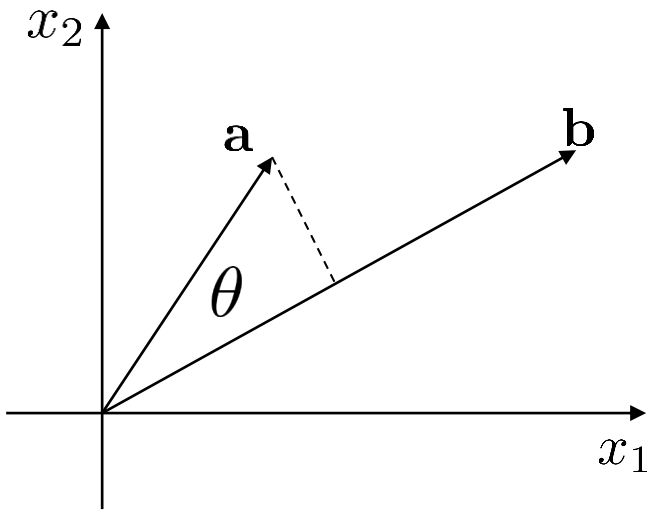
COGS 181, Fall 2017

Neural Networks and Deep Learning

Lecture 3: Optimization and estimation

Vector: Projection (inner product)

(one of the most important concepts in machine learning)



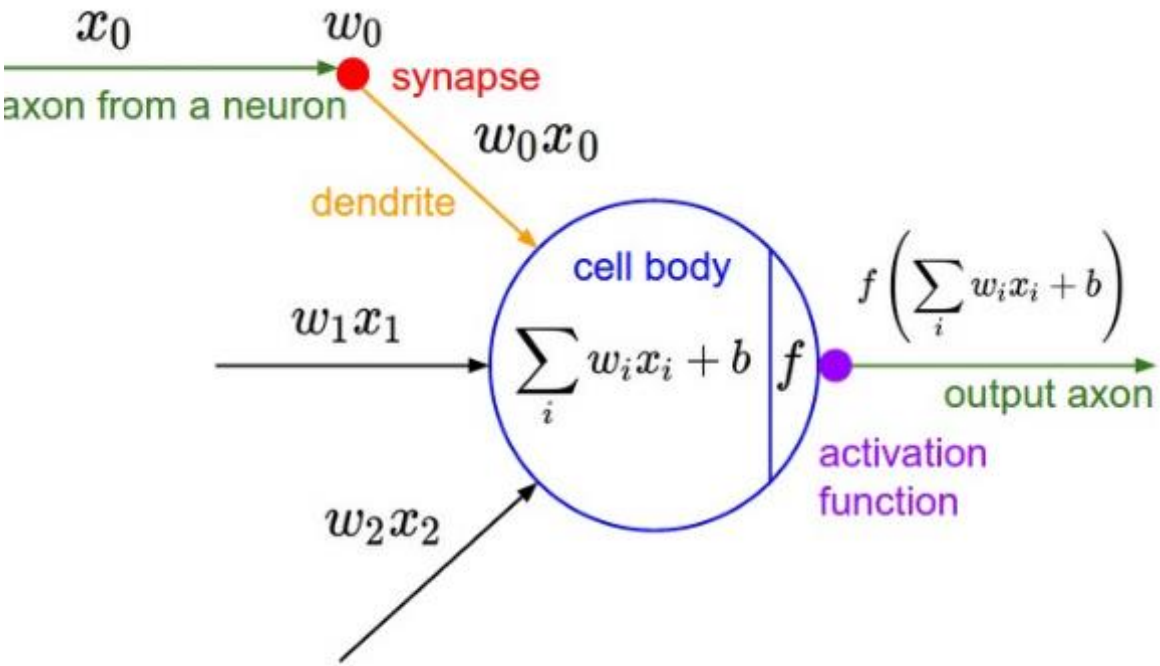
$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$< \mathbf{a}, \mathbf{b} > \equiv \mathbf{a} \cdot \mathbf{b} \equiv \mathbf{a}^T \mathbf{b} \equiv a_1 b_1 + a_2 b_2 + a_3 b_3$$

It's a scalar!

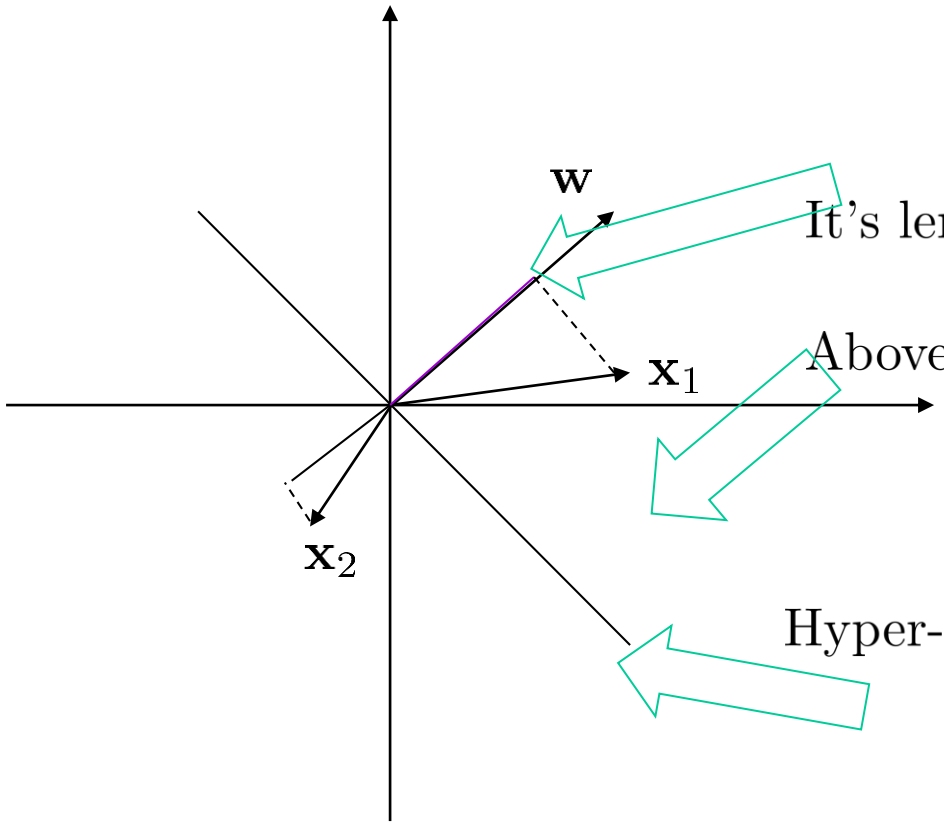
$$cos(\theta) = \frac{<\mathbf{a},\mathbf{b}>}{||\mathbf{a}||_2 \times ||\mathbf{b}||_2}$$

Perceptron


x_0
w_0
axon from a neuron
synapse
$w_0 x_0$
dendrite
cell body
$w_1 x_1$
$\sum_i w_i x_i + b$
f
$f\left(\sum_i w_i x_i + b\right)$
output axon
activation function
$w_2 x_2$

Orthogonal

$||\mathbf{w}||_2 = 1$: unit vector



It's length: $| < \mathbf{w}, \mathbf{x}_1 > |$

Above or below: $sign(< \mathbf{w}, \mathbf{x}_1 >)$

Hyper-plane: $\{\mathbf{x} :< \mathbf{w}, \mathbf{x} >= 0\}$

Calculus

Vector:

Vector-by-scalar

$$Y(x) = \begin{pmatrix} y_1(x) & y_2(x) & y_3(x) \end{pmatrix}$$

$$\frac{dY(x)}{dx} = \begin{pmatrix} \frac{dy_1(x)}{dx} & \frac{dy_2(x)}{dx} & \frac{dy_3(x)}{dx} \end{pmatrix}$$

Vector-by-vector

$$Y(X) = \begin{pmatrix} y_1(X) & ..., & y_m(X) \end{pmatrix} \qquad X = \begin{pmatrix} x_1 & ..., & x_n \end{pmatrix}$$

$$\frac{dY(X)}{dX} = \begin{pmatrix} \frac{dy_1(X)}{\partial x_1} & ..., & \frac{dy_m(X)}{\partial x_1} \\ . & . & . \\ \frac{dy_1(X)}{\partial x_n} & ..., & \frac{dy_m(X)}{\partial x_n} \end{pmatrix}$$

Calculus

vector-by-vector

$$A = \begin{pmatrix} a_{11} & , \ldots, & a_{1m} \\ . & . & . \\ a_{n1} & , \ldots, & a_{nm} \end{pmatrix} \qquad X = \begin{pmatrix} x_1 \\ . \\ x_m \end{pmatrix}$$

$$\frac{\partial AX}{\partial X} = A^T \text{: denominator layout}$$

$$\frac{\partial X^T A^T}{\partial X} = A^T \text{: denominator layout}$$

Vector calculus

Identities: vector-by-vector $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$

Condition	Expression	Numerator layout, i.e. by y and $\mathbf{x}^\mathsf{T}$	Denominator layout, i.e. by $\mathbf{y}^\mathsf{T}$ and x
a is not a function of **x**	$\frac{\partial \mathbf{a}}{\partial \mathbf{x}} =$	$\mathbf{0}$	
	$\frac{\partial \mathbf{x}}{\partial \mathbf{x}} =$	$\mathbf{I}$	
A is not a function of **x**	$\frac{\partial \mathbf{A}\mathbf{x}}{\partial \mathbf{x}} =$	$\mathbf{A}$	$\mathbf{A}^\top$
A is not a function of **x**	$\frac{\partial \mathbf{x}^\top \mathbf{A}}{\partial \mathbf{x}} =$	$\mathbf{A}^\top$	$\mathbf{A}$
a is not a function of **x**, **u** = **u**(**x**)	$\frac{\partial a\mathbf{u}}{\partial \mathbf{x}} =$	$a\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$	
a = *a*(**x**), **u** = **u**(**x**)	$\frac{\partial a\mathbf{u}}{\partial \mathbf{x}} =$	$a\frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \mathbf{u}\frac{\partial a}{\partial \mathbf{x}}$	$a\frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \frac{\partial a}{\partial \mathbf{x}}\mathbf{u}^\top$
A is not a function of **x**, **u** = **u**(**x**)	$\frac{\partial \mathbf{A}\mathbf{u}}{\partial \mathbf{x}} =$	$\mathbf{A}\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{u}}{\partial \mathbf{x}}\mathbf{A}^\top$
u = **u**(**x**), **v** = **v**(**x**)	$\frac{\partial (\mathbf{u}+\mathbf{v})}{\partial \mathbf{x}} =$	$\frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \frac{\partial \mathbf{v}}{\partial \mathbf{x}}$	
u = **u**(**x**)	$\frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{x}} =$	$\frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{u}}{\partial \mathbf{x}}\frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{u}}$

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

Calculus

Scalar-by-vector

A is asymmetric

$$A = \begin{pmatrix} a_{11} & , \ldots, & a_{1m} \\ . & . & . \\ a_{m1} & , \ldots, & a_{mm} \end{pmatrix} \qquad X = \begin{pmatrix} x_1 \\ . \\ x_m \end{pmatrix}$$

$$\frac{\partial X^T A X}{\partial X} = (A + A^T)X \text{: denominator layout}$$

A is symmetric

$$A = \begin{pmatrix} a_{11} & , \ldots, & a_{1m} \\ . & . & . \\ a_{1m} & , \ldots, & a_{mm} \end{pmatrix} \qquad X = \begin{pmatrix} x_1 \\ . \\ x_m \end{pmatrix}$$

$$\frac{\partial X^T A X}{\partial X} = 2AX \text{: denominator layout}$$

Matrix calculus

a is not a function of **x**	$\frac{\partial(\mathbf{a}\cdot\mathbf{x})}{\partial\mathbf{x}} = \frac{\partial(\mathbf{x}\cdot\mathbf{a})}{\partial\mathbf{x}} =$ $\frac{\partial\mathbf{a}^\top\mathbf{x}}{\partial\mathbf{x}} = \frac{\partial\mathbf{x}^\top\mathbf{a}}{\partial\mathbf{x}} =$	$\mathbf{a}^\top$	$\mathbf{a}$
A is not a function of **x** **b** is not a function of **x**	$\frac{\partial\mathbf{b}^\top\mathbf{A}\mathbf{x}}{\partial\mathbf{x}} =$	$\mathbf{b}^\top\mathbf{A}$	$\mathbf{A}^\top\mathbf{b}$
A is not a function of **x**	$\frac{\partial\mathbf{x}^\top\mathbf{A}\mathbf{x}}{\partial\mathbf{x}} =$	$\mathbf{x}^\top(\mathbf{A}+\mathbf{A}^\top)$	$(\mathbf{A}+\mathbf{A}^\top)\mathbf{x}$
A is not a function of **x** **A** is symmetric	$\frac{\partial\mathbf{x}^\top\mathbf{A}\mathbf{x}}{\partial\mathbf{x}} =$	$2\mathbf{x}^\top\mathbf{A}$	$2\mathbf{A}\mathbf{x}$
A is not a function of **x**	$\frac{\partial^2\mathbf{x}^\top\mathbf{A}\mathbf{x}}{\partial\mathbf{x}^2} =$	$\mathbf{A}+\mathbf{A}^\top$	
A is not a function of **x** **A** is symmetric	$\frac{\partial^2\mathbf{x}^\top\mathbf{A}\mathbf{x}}{\partial\mathbf{x}^2} =$	$2\mathbf{A}$	

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

Three key things you are learning in this class:

Representation: With better and better understanding of the underlining statistics about the data and methods.

Evaluation: The ideal strategy is always to aim at your target directly (take non-stop flight as opposed to having multiple stops).

Optimization: Based on the chosen representation and evaluation, you pick a strategy (mathematical/statistical) to achieve your goal.

Understanding the difference between training and testing

Regardless the situation of supervised, unsupervised (or even semi-supervised, weakly-supervised, reinforcement, …), we often define a loss (or error) function:

$$S_{training} = \{\mathbf{x}_i, i = 1..n\}$$

$$loss_{training} = \sum_{i=1}^{n} weight_i \cdot l(\mathbf{x}_i)$$

$weight_i$ and $l(\mathbf{x}_i)$ are weight and loss for each sample i

$$S_{testing} = \{\mathbf{x}_i, i = 1..u\}$$

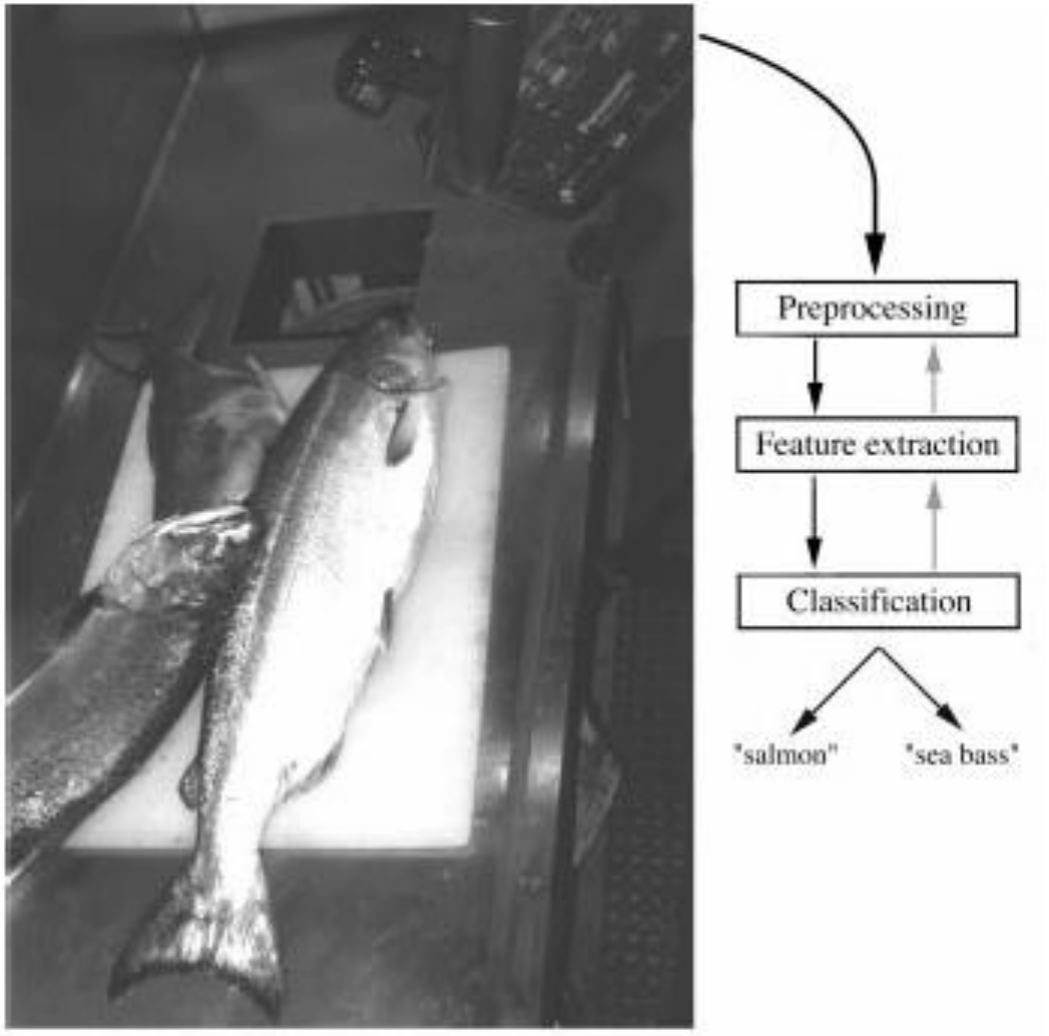
$$loss_{testing} = \sum_{i=1}^{u} weight_i \cdot l(\mathbf{x}_i)$$

$weight_i$ and $l(\mathbf{x}_i)$ are weight and loss for each sample i

$$loss_{testing} \neq loss_{training}$$

$$loss_{testing} \rightarrow loss_{training}$$

An example


Preprocessing
Feature extraction
Classification
"salmon"
"sea bass"

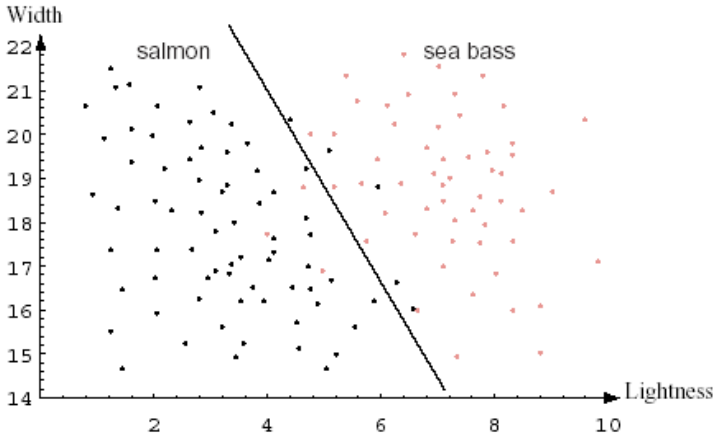
Summary of the problem

Let **x** be the input vector (observation) and y be its label:

Often, we are given a set of training data

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\} \qquad \mathbf{x} = (x_1, ..., x_m), x_i \in \mathcal{R}, \quad \mathbf{x} \in \mathcal{R}^m$$

We use the training set to train a classifier f(**x**).



Given a set of testing data, we make the prediction of each input and evaluate the algorithm.

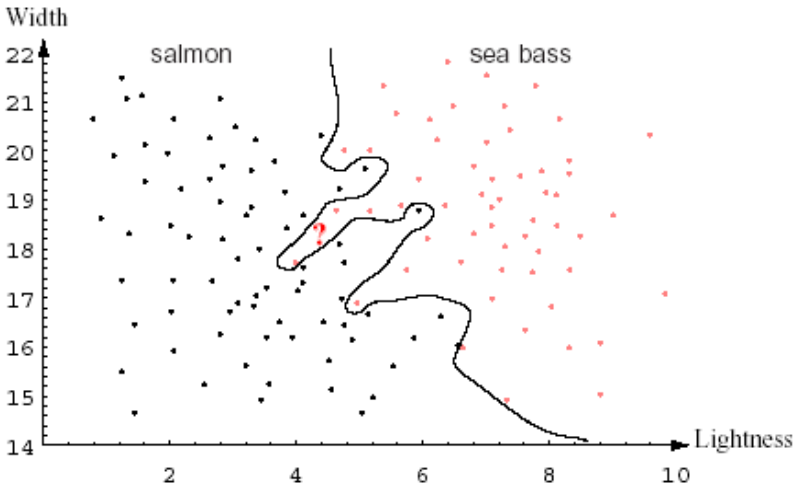
$$S_{testing} = \{(\mathbf{x}_i, y_i), i = 1..q\}.$$

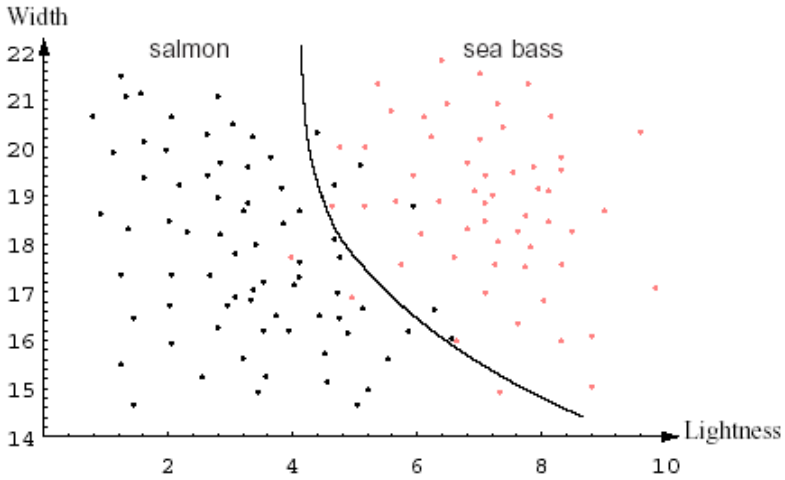
For each $\mathbf{x}_i$ we want to predict its y_i.

y_i is given to evaluate the quality of a classifier and is not given in reality.

Error

Now, let $f(\mathbf{x})$ be one classifier which makes the prediction for the label y, we define the error on a set of input as:





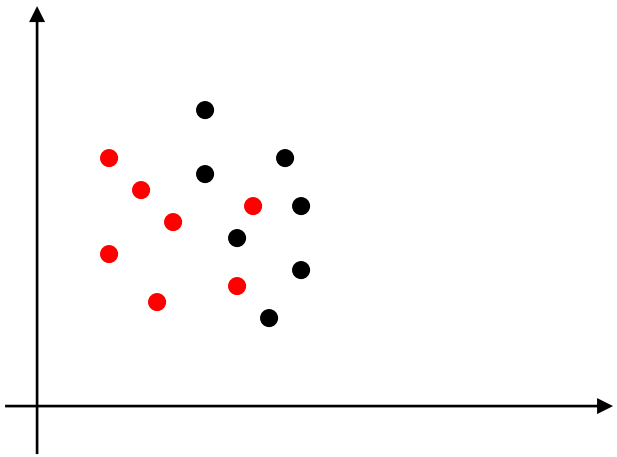
$$e_{training} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(y_i \neq f(\mathbf{x}_i)) \qquad \mathbf{1}(z) = \begin{cases} 1 & if\ z = TRUE \\ 0 & otherwise \end{cases}$$

$$e_{testing} = \frac{1}{q}\sum_{i=1}^{q} \mathbf{1}(y_i \neq f(\mathbf{x}_i))$$

Testing error

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\} \qquad S_{testing} = \{(\mathbf{x}_i, y_i), i = 1..q\}$$

$$e_{testing} = e_{training} + generalization(f)$$



$$e_{testing} = \frac{1}{q}\sum_{i=1}^{q} \mathbf{1}(y_i \neq f(\mathbf{x}_i))$$

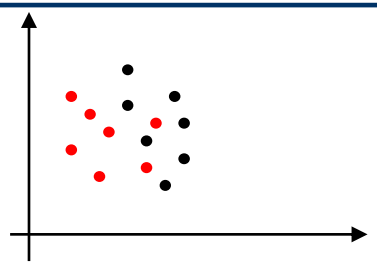
$$e_{testing} = 0.5?$$

1. $e_{testing} = 0.5 + 0.0$
2. $e_{testing} = 0.0 + 0.5$

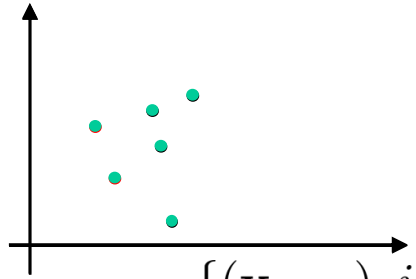
Question: What are the two extreme cases leading to the same testing error=0.5 (Assume we have the same number of positives and negatives.)

Testing error

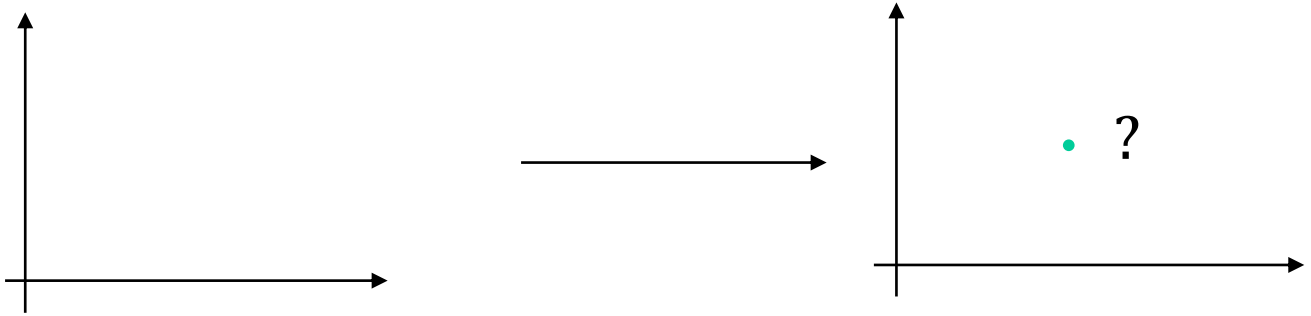
$$e_{testing} = e_{training} + generalization(f)$$



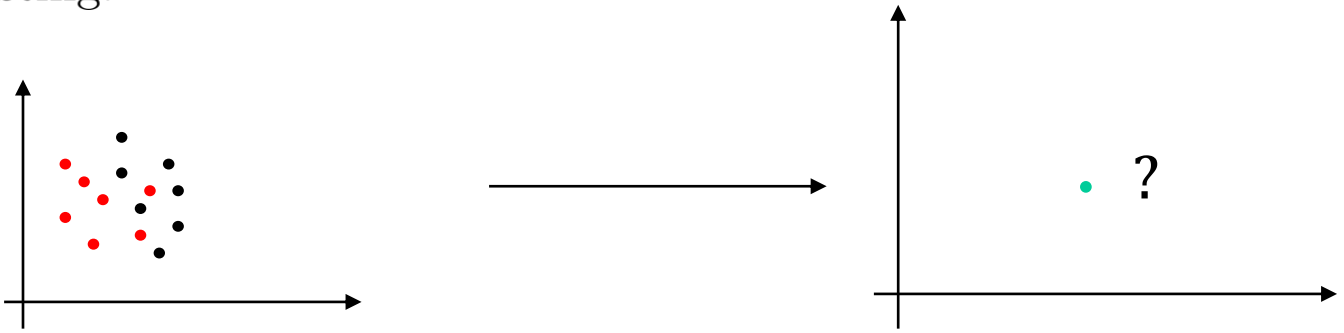
$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$



$$S_{testing} = \{(\mathbf{x}_i, \ ), i = 1..q\}$$



1. $e_{testing} = 0.5 + 0.0$: didn't learn anything during training; random guess at testing.



2. $e_{testing} = 0.0 + 0.5$: perfectly remember everything during training; many wrong predictions at testing.

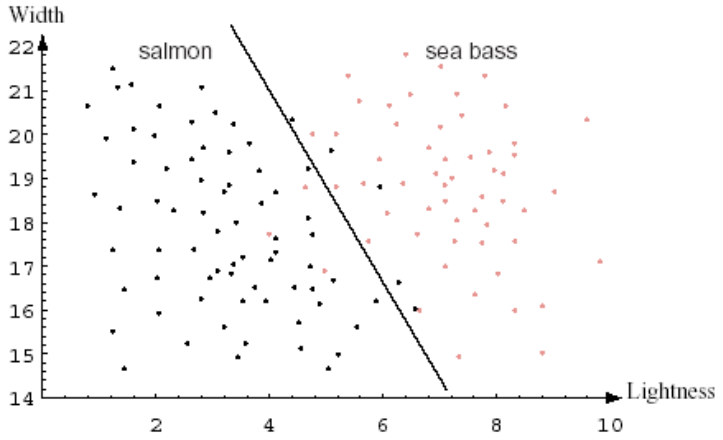
Decision boundary

Let **x** be the input vector (observation) and y be its label:

Often, we are given a set of training data

$$S = \{(\mathbf{x}_i, y_i), i = 1..n\} \quad \mathbf{x} = (x_1, ..., x_m), x_i \in \mathcal{R}, \quad \mathbf{x} \in \mathcal{R}^m$$

A classifier f(**x**):



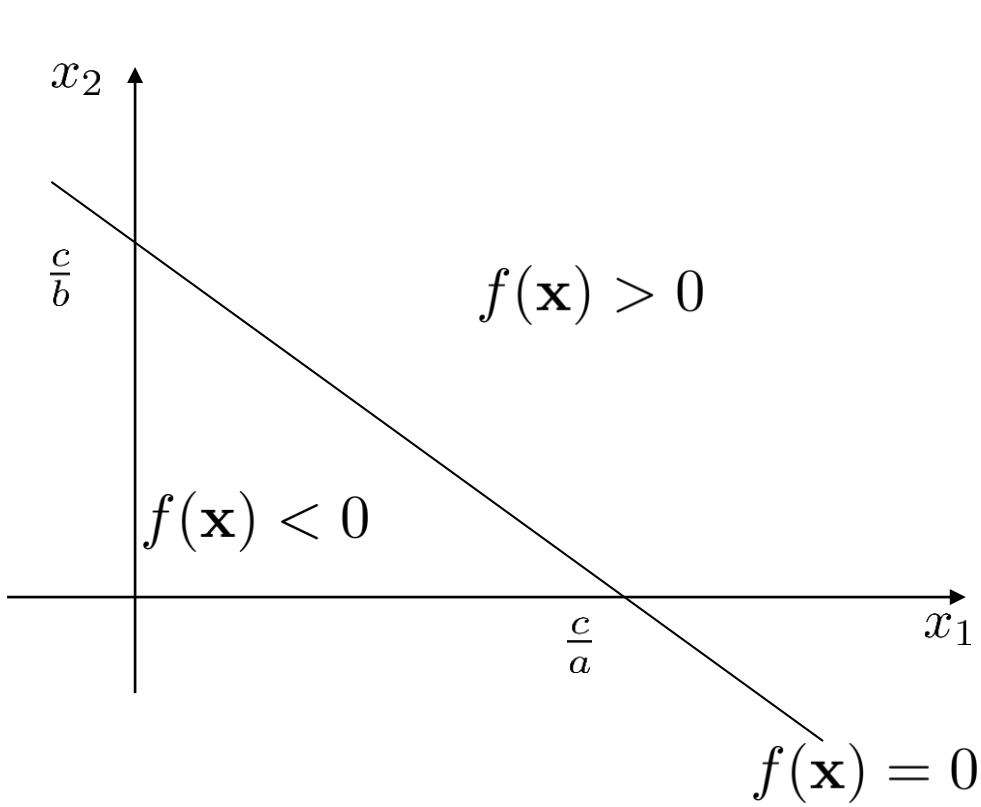
Decision boundary:
$\{\mathbf{x}_i, f(\mathbf{x}_i) = 0\}$

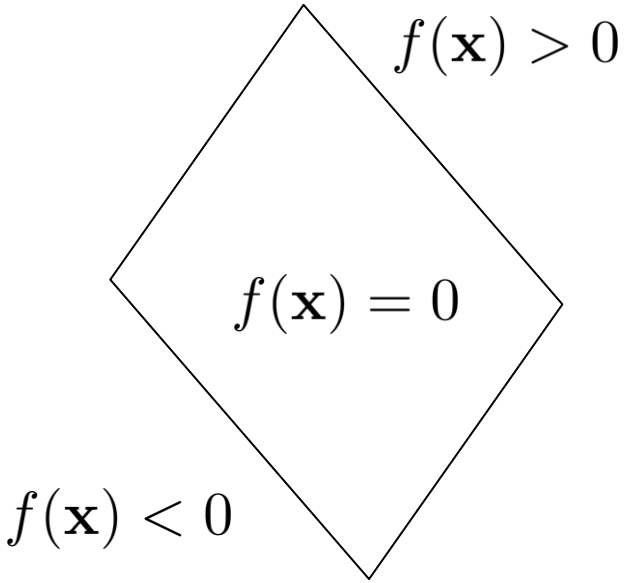
Decision boundary

Let **x** be the input vector (observation) and y be its label:

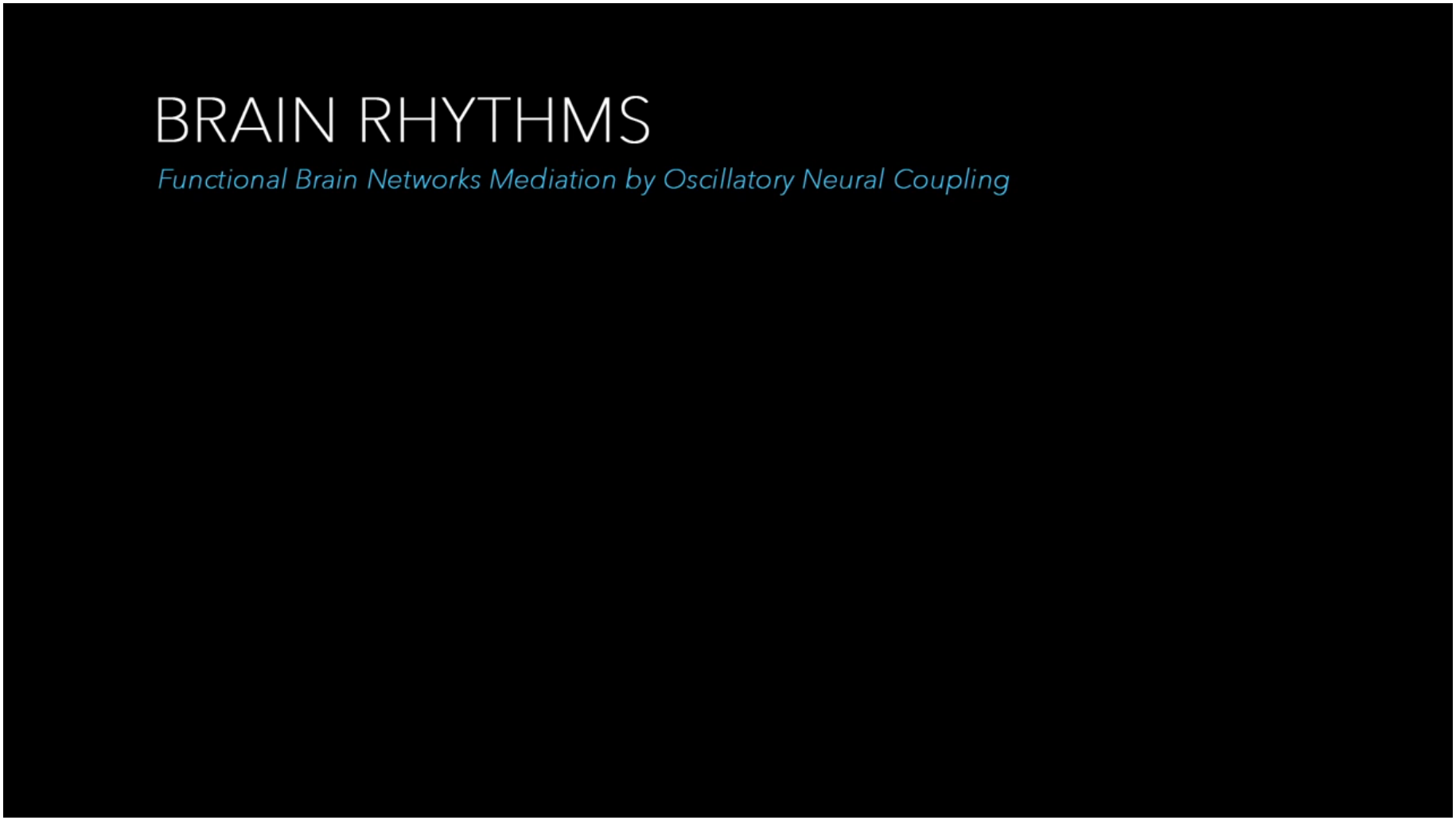
Decision boundary:
$\{\mathbf{x}_i, f(\mathbf{x}_i) = 0\}$

$$f(\mathbf{x}) = a \times x_1 + b \times x_2 - c$$





Brain Rhythms

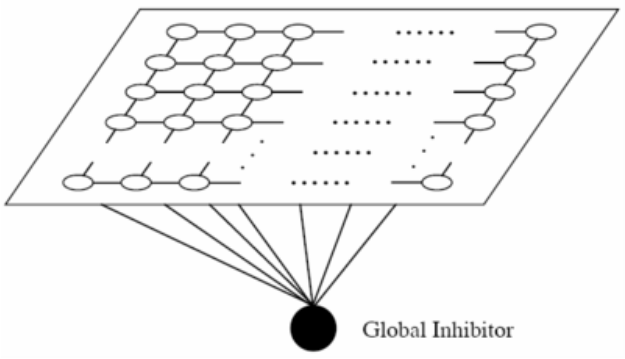


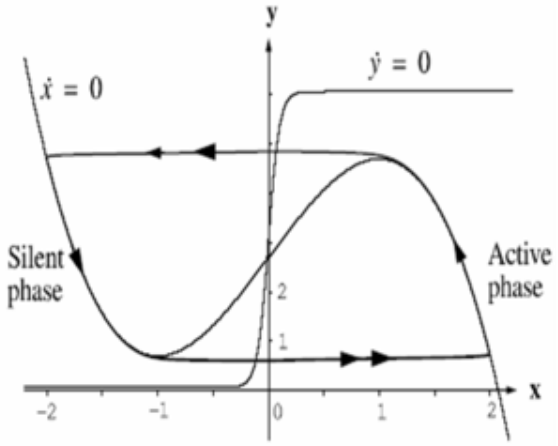
https://www.youtube.com/watch?v=OCpYdSN_kts

Oscillation Synchronization

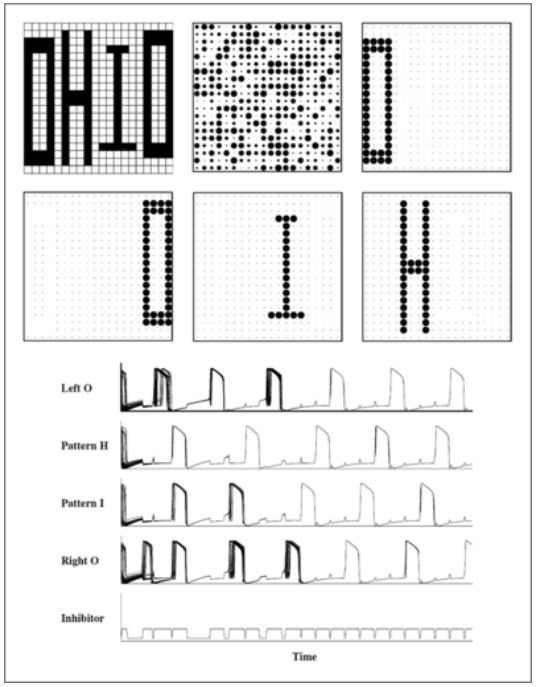
https://www.youtube.com/watch?v=St8wgLMR5Fo

"LEGION: locally excitatory globally inhibitory oscillator networks", Deliang Wang 2006.

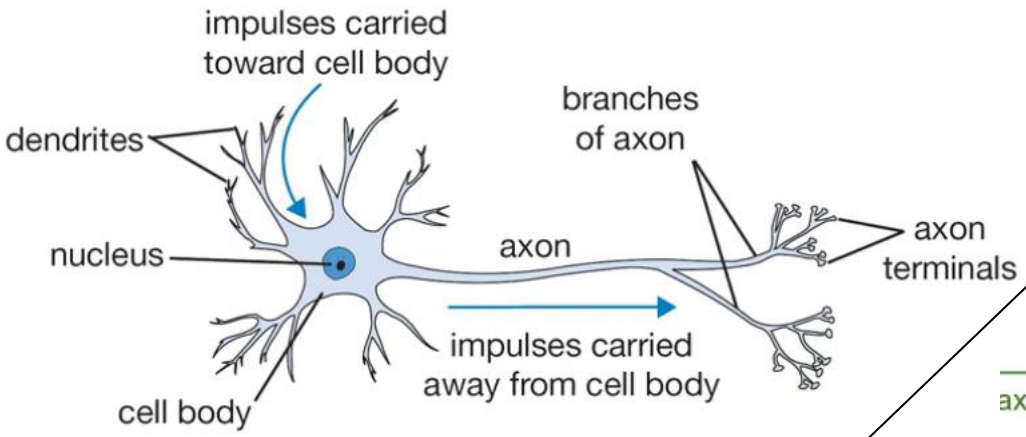


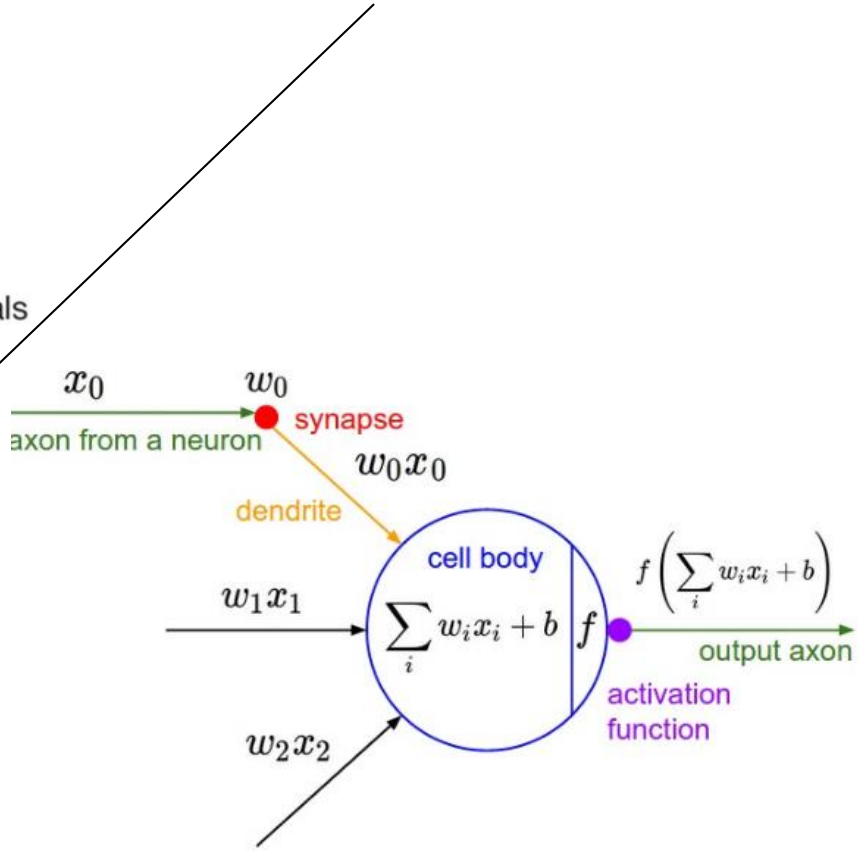


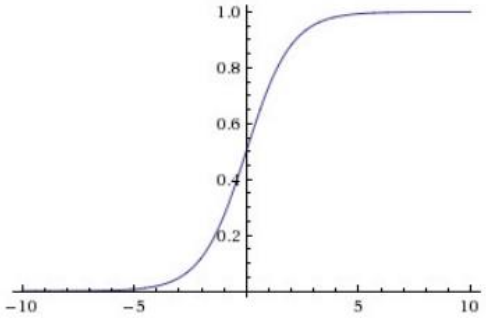
$$\begin{aligned} \dot{x} &= 3x - x^3 + 2 - y + I \\ \dot{y} &= \varepsilon(\alpha(1 + \tanh(x/\beta)) - y) \end{aligned}$$



Perceptron





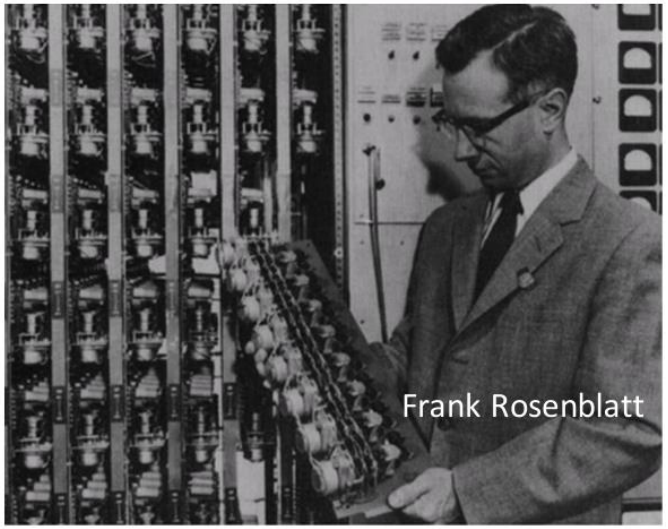


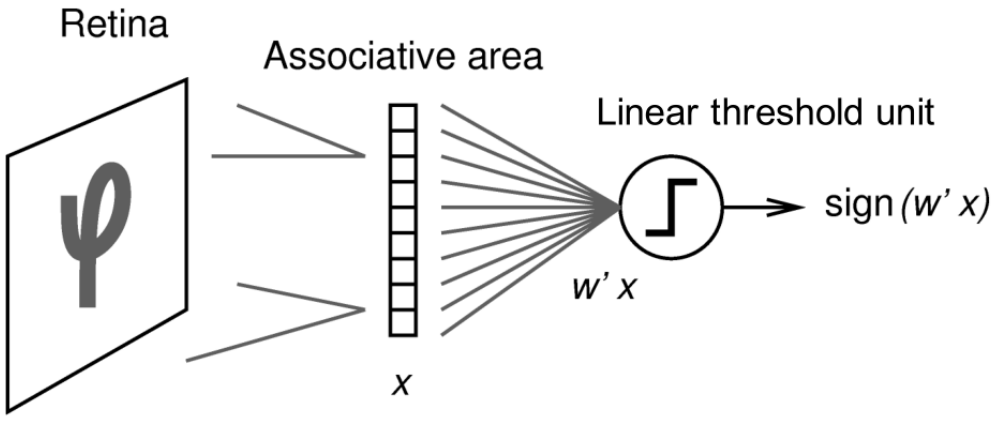
Sigmoid function

$$f(x) = \frac{1}{1+e^{-x}}$$

Fei-fei and Karpathy

Perceptron

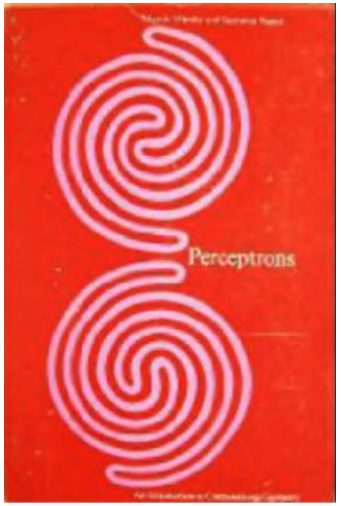




Supervised learning of the weights w using the Perceptron algorithm.

Slides from L. Bottou

Perceptrons (Frank Rosenblatt)

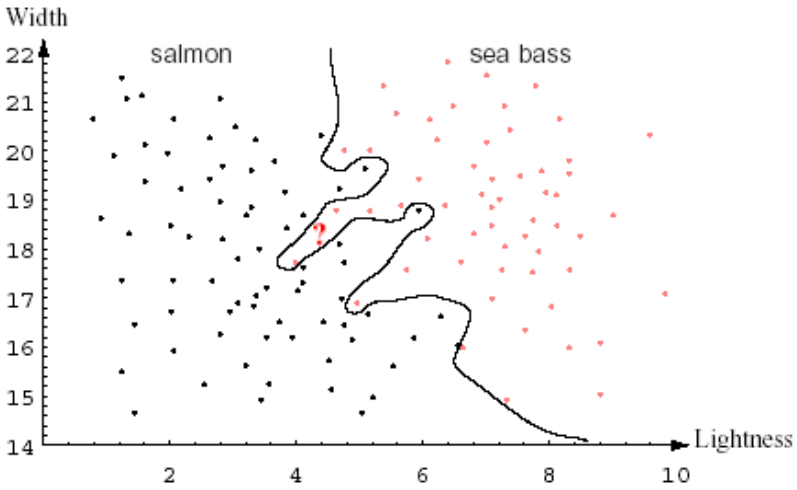


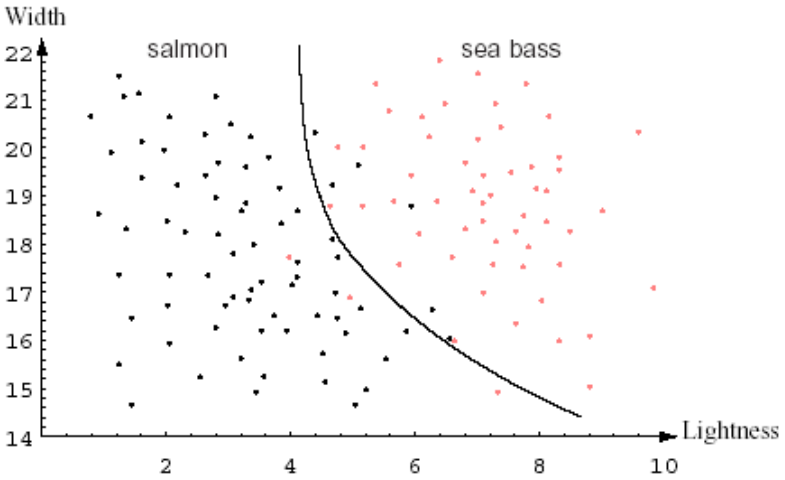
- Linear threshold units as Boolean gates.
- Circuit theory is poorly known.
- Learning deep circuits means .
- Solving the credit assignment problem.
- Linearly separable problems are few.
- Elementary problems need complex circu its (parity, connexity, invariances).
- But have simple algorithmic solutions (programming versus learning).

Slides from L. Bottou

Error

Now, let $f(\mathbf{x})$ be one classifier which makes the prediction for the label y, we define the error on a set of input as:





$$e_{training} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(y_i \neq f(\mathbf{x}_i)) \qquad \mathbf{1}(z) = \begin{cases} 1 & if\ z = TRUE \\ 0 & otherwise \end{cases}$$

$$e_{testing} = \frac{1}{q}\sum_{i=1}^{q} \mathbf{1}(y_i \neq f(\mathbf{x}_i))$$

Error measures and metrics

$$e_{training} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}(y_i \neq f(\mathbf{x}_i))$$

$$accuracy_{training} = 1 - e_{training}$$

$$e_{testing} = \frac{1}{q} \sum_{i=1}^{q} \mathbf{1}(y_i \neq f(\mathbf{x}_i))$$

$$accuracy_{testing} = 1 - e_{testing}$$

Error measures and metrics

	test	
Sick+	Sick+ and test+	Sick+ and test-
	Sick- and test-	Sick- and test-

True positive rate (correctly identified): $P(test+|sick+)$
False positive rate (incorrectly identified): $P(test+|sick-)$
True negative rate (correctly rejected): $P(test-|sick-)$
False negative rate (incorrectly rejected): $P(test-|sick+)$

Error metrics

		Condition (as determined by "Gold standard")			
	Total population	Condition positive	Condition negative	Prevalence = Σ Condition positive / Σ Total population	
Test outcome	Test outcome positive	**True positive**	**False positive** (Type I error)	Positive predictive value (PPV, Precision) = Σ True positive / Σ Test outcome positive	False discovery rate (FDR) = Σ False positive / Σ Test outcome positive
	Test outcome negative	**False negative** (Type II error)	**True negative**	False omission rate (FOR) = Σ False negative / Σ Test outcome negative	Negative predictive value (NPV) = Σ True negative / Σ Test outcome negative
	Positive likelihood ratio (**LR+**) = TPR/FPR	True positive rate (TPR, Sensitivity, Recall) = Σ True positive / Σ Condition positive	False positive rate (FPR, Fall-out) = Σ False positive / Σ Condition negative	Accuracy (ACC) = Σ True positive + Σ True negative / Σ Total population	
	Negative likelihood ratio (**LR−**) = FNR/TNR	False negative rate (FNR) = Σ False negative / Σ Condition positive	True negative rate (TNR, Specificity, SPC) = Σ True negative / Σ Condition negative		
	Diagnostic odds ratio (**DOR**) = LR+/LR−				

$$\text{sensitivity} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}}$$

$$= \frac{\text{number of true positives}}{\text{total number of sick individuals in population}}$$

$$= \text{probability of a positive test, given that the patient is ill}$$

$$\text{specificity} = \frac{\text{number of true negatives}}{\text{number of true negatives} + \text{number of false positives}}$$

$$= \frac{\text{number of true negatives}}{\text{total number of well individuals in population}}$$

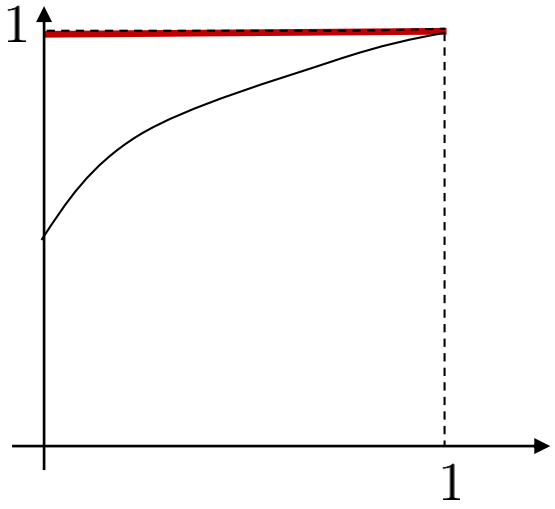
$$= \text{probability of a negative test given that the patient is well}$$

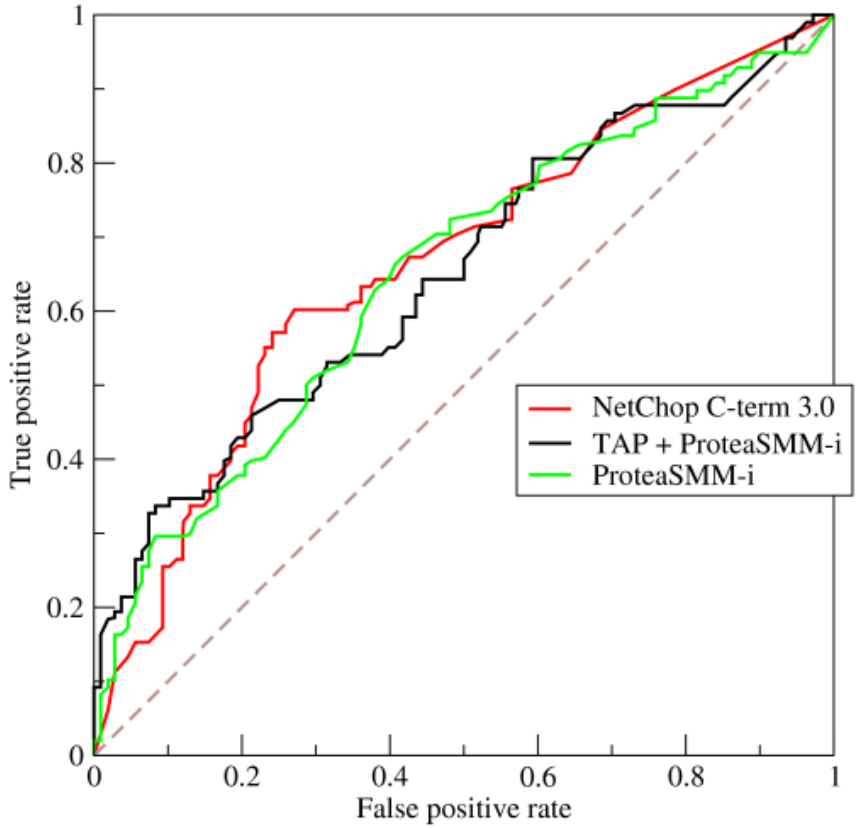
Error metrics: receiver operating characteristic (ROC)

sensitivity

recall

$$P(test+|sick+) = \frac{P(sick+,test+)}{P(sick+)}$$





1-specificity

$$P(test+|sick-) = \frac{P(sick-,test+)}{P(sick-,test+)+P(sick-,test-)}$$

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

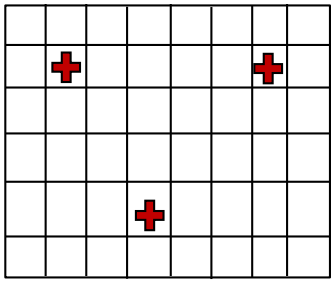
Error metric

$$\text{precision}=P(target|hit) = \frac{\#(target,hit)}{\#(hit)}$$

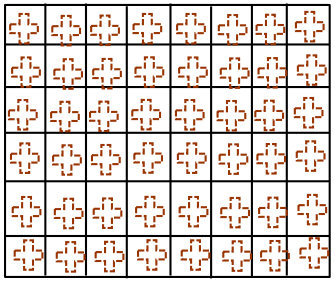
$$\text{recall}=P(hit|target) = \frac{\#(target,hit)}{\#(hit)}$$

$$\text{F-value}=\frac{2\times precision\times recall}{precision+recall}$$

6×8 possible locations



3 targets to hit

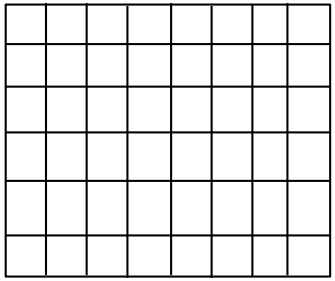


all hit

precision$=\frac{3}{48}$

recall$=\frac{3}{3}$

F-value$=\frac{0.03125}{1.0625}$

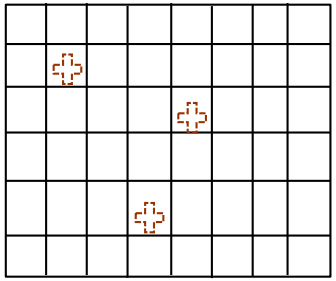


zero hit

precision$=\frac{0}{0}$

recall$=\frac{0}{3}$

F-value$=\frac{0}{1}$



miss one

precision$=\frac{2}{3}$

recall$=\frac{2}{3}$

F-value≈ 0.667

Error metrics

$$\text{precision} = \frac{sick\ and\ test+}{total\ \ test+}$$

$$\text{recall} = \frac{sick\ \ and\ \ test+}{totalsick+}$$

Problem overview

$$e_{testing} = e_{training} + generalization(f)$$

We will focus on training error for the moment:

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..m\}$$

$$\mathbf{x} = [X_1, ..., X_k], X_1 \in \mathcal{R}, \quad \mathbf{x} \in \mathcal{R}^k \qquad \mathbf{x} \sim p(\mathbf{x})$$

$$e_{training} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}(y_i \neq f(\mathbf{x}_i))$$

In general:

$$e_{training} = \sum_{i=1}^{m} p(\mathbf{x}_i)(y_i - f(\mathbf{x}_i; \theta))^2$$

Estimation and optimization

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..m\}$$

$$\mathbf{x} = [X_1, ..., X_k], X_1 \in \mathcal{R}, \quad \mathbf{x} \in \mathcal{R}^k \qquad \mathbf{x} \sim p(\mathbf{x})$$

Different choices of the penalty will lead to different robustness measure:

L2 norm:

$$e = \sum_{i=1}^{m} p(\mathbf{x}_i)(y_i - f(\mathbf{x}_i; \theta))^2$$

L1 norm:

$$e = \sum_{i=1}^{m} p(\mathbf{x}_i)|y_i - f(\mathbf{x}_i; \theta)|$$

Estimation and optimization

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..m\}$$

$$\mathbf{x} = [X_1, ..., X_k], X_1 \in \mathcal{R}, \quad \mathbf{x} \in \mathcal{R}^k \qquad \mathbf{x} \sim p(\mathbf{x})$$

Different choices of the penalty will lead to different robustness measure:

L2 norm:

$$e = \sum_{i=1}^{m} p(\mathbf{x}_i)(y_i - f(\mathbf{x}_i; \theta))^2$$

L1 norm:

$$e = \sum_{i=1}^{m} p(\mathbf{x}_i)|y_i - f(\mathbf{x}_i; \theta)|$$

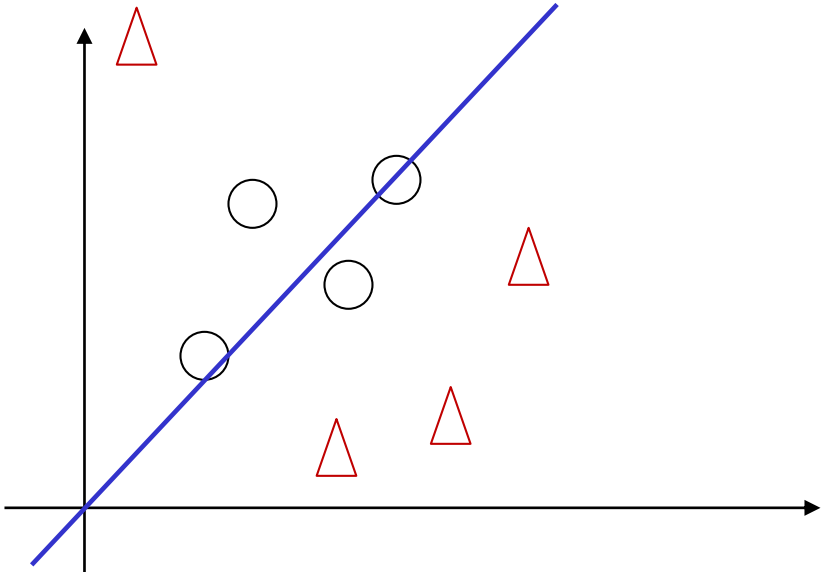
Estimation and optimization

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..m\}$$

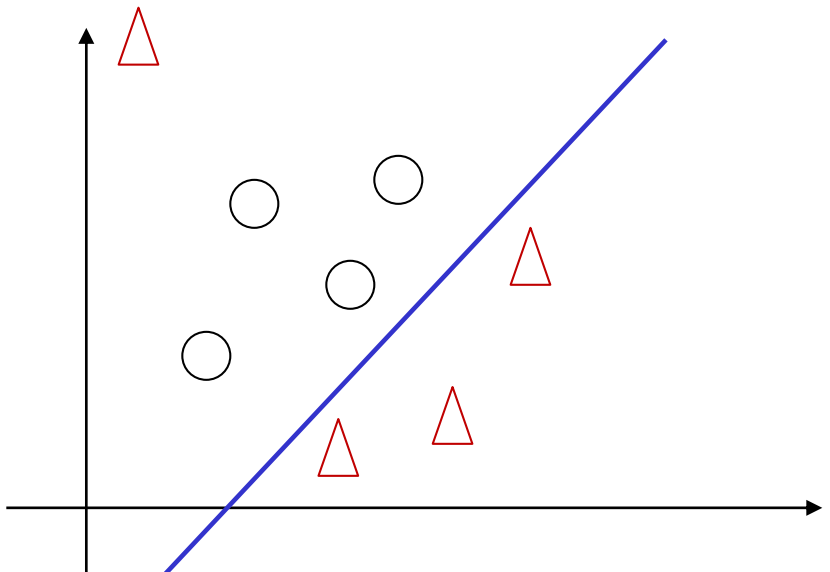
$$\mathbf{x} = [X_1, ..., X_k], X_1 \in \mathcal{R}, \quad \mathbf{x} \in \mathcal{R}^k \qquad \mathbf{x} \sim p(\mathbf{x})$$

A general form:

$$\theta^* = \arg min_\theta \sum_{i=1}^{m} p(\mathbf{x}_i) cost(y_i - f(\mathbf{x}_i; \theta))$$



$cost(y_i - f(\mathbf{x}_i; \theta)) = (y_i - f(\mathbf{x}_i; \theta))^2$



$cost(y_i - f(\mathbf{x}_i; \theta)) = |y_i - f(\mathbf{x}_i; \theta)|$

Estimation and optimization

$$\theta^* = \arg min_\theta \quad g(\theta)$$

Learning and estimation with convex functions:

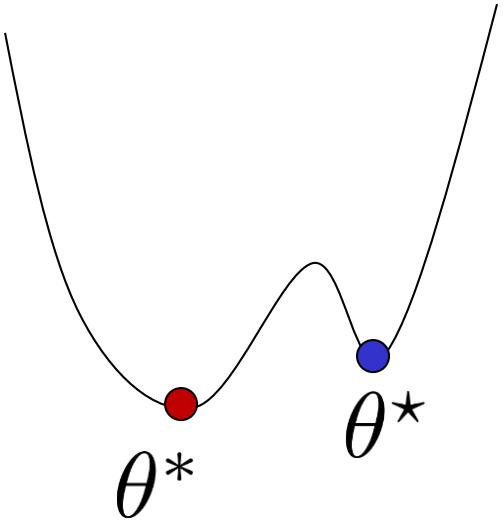
http://stanford.edu/~boyd/

Convex Optimization
Stephen Boyd and Lieven Vandenberghe

Cambridge University Press

A new MOOC on convex optimization, CVX101, will run from 1/21/14 to 3/14/14.

Optimization



Things we ofen need to be able to do to solve optimization problem:

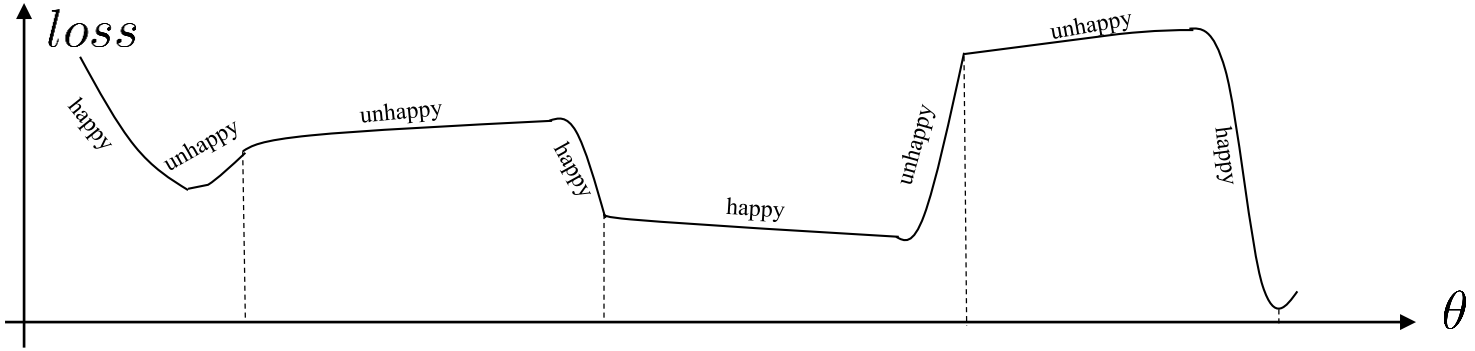
1. $\forall\theta$, check if $\theta \in \Omega$?
2. For $\forall\theta$, computing $g(\theta)$, $\nabla g(\theta)$, $\nabla^2 g(\theta)$.

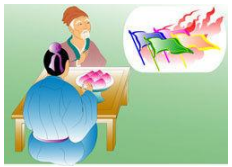
Definition:

1. θ^* is a globally optimal solution for $\theta^* \in \Omega$ and $g(\theta^*) \leq g(\theta) \forall\theta \in \Omega$
2. $\theta^\star$ is a locally optimal solution if there is a neighborhood $\mathcal{N}$ around θ such that $\theta^\star \in \Omega$, $g(\theta^\star) \leq g(\theta)$, $\forall\theta \in \mathcal{N} \cap \Omega$.

A perspective of estimation

Mr. Sai
http://www.baike.com/wiki/





An analogy:
we want to find the lowest point in the figure.

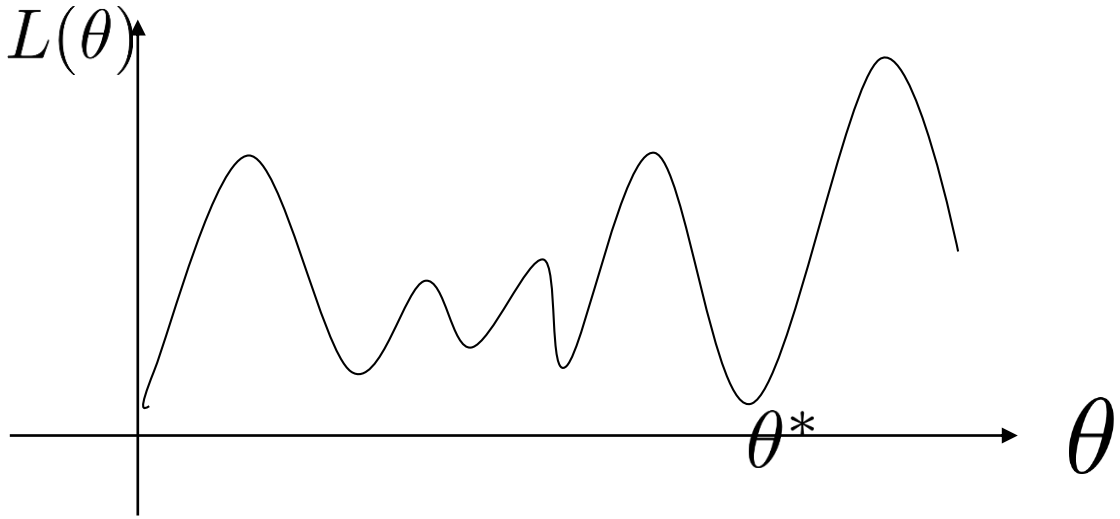
http://menpiao.daiwoqu.com/

Optimization: argmin

$$\theta^* = \arg\min_\theta L(\theta)$$

The operator arg min defines the optimal value (in the argument of function $L()$) θ^* that minimizes $L(\theta)$

$\arg\min L(\theta)$ doesn't return the value of $L(\theta)$

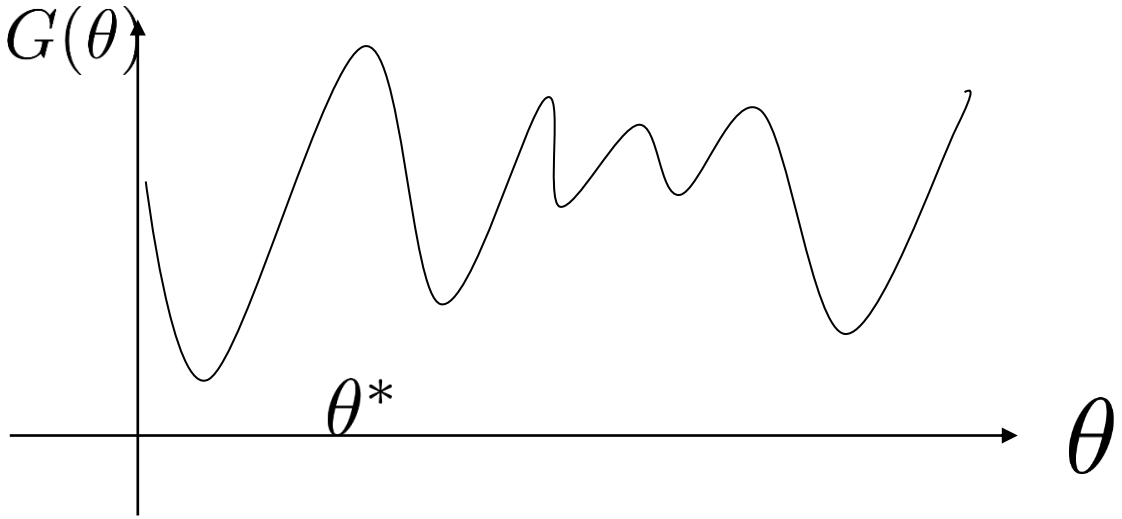


Optimization: argmax

$$\theta^* = \arg\max_\theta G(\theta)$$

The operator arg max defines the optimal value (in the argument of function $G()$) θ^* that maximizes $G(\theta)$

$\arg\max G(\theta)$ doesn't return the value of $G(\theta)$



Optimization: argmin

$$\theta^* = \arg\min_\theta L(\theta)$$

If a function $g(v)$ is monotonic, e.g. $\forall v_1 > v_2$ it is always true that $g(v_1) > g(v_2)$, then:

$$\theta^* = \arg\min_\theta L(\theta) = \arg\min_\theta g(L(\theta))$$

For example,

if $g(v) = 2 \times v + 10$

$$\theta^* = \arg\min_\theta L(\theta) = \arg\min_\theta 2 \times L(\theta) + 10$$

Optimization: argmax

$$\theta^* = \arg\max_\theta G(\theta)$$

If a function $g(v)$ is monotonic, e.g. $\forall v_1 > v_2$ it is always true that $g(v_1) > g(v_2)$, then:

$$\theta^* = \arg\max_\theta G(\theta) = \arg\max_\theta g(G(\theta))$$

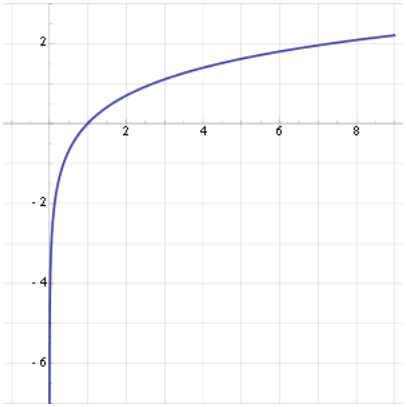
For example,

if $g(v) = 2 \times v + 10$

$$\theta^* = \arg\max_\theta G(\theta) = \arg\max_\theta 2 \times G(\theta) + 10$$

argmin and argmax

The function $\ln(v)$ is monotonic, e.g. $\forall v_1 > v_2$ it is always true that $\ln(v_1) > \ln(v_2)$, then:



$$\theta^* = \arg\max_\theta G(\theta)$$

$$= \arg\max_\theta \ln(G(\theta))$$

$$= \arg\min_\theta -\ln(G(\theta))$$

Problem Definition and High-level Understanding

Regression: predicting blood presure

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\} \qquad \mathbf{x}_i = (x_{i1}, ..., x_{im}) \qquad y \in \mathcal{R}$$

blood presure	age	male or female	weight (lb)	height (cm)
y_1=131	$x_{11} = 22$	$x_{12} = M$	$x_{13} = 160$	$x_{14} = 180$
y_2=150	$x_{21} = 51$	$x_{22} = M$	$x_{23} = 190$	$x_{24} = 175$
y_3=105	$x_{31} = 43$	$x_{32} = F$	$x_{33} = 120$	$x_{34} = 165$

$$Y = \begin{pmatrix} 131 \\ 150 \\ 105 \end{pmatrix} \qquad X = \begin{pmatrix} 22 & 51 & 43 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 160 & 190 & 120 \\ 180 & 175 & 165 \end{pmatrix}$$

$$\theta^* = \arg\min_\theta L(\theta)$$

$$Loss: L(\theta) = ||Y - X^T\theta||$$

Difference between training values Y and predicted values $X^T\theta$.

Problem Definition and High-level Understanding

Classification: predicting if someone is doing exercising

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\} \qquad \mathbf{x}_i = (x_{i1}, ..., x_{im}) \qquad y \in \{-1, +1\}$$

exercise	age	male or female	weight (lb)	height (cm)
yes (+1)	$x_{11} = 22$	$x_{12} = M$	$x_{13} = 160$	$x_{14} = 180$
no (−1)	$x_{21} = 51$	$x_{22} = M$	$x_{23} = 190$	$x_{24} = 175$
yes (+1)	$x_{31} = 43$	$x_{32} = F$	$x_{33} = 120$	$x_{34} = 165$

$$Y = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \qquad X = \begin{pmatrix} 22 & 51 & 43 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 160 & 190 & 120 \\ 180 & 175 & 165 \end{pmatrix}$$

$$\theta^* = \arg\min_\theta L(\theta)$$

$$Loss: L(\theta) = ||Y - f(X^T\theta)||$$

elementwise

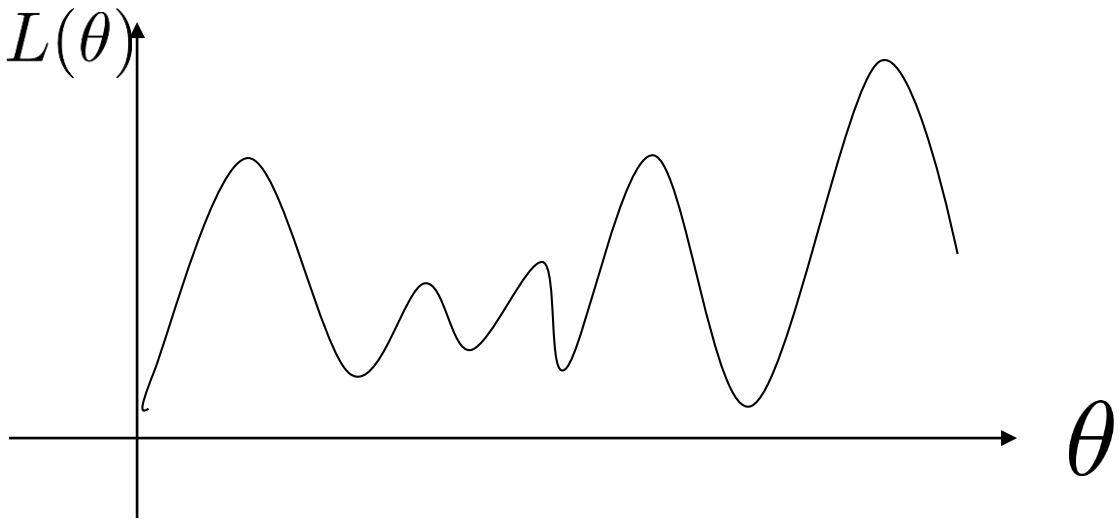
$$f(X^T\theta) = \begin{cases} 1 & if\ X^T\theta \geq 0 \\ -1 & otherwise \end{cases}$$

Difference between training values Y and predicted values $X^T\theta$.

Learning

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\} \qquad \mathbf{x}_i = (x_{i1}, ..., x_{im}) \qquad y \in \{-1, +1\}$$

$$\theta^* = \arg\min_\theta L(\theta)$$



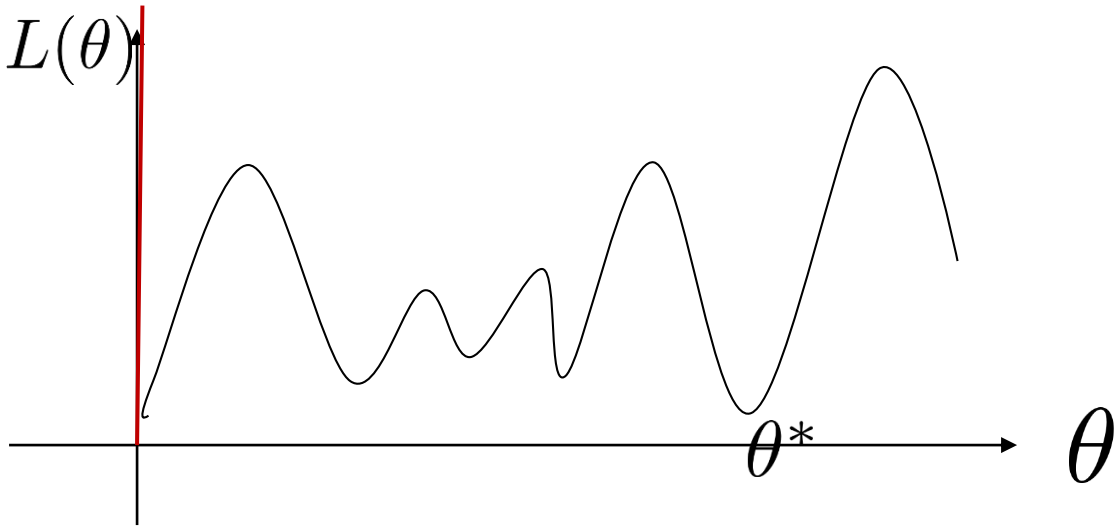
Given a θ, you can always evaluate the $L(\theta)$.

But you don't know which θ^* results in the minimum value of $L(\theta)$.

Optimization: exhaustive search

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\} \qquad \mathbf{x}_i = (x_{i1}, ..., x_{im}) \qquad y \in \{-1, +1\}$$

$$\theta^* = \arg\min_\theta L(\theta)$$



If you don't make any assumption about $L(\theta)$, a straight-forward way is to perform exhustive search: searching for all possible θ.

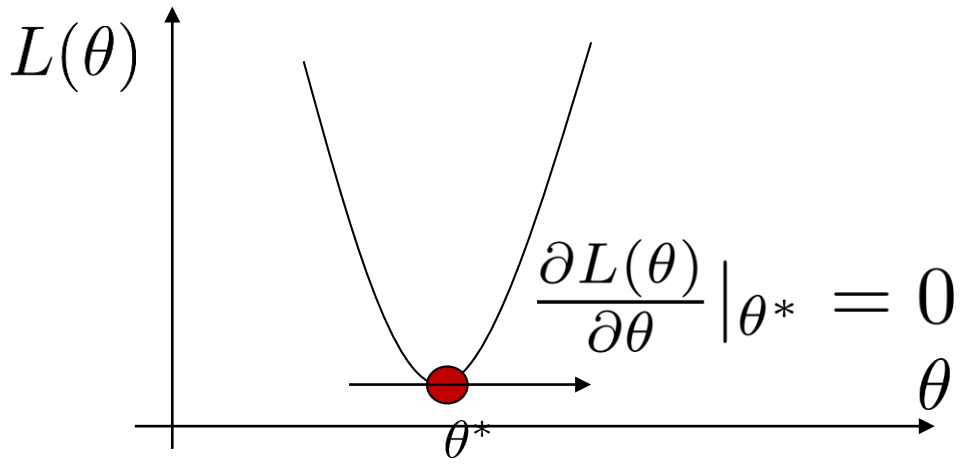
You are guaranteed to find the optimal θ^*,

but with an extremely high computational cost.

Optimization: closed form solution

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\} \qquad \mathbf{x}_i = (x_{i1}, ..., x_{im}) \qquad y \in \{-1, +1\}$$

$$\theta^* = \arg\min_\theta L(\theta)$$



If you $L(\theta)$ is convex and everywhere differentiable:

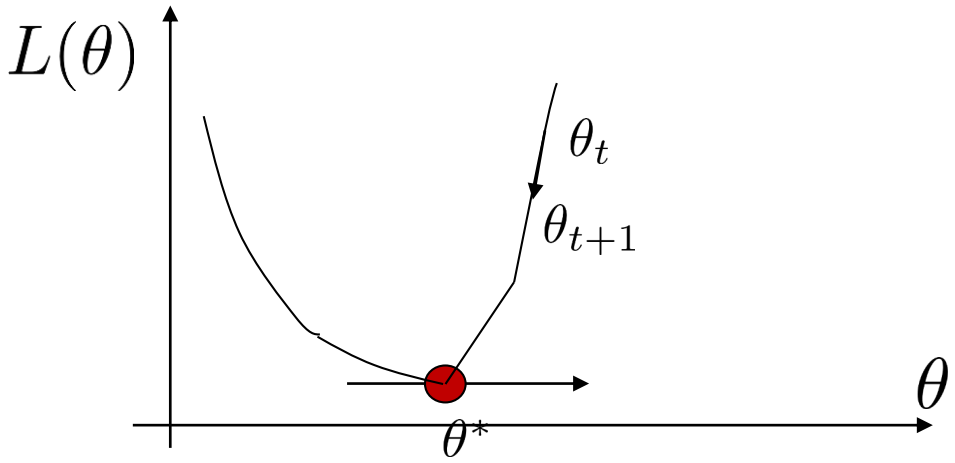
$$\text{Set: } \frac{\partial L(\theta)}{\partial \theta} = 0$$

$$\theta^* \text{ is the solution to } \frac{\partial L(\theta)}{\partial \theta}|_{\theta^*} = 0$$

Optimization: gradient descent

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\} \qquad \mathbf{x}_i = (x_{i1}, ..., x_{im}) \qquad y \in \{-1, +1\}$$

$$\theta^* = \arg\min_\theta L(\theta)$$



If you $L(\theta)$ is convex but NOT everywhere differentiable:

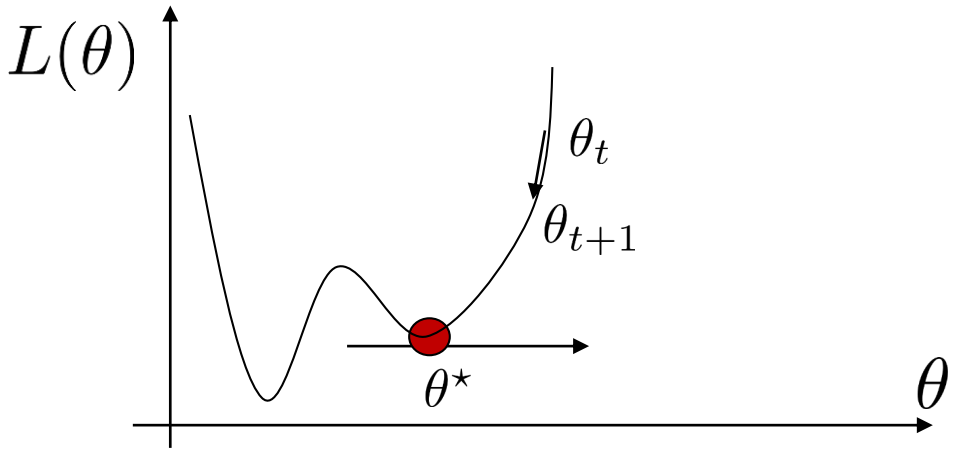
Search for θ^* such that $\frac{\partial L(\theta)}{\partial \theta}|_{\theta^*} = 0$

using gradient descent $\theta_{t+1} = \theta_t - \lambda_t \frac{\partial L(\theta)}{\partial \theta}$

Optimization: gradient descent

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\} \qquad \mathbf{x}_i = (x_{i1}, ..., x_{im}) \qquad y \in \{-1, +1\}$$

$$\theta^* = \arg\min_\theta L(\theta)$$



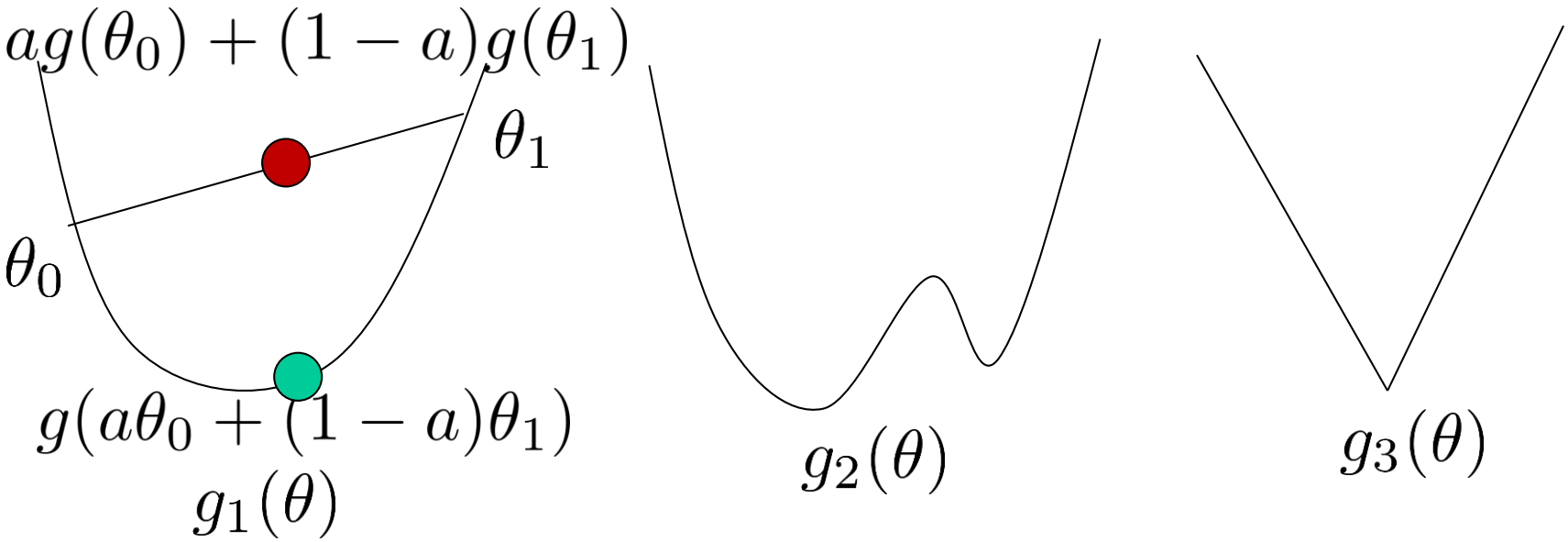
If you $L(\theta)$ is non-convex:

Search for θ^* such that $\frac{\partial L(\theta)}{\partial \theta}|_{\theta^*} = 0$

using gradient descent $\theta_{t+1} = \theta_t - \lambda_t \frac{\partial L(\theta)}{\partial \theta}$

Convex functions

$$\theta^* = \arg min_\theta \quad g(\theta)$$



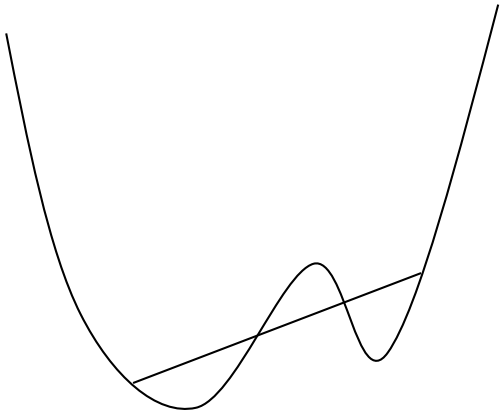
Definition:

$$\forall \theta_0, \theta_1, a \in [0, 1]$$

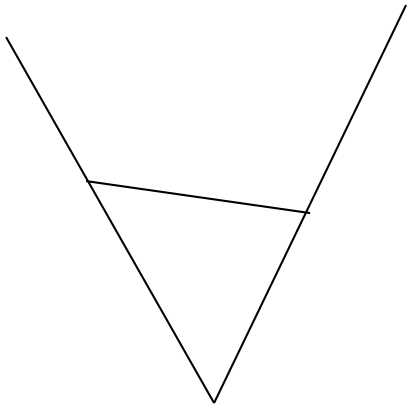
$$ag(\theta_0) + (1-a)g(\theta_1) \geq g(a\theta_0 + (1-a)\theta_1)$$

Convex functions

$$\theta^* = \arg min_\theta \quad g(\theta)$$



$g_2(\theta)$



$g_3(\theta)$

$$\forall \theta_0, \theta_1, a \in [0,1]$$

$$ag(\theta_0) + (1-a)g(\theta_1) \geq g(a\theta_0 + (1-a)\theta_1)$$

Alternatively (for differentiable function):

$$g(\theta') \geq g(\theta) + < \nabla g(\theta), \theta' - \theta >$$

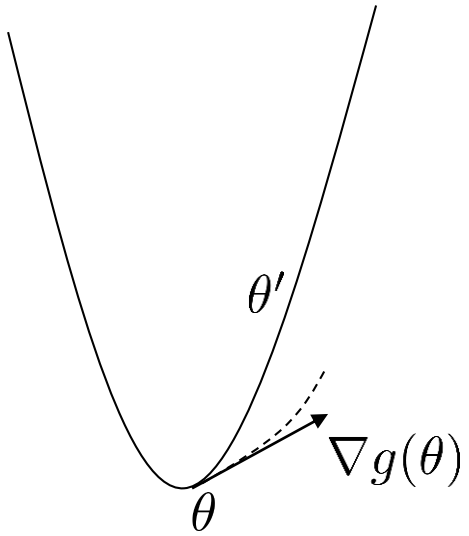
Strongly convex

Strongly convex: with parameter $\lambda > 0$

$$< \nabla g(\theta_0) - \nabla g(\theta_1), \theta_0 - \theta_1 > \quad \geq \lambda ||\theta_0 - \theta_1||_2^2$$

Equivalently:

$$g(\theta_1) \geq g(\theta_0) + < \nabla g(\theta_0), \theta_1 - \theta_0 > + \frac{\lambda}{2} ||\theta_1 - \theta_0||_2^2$$



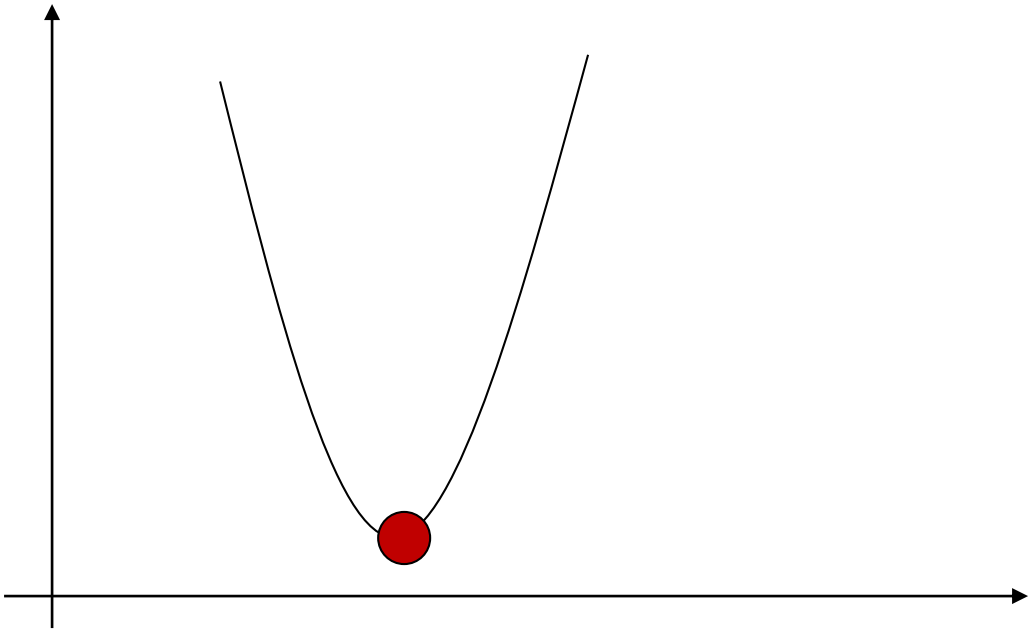
A strongly convex function is convex.

$$\theta^* = \arg min_\theta \quad f(\theta) = \arg_\theta [f'(\theta) \equiv 0]$$

A convex function is not necessarily strongly convex.

$$2 \times (\theta - 3) = 0 \rightarrow \theta = 3$$

Convex function: differentiable



$$\theta^* = \arg min_\theta g(\theta)$$

$$g(\theta) = (\theta - 3)^2 + 4$$

$$\frac{dg(\theta)}{d\theta} = 2 \times (\theta - 3) \qquad \frac{dg(\theta)}{d\theta} = 0$$

$$2 \times (\theta - 3) = 0 \rightarrow \theta = 3$$

General approaches for optimization

- Exhaustive search
- Gradient descent
- Coordinate descent
- Newton's method
- Line search
- Stochastic computing
- Stochastic sampling (Markov chain Monte Carlo)
- ….

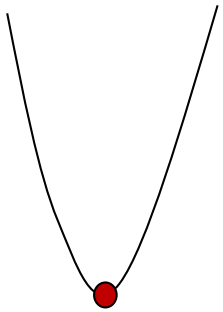
Parameters

Throughout the quarter in the class, we use either

$$W \text{ and } \theta$$

to denote the underlying model parameters that we want to learn.

Quadratic function: least square estimation



$$S_{training} = \{(x_i, y_i), i = 1..n\} \quad y_i \in \mathcal{R}$$

Obtain/train: $f(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2$

$$W = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$$

$$W^* = \arg min_W \sum_i (\mathbf{x}_i^T \cdot W - y_i)^2$$

Let: $\mathbf{X} = (\mathbf{x}_i, ..., \mathbf{x}_n)^T \qquad Y = (y_1, ..., y_n)^T$

$$\mathbf{x}_i = \begin{pmatrix} 1 \\ x_i \\ x_i^2 \end{pmatrix}$$

$$W^* = \arg min_W = \arg min_W g(W) = (X \cdot W - Y)^T (X \cdot W - Y)$$

$$g(W) = W^T X^T X W - W^T X^T Y - Y^T X W + Y^T Y$$

$$\frac{dg(W)}{dW} = 2X^T X W - 2X^T Y = 0$$

$$W^* = (X^T X)^{-1} X^T Y$$

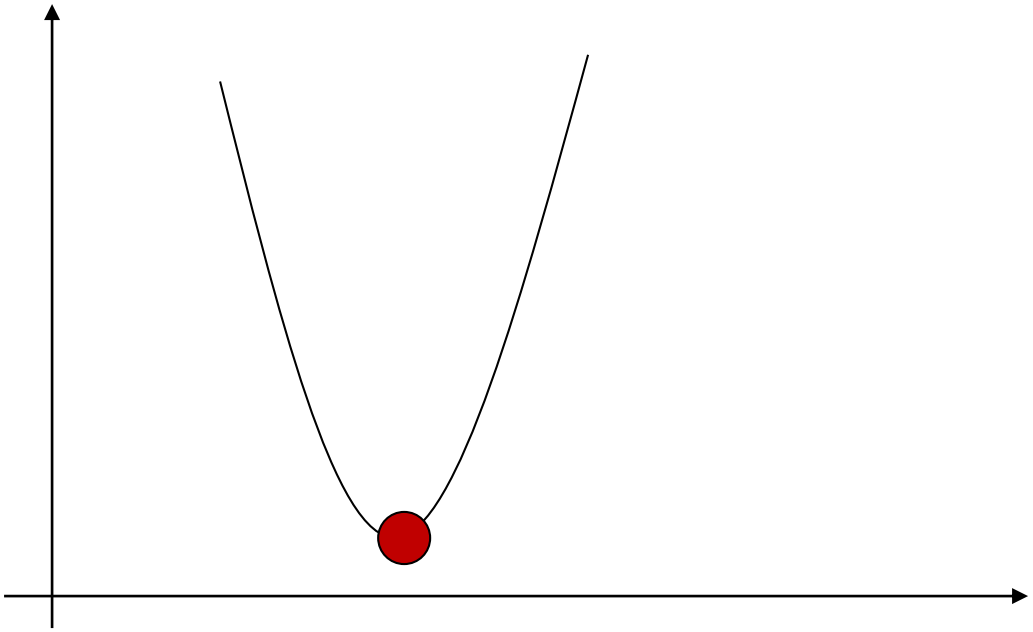
In matlab, you can simply call $X \setminus Y$ or $pinv(X) * Y$

In matlab

```
X=rand(1,20);
Y=10.0+X*3.0+randn(1,20)/2.0;
plot(X,Y,'.');
X(2,:)=1.0;

c=inv(X*X')*X*Y'
hold on;
plot([0,1],[c(2),c(2)+c(1)],'r');
```

Convex function: differentiable



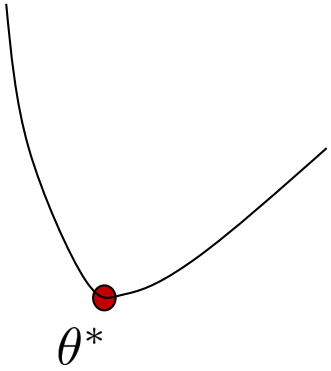
$$\theta^* = \arg min_\theta g(\theta)$$

$$g(\theta) = (\theta - 3)^2 + 4$$

$$\frac{df(\theta)}{d\theta} = 2 \times (\theta - 3) \qquad \frac{df(\theta)}{d\theta} = 0$$

$$2 \times (\theta - 3) = 0 \rightarrow \theta = 3$$

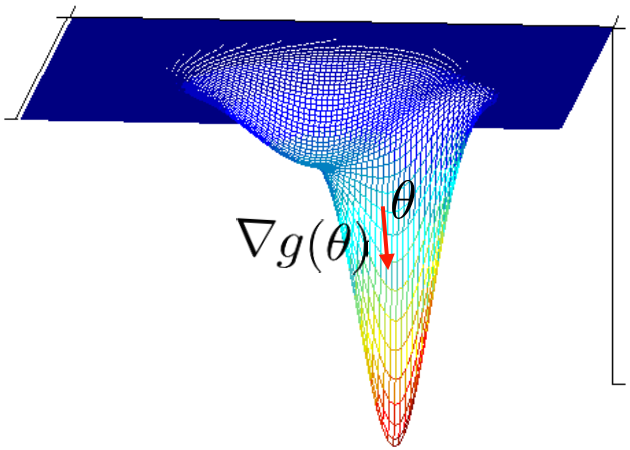
Convex and differentiable: gradient descent



Definition:

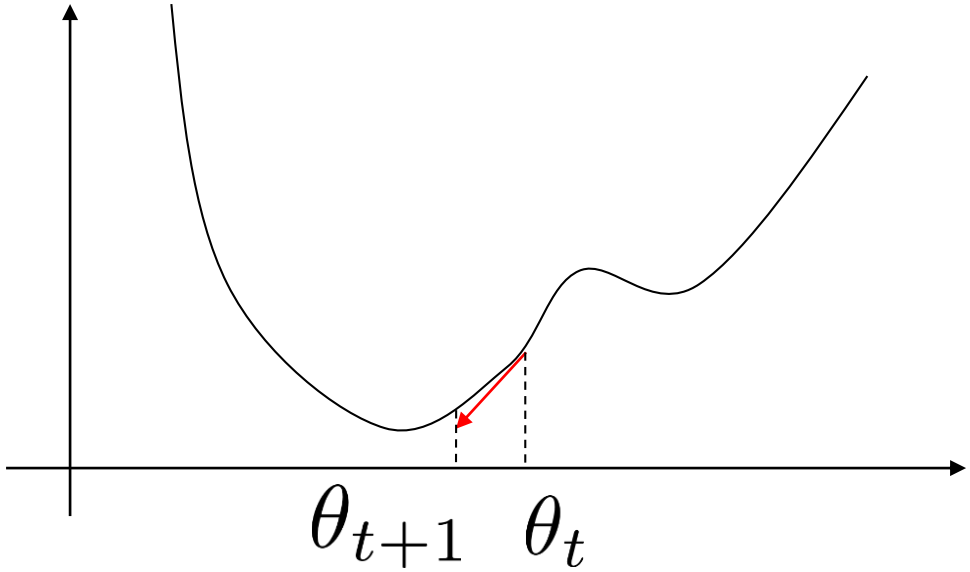
1. x^* is a globally optimal solution for $\theta^* \in \Omega$ and $g(\theta^*) \leq g(\theta) \forall \theta \in \Omega$
2. $x^\star$ is a locally optimal solution if there is a neighborhood $\mathcal{N}$ around x such that $\theta^\star \in \Omega$, $g(\theta^\star) \leq g(\theta)$, $\forall \theta \in \mathcal{N} \cap \Omega$.

Gradient: $\nabla g(\theta) = [\frac{\partial g}{\partial \theta_i}]_{i=1,\ldots,d}$ (its a vector!)



Hessian: $\nabla^2 g(\theta) = [\frac{\partial^2 g}{\partial \theta_i \theta_j}]_{i,j=1,\ldots,d}$ (its a matrix!)

Gradient descent (ascent)



Gradient Method as a Line Search Method $\rightarrow$ Descent Direction

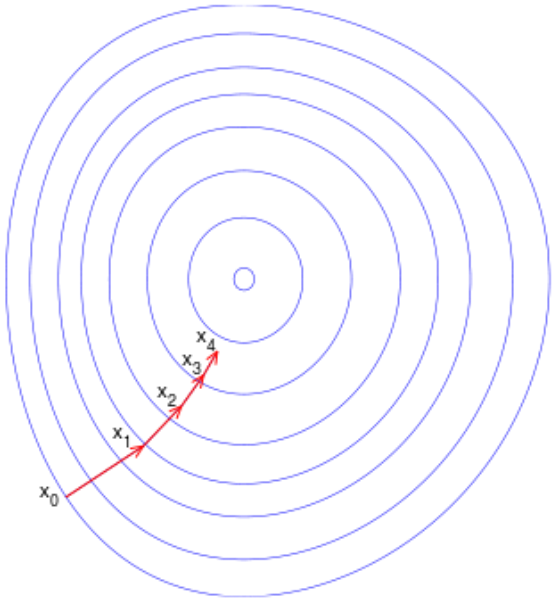
(a) Pick a direction v

(b) Pick a step size λ

(c) $\theta_{t+1} = \theta_t - \lambda \times v$ such that function decreases

(d) Repeat

Gradient descent



$$\theta_{t+1} \leftarrow \theta_t - \lambda_t \nabla g(\theta_t) \qquad \lambda_t : stepsize$$

Theorem:

For a continuous differentiable function f on a neighborhood of θ_0, if $v^T \nabla g(\theta) < 0$, then there exists $T > 0$ such that $g(\theta_0 + tv) < g(\theta_0) \forall t \in (0, T]$.

L1 Loss

$$S_{training} = \{(x_i, y_i), i = 1..n\} \quad y_i \in \mathcal{R}$$

Obtain/train: $f(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2$

$$W = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$$

$$W^* = \arg min_W \sum_{i=1}^{n} |\mathbf{x}_i^T \cdot W - y_i|$$

$$\mathbf{x}_i = \begin{pmatrix} 1 \\ x_i \\ x_i^2 \end{pmatrix}$$

$$\frac{\partial |f(w)|}{\partial w} = \begin{cases} \frac{\partial |f(w)|}{\partial w} & if\, f(w) \geq 0 \\ -\frac{\partial |f(w)|}{\partial w} & otherwise \end{cases}$$

$$= sign(f(w)) \cdot \frac{\partial f(w)}{\partial w}$$

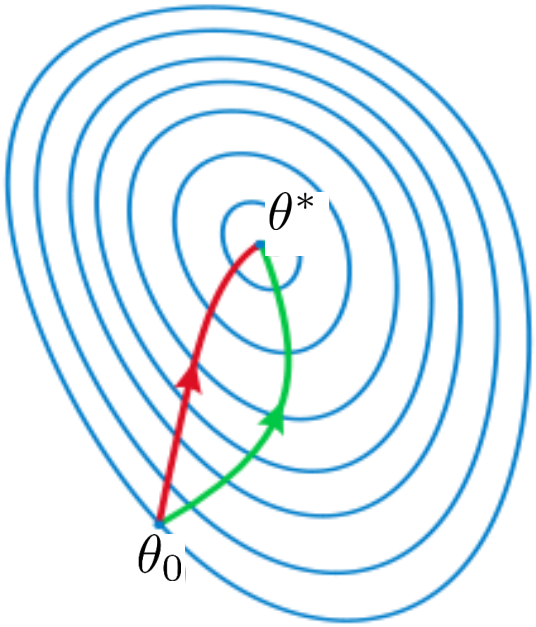
$$L(W) = \sum_{i=1}^{n} |\mathbf{x}_i^T W - y_i|$$

$$\frac{\partial L(W)}{\partial W} = \sum_{i=1}^{n} sign(\mathbf{x}_i^T W - y_i) \cdot \mathbf{x}_i$$

$$W_{t+1} = W_t - \lambda_t \frac{\partial L(W)}{\partial W}$$

Newton's method

$$\theta_{t+1} \leftarrow \theta_t - [\nabla^2 g(\theta_t)]^{-1} \nabla g(\theta_t)$$



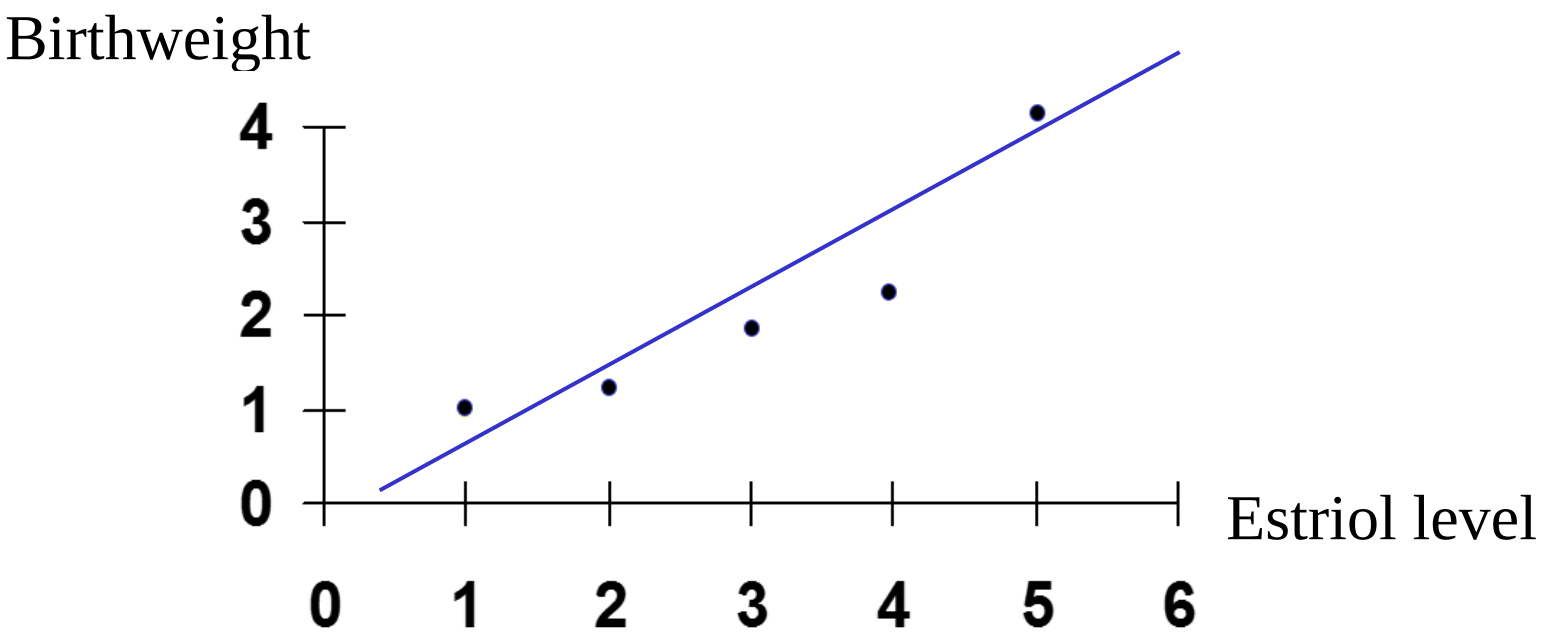
A comparison of gradient descent (green) and Newton's method (red) for minimizing a function (with small step sizes). Newton's method uses curvature information to take a more direct route.

An example

- What is the relationship between Mother's Estriol level & Birthweight using the following data?

Estriol (mg/24h)	Birthweight (g/1000)
1	1
3	1.9
2	1.05
5	4.1
4	2.1

Scatterplot
Birthweight vs. Estriol level



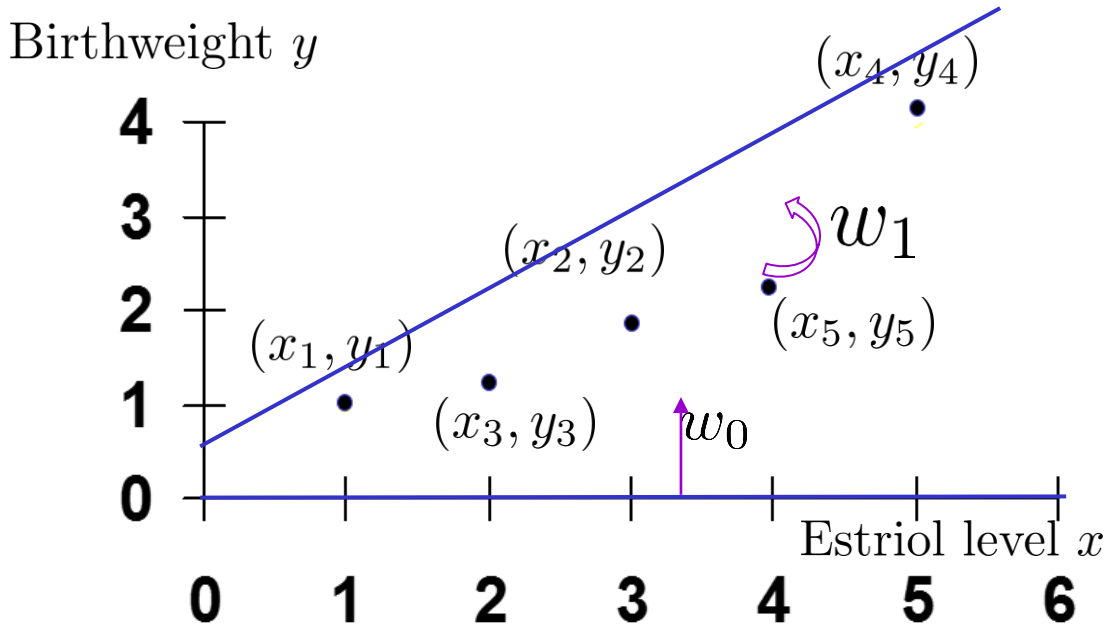
Univariate linear regression: the simplest case

$$S_{training} = \{(x_i, y_i), i = 1..n\}$$

Input: $x_i,\ x_i \in R$

Model parameter: $\mathbf{w} = (w_0, w_1),\ w_0, w_1 \in R$

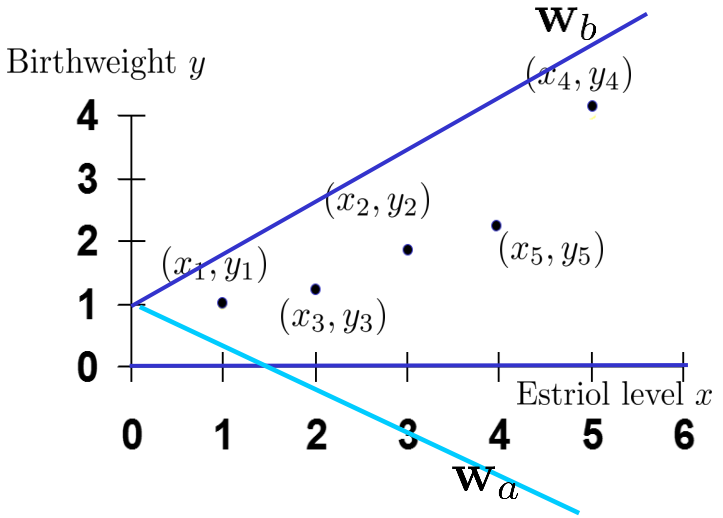
Output: $y_i = w_0 + w_1 x, \quad y_i \in R$



An example

- **Training data**

Estriol (mg/24h)	Birthweight (g/1000)
1	1
3	1.9
2	1.05
5	4.1
4	2.1



$$S_{training} = \{(x_i, y_i), i = 1..n\} = \{(1,1), (3,1.9), (2,1.05), (5,4.1), (4,2.1)\}$$

If $\mathbf{w}_a = (w_0, w_1) = (1, -0.5)$

$$e_{training}(\mathbf{w}_a) = \frac{1}{5}\sum_{i=1}^{5}(y_i - (1 - 0.5x_i))^2 = 9.62$$

If $\mathbf{w}_b = (w_0, w_1) = (1, 0.5)$

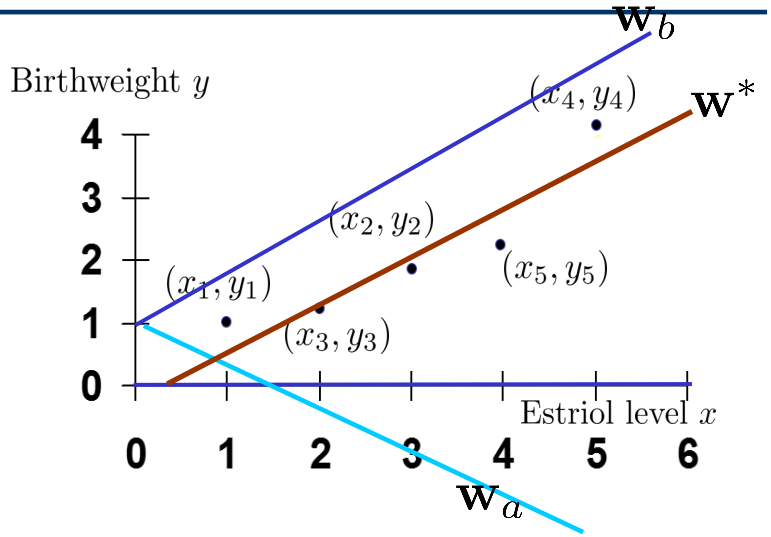
$$e_{training}(\mathbf{w}_b) = \frac{1}{5}\sum_{i=1}^{5}(y_i - (1 + 0.5x_i))^2 = 0.54$$

$\mathbf{w}_b$ is better than $\mathbf{w}_a$ since $e_{training}(\mathbf{w}_b) < e_{training}(\mathbf{w}_a)$

An example

- **Training data**

Estriol (mg/24h)	Birthweight (g/1000)
1	1
3	1.9
2	1.05
5	4.1
4	2.1



$$S_{training} = \{(x_i, y_i), i = 1..n\} = \{(1,1), (3,1.9), (2,1.05), (5,4.1), (4,2.1)\}$$

Our goal is find the optimal $\mathbf{w}^* = (w_0, w_1)^*$:

$$\mathbf{w}^* = \arg_{\mathbf{w}} \min e_{training}(\mathbf{w}) = \frac{1}{5} \sum_{i=1}^{5} (y_i - (w_0 + w_1 * x_i))^2$$

Least square estimation

$$S_{training} = \{(x_i, y_i), i = 1..n\} \qquad y_i \in \mathcal{R}$$

Obtain/train: $f(x, \mathbf{w}) = w_0 + w_1 x$

$$W = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}$$

$$W^* = \arg min_W \sum_i (\mathbf{x}_i^T \cdot W - y_i)^2$$

$$\mathbf{x}_i = \begin{pmatrix} 1 \\ x_i \end{pmatrix}$$

Let: $\mathbf{X} = (\mathbf{x}_i, ..., \mathbf{x}_n)^T \qquad Y = (y_1, ..., y_n)^T$

$$W^* = \arg min_W = \arg min_W g(W) = (X \cdot W - Y)^T (X \cdot W - Y)$$

$$g(W) = W^T X^T X W - W^T X^T Y - Y^T X W + Y^T Y$$

$$\frac{dg(W)}{dW} = 2X^T X W - 2X^T Y = 0$$

$$W^* = (X^T X)^{-1} X^T Y$$

In matlab, you can simply call $X \setminus Y$ or $pinv(X) * Y$

Linear regression algorithm

- Step 1: Select a model.
  - $y = w_0 + w_1 x$ (For us right now – not always)
- Step 2: Put data into matrix form.
- Step 3: Solve for the weights (**w**) in Matlab.
- Step 4: Report the model.
- Step 5: Visualize the model using model predictions.

Linear Regression Algorithm

- Step 1: Select a model.
  - $y = w_0 + w_1 x$ (For us right now – not always)
- Step 2: Put data into matrix form.
  - $\mathbf{y} = A\mathbf{w}$;
- Step 3: Solve for the weights ($\mathbf{w}$).
- Step 4: Report the model.
- Step 5: Visualize the model using model predictions.

Step 2: Put Data Into Matrix Form

$$S_{training} = \{(x_i, y_i), i = 1..n\} = \{(1,1), (3,1.9), (2,1.05), (5,4.1), (4,2.1)\}$$

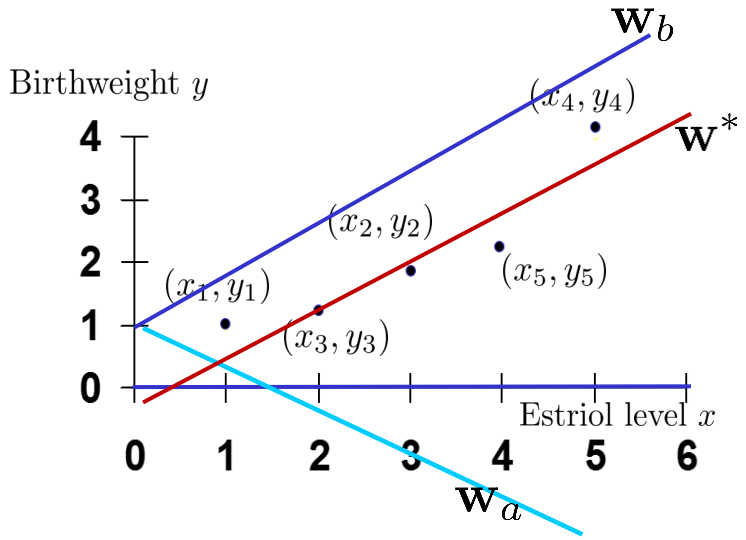
Basic Equations

$$y = w_0 + w_1 x$$

$$1 = w_0 + w_1 \times 1$$
$$1.9 = w_0 + w_1 \times 3$$
$$1.05 = w_0 + w_1 \times 2$$
$$4.1 = w_0 + w_1 \times 5$$
$$2.1 = w_0 + w_1 \times 4$$

Matrix Form

$$Y = X \cdot W$$



$$\begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix} = \begin{pmatrix} 1,1 \\ 1,3 \\ 1,2 \\ 1,5 \\ 1,4 \end{pmatrix} \cdot \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}$$

In matlab, you can simply call $X \setminus Y$ or $pinv(X) * Y$

$$\begin{pmatrix} -0.145 \\ 0.725 \end{pmatrix}$$

$$e_{training}(\mathbf{w}_a) = 9.62$$
$$e_{training}(\mathbf{w}_b) = 0.54$$
$$e_{training}(\mathbf{w}^*) = 0.21$$

Linear Regression Algorithm

- Step 1: Select a model.
  - $y = w_0 + w_1 x$ (For us right now – not always)
- Step 2: Put data into matrix form.
  - **y** = A**w;**
- Step 3: Solve for the weights (**w**) in Matlab.
  - **w = A\y;** or **w = pinv(A)*y;**
- Step 4: Report the model.

- Step 5: Visualize the model using model predictions.

Linear Regression Algorithm

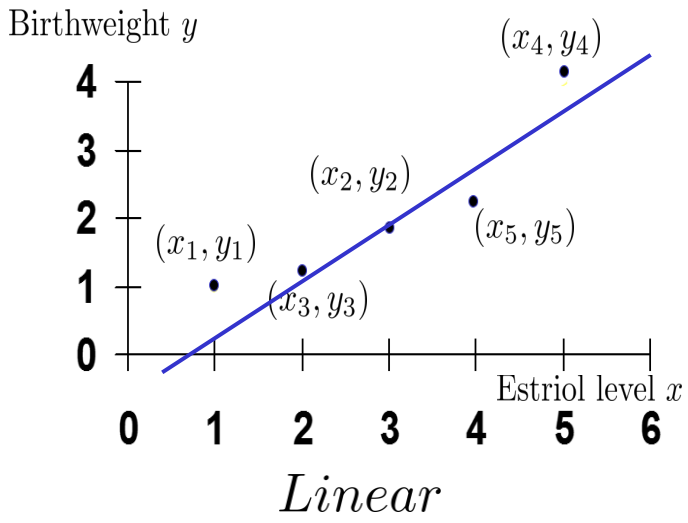
- Step 1: Select a model.
  - $y = w_0 + w_1 x$ (For us right now – not always)
- Step 2: Put data into matrix form.
  - **y** = A**w;**
- Step 3: Solve for the weights (**w**) in Matlab.
  - **w = A\y;** or **w = pinv(A)*y;**
- Step 4: Report the model.
  - Fill in the **w** values in our model (step 1) and write out again.
  - **y = -0.145 + 0.725x**
- Step 5: Visualize the model using model predictions.

Univariate linear regression

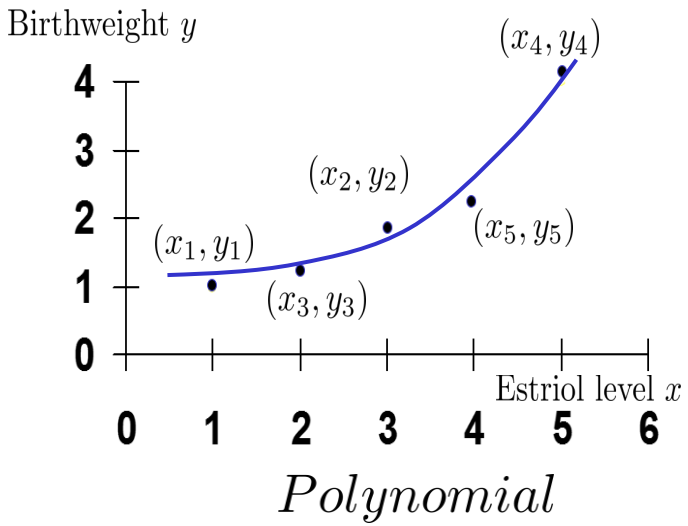
Why is this supervised? How is it labeled?

We are treating the variable we are plotting on the y-axis as a label – as a truth that we want to estimate and predict for new data.

There are different types of regressors, leading to different levels of complexity.



Linear



Polynomial

More Complex Linear Combinations

- Polynomial Linear Regression

- Multivariate Linear Regression

Polynomial Linear Regression

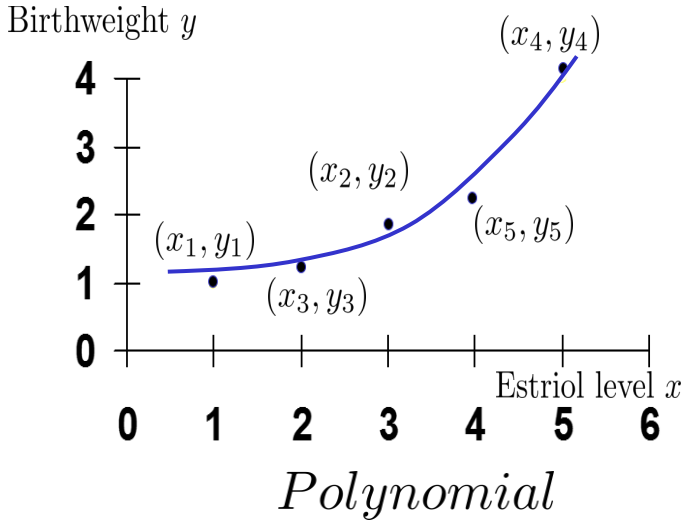
$$S_{training} = \{(x_i, y_i), i = 1..n\}$$

Input: x, $x \in R$

Model parameter: $\mathbf{w} = (w_0, w_1, ..., w_d)$, $w_i \in R$

Output: $y = w_0 + w_1 x_1 + w_2 x_1^2 + ... + w_q x_m^q$

The combination of terms has your input variable raised to an addition power for each subsequent term.



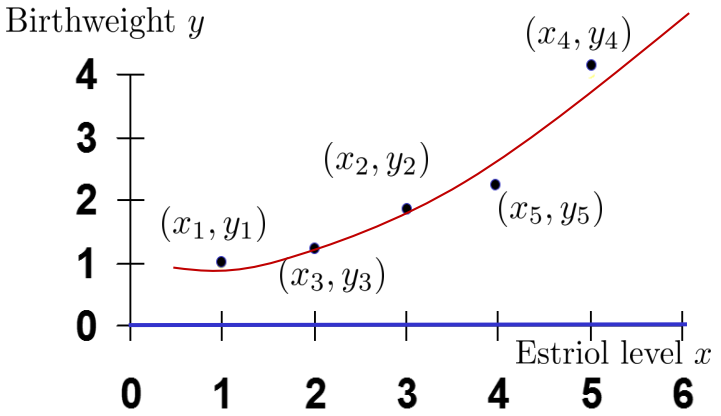
Put Data Into Matrix Form

$$S_{training} = \{(x_i, y_i), i = 1..n\} = \{(1,1),(3,1.9),(2,1.05),(5,4.1),(4,2.1)\}$$

Basic Equations **Matrix Form**

$$y = w_0 + w_1 x + w_2 x^2 \qquad Y = X \cdot W$$

$$1 = w_0 + w_1 \times 1 + w_2 \times 1$$
$$1.9 = w_0 + w_1 \times 3 + w_2 \times 9$$
$$1.05 = w_0 + w_1 \times 2 + w_2 \times 4$$
$$4.1 = w_0 + w_1 \times 5 + w_2 \times 25$$
$$2.1 = w_0 + w_1 \times 4 + +w_2 \times 16$$



In matlab, you can simply call
$X \setminus Y$ or $pinv(X) * Y$

$$\begin{pmatrix} 1.48 \\ -0.67 \\ 0.2321 \end{pmatrix}$$

$$Y = X \cdot W$$

$$\begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix} = \begin{pmatrix} 1,1,1 \\ 1,3,9 \\ 1,2,4 \\ 1,5,25 \\ 1,4,16 \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$$

$$e_{training}(\mathbf{w}^*) = 0.063$$

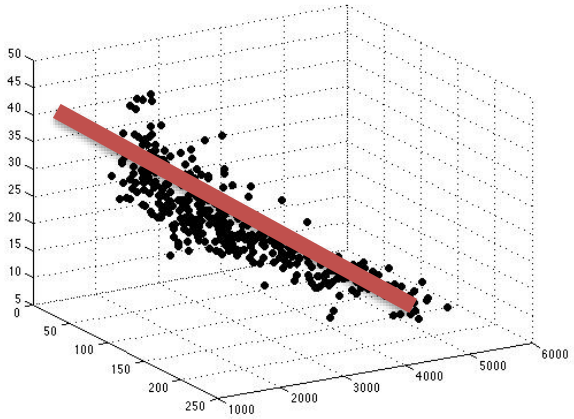
$$e_{training}(\mathbf{w}_a) = 9.62$$

$$e_{training}(\mathbf{w}_b) = 0.54$$

$$e_{training}(\mathbf{w}^*) = 0.21$$

previous linear model

Multi-variate Linear Regression



Input: $\mathbf{x} = (x_1, ..., x_m),\ x_i \in R$

Model parameter: $\mathbf{w} = (w_0, w_1, ..., w_m),\ w_i \in R$

Output: $y = w_0 + w_1 x_1 + w_2 x_2 + ... + w_m x_m$

We apply do the least squre fitting method again.

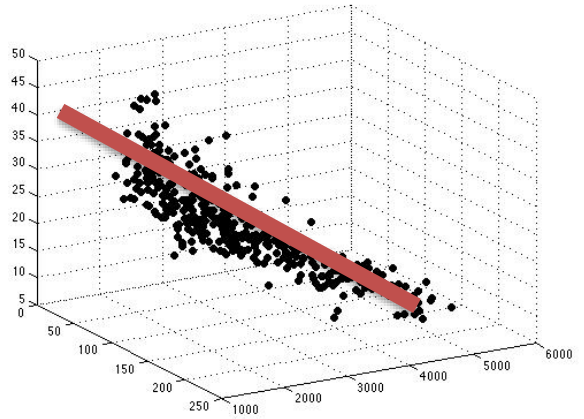
Step 2: Put Data Into Matrix Form

$$S_{training} = \{((x_{i1}, x_{i2}), y_i), i = 1..n\}$$
$$= \{((1, 0.5), 1), ((3, 0.9), 1.9), ((2, 1.0), 1.05), ((5, 6.7), 4.1), ((4, 2.5), 2.1)\}$$

Basic Equations **Matrix Form**

$$y = w_0 + w_1 x_1 + w_2 x_2 \qquad Y = X \cdot W$$

$$1 = w_0 + w_1 \times 1 + w_2 \times 0.5$$
$$1.9 = w_0 + w_1 \times 3 + w_2 \times 0.9$$
$$1.05 = w_0 + w_1 \times 2 + w_2 \times 1.0$$
$$4.1 = w_0 + w_1 \times 5 + w_2 \times 6.7$$
$$2.1 = w_0 + w_1 \times 4 + w_2 \times 2.5$$

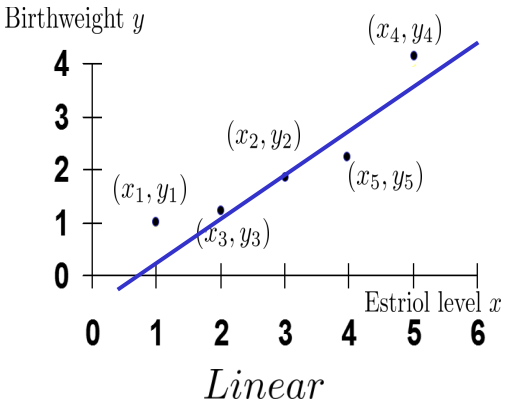


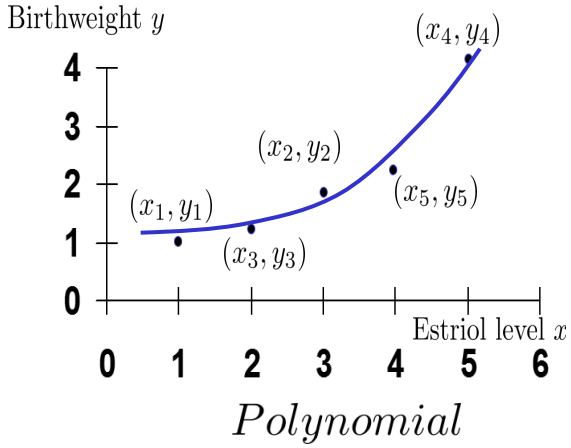
In matlab, you can simply call $X \setminus Y$ or $pinv(X) * Y$

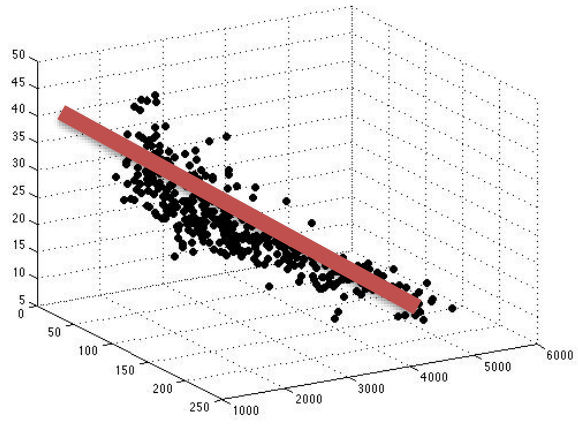
$$Y = X \cdot W$$

$$\begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix} = \begin{pmatrix} 1, 1, 0.5 \\ 1, 3, 0.9 \\ 1, 2, 1.0 \\ 1, 5, 6.7 \\ 1, 4, 2.5 \end{pmatrix} \cdot \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} \qquad \begin{pmatrix} 0.482 \\ 0.2552 \\ 0.338 \end{pmatrix}$$

Select your model







$e_{training}(\mathbf{w}^*) = 0.21$

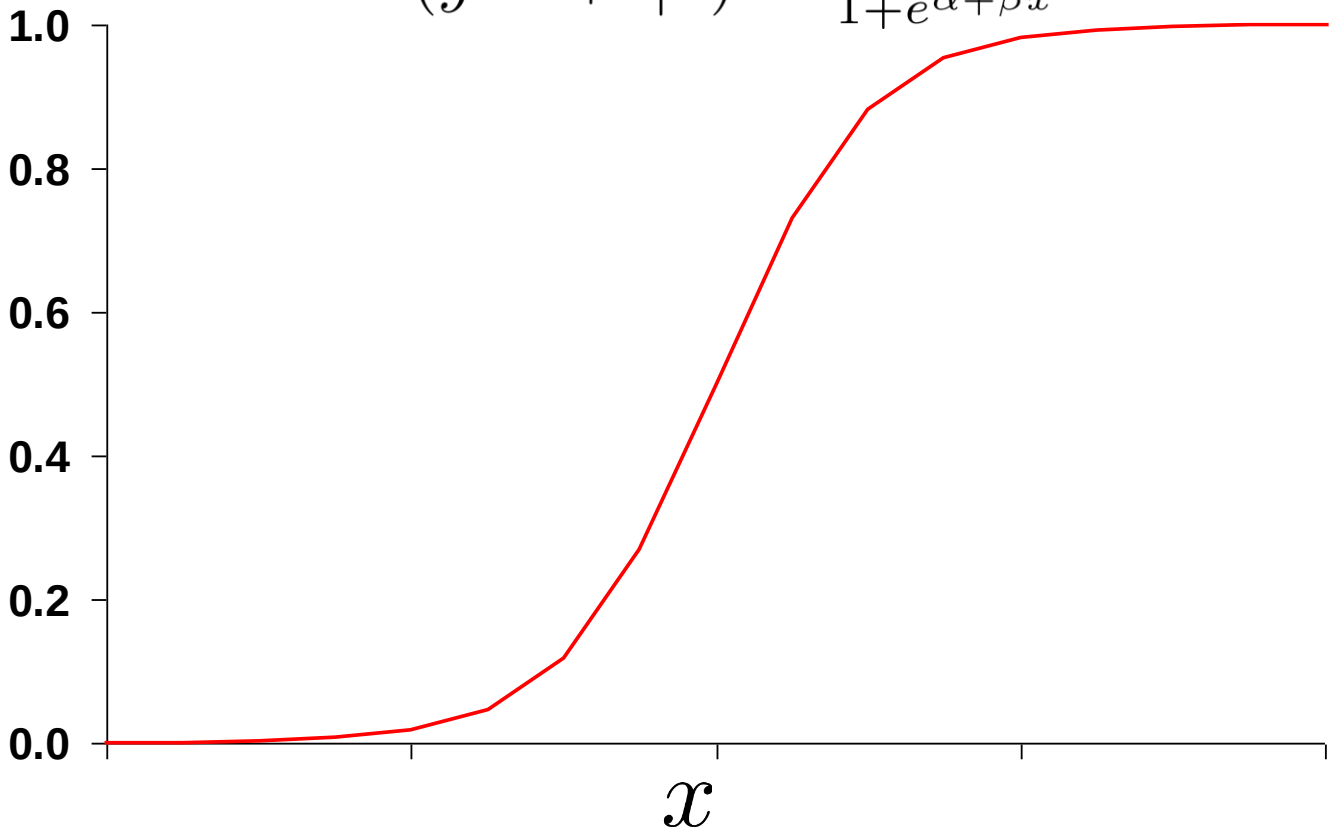
$e_{training}(\mathbf{w}^*) = 0.063$

$e_{training}(\mathbf{w}^*) = 0.052$

The logistic function

Probability of being positive

$$P(y = +1|x) = \frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}}$$



The logistic function

$$p(y=1|x) = \frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}} = \frac{1}{1+e^{-(\alpha+\beta x)}}$$

$$\ln\left(\frac{p(y=1|x)}{1-p(y|x)}\right) = \alpha + \beta x \qquad \text{logit of } p(y=1|x)$$

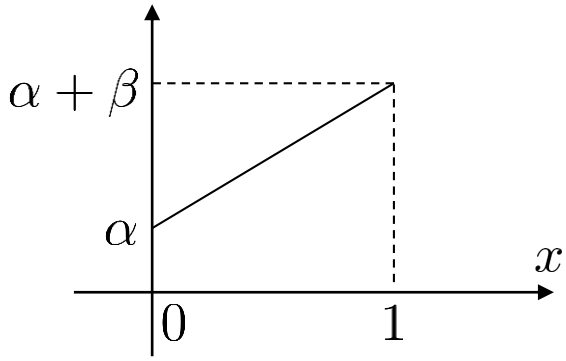
Advantages of the logit

- Simple transformation of P(y|x)
- Linear relationship with x
- Can be continuous (Logit between - ∞ to + ∞)
- Known binomial distribution (P between 0 and 1)
- Directly related to the notion of odds of disease

$$\ln\left(\frac{p}{1-p}\right) = \alpha + \beta x \qquad \frac{p}{1-p} = e^{\alpha+\beta x}$$

The logistic function

Logit of $P(y = +1|x)$: $\ln(\frac{P(y=+1|x)}{1-P(y=+1|x)}) = \alpha + \beta x$



Advantages of the logit:

1. Simple transformation of $p(y|x)$
2. Linear relationship with x
3. Can be continuous (Logit between $-\infty$ to $+\infty$)
4. Known binomial distribution (P between 0 and 1)

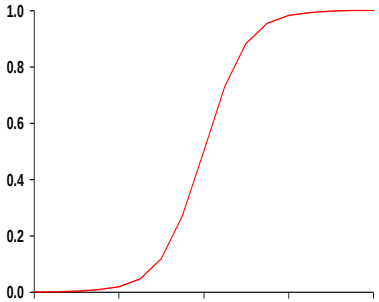
The logistic function

Logit of $P(y=+1|x)$: $\ln(\frac{P(y=+1|x)}{1-P(y=+1|x)}) = \alpha + \beta x$

$$P(y=+1|x) = \frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}} = \frac{1}{1+e^{-(\alpha+\beta x)}}$$

$$P(y=-1|x) = \frac{1}{1+e^{\alpha+\beta x}}$$

$$P(y=+1|x) + P(y=-1|x) = 1$$



An agnostic notation: $P(y|x) = \frac{1}{1+e^{-y(\alpha+\beta x)}}$

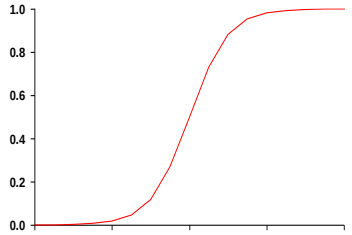
Training a logistic regression classifier

$$S_{training} = \{(x_i, y_i), i = 1..n\} \qquad x_i \in R, i = 1..n \quad y_i \in \{-1, +1\}, i = 1..n$$

$$S_{training} = \{(-1.1, -1), (3.2, +1), (2.5, -1), (5.0, +1), (4.3, +1)\}$$

Train a classifier $f(x)$:

If $f(x)$ is a logistic regression classifier:



$$f(x) = \begin{cases} +1 & if \ \frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}} \geq 0.5 \\ -1 & otherwise \end{cases}$$

$$e_{training} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}(y_i \neq f(x_i))$$

Training a logistic regression classifier

$S_{training} = \{(-1.1, -1), (3.2, +1), (2.5, -1), (5.0, +1), (4.3, +1)\}$

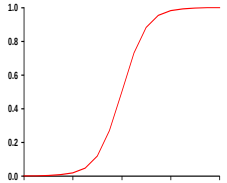
$x_i \in R, i = 1..n$

$y_i \in \{-1, +1\}, i = 1..n$

Train a logistic regression classifier $f(x)$:

$$P(y = +1|x) = \frac{1}{1+e^{-(\alpha+\beta x)}}$$

$$f(x) = \begin{cases} +1 & if \ \frac{1}{1+e^{-(\alpha+\beta x)}} \geq 0.5 \\ -1 & otherwise \end{cases}$$



$$(\alpha, \beta)^* = \arg\max_{(\alpha,\beta)} \prod_{i=1}^{n} [P(y_i = +1|x_i)]^{0.5+y_i/2} \times [P(y_i = -1|x_i)]^{0.5-y_i/2}$$

$$P(y_i|x_i) = \frac{1}{1+e^{-y_i(\alpha+\beta x_i)}}$$

$$(\alpha, \beta)^* = \arg\max_{(\alpha,\beta)} \prod_{i=1}^{n} \frac{1}{1+e^{-y_i(\alpha+\beta x_i)}}$$

$$(\alpha, \beta)^* = \arg\min_{(\alpha,\beta)} -\sum_{i=1}^{n} \ln(\frac{1}{1+e^{-y_i(\alpha+\beta x_i)}}) = \arg\min_{(\alpha,\beta)} \sum_{i=1}^{n} \ln(1+e^{-y_i(\alpha+\beta x_i)})$$

$$(\alpha, \beta)^* = \arg\min_{(\alpha,\beta)} [\ln(1+e^{(\alpha-1.1\beta)} + \ln(1+e^{-(\alpha+3.2\beta)} +$$

$$\ln(1+e^{(\alpha+2.5\beta)} + \ln(1+e^{-(\alpha+5.0\beta)} + \ln(1+e^{-(\alpha+4.3\beta)}]$$

Training a logistic regression classifier

We want to maximize the probability for a given training set. Suppose we have two points $\{(x_1, y_1 = +1), (x_2, y_2 = -1)\}$

$(\alpha, \beta)^* = \arg\max_{(\alpha,\beta)} \prod_{i=1}^{2} [P(y_i|x_i)]$

where

$P(y_i|x_i) = [P(y_i = +1|x_i)]^{0.5+y_i/2} \times [P(y_i = -1|x_i)]^{0.5-y_i/2}$

In the example given above:

$P(y_1|x_1) = [P(y_1 = +1|x_1)]^{0.5+1/2} \times [P(y_1 = -1|x_1)]^{0.5-1/2} = [P(y_1 = +1|x_1)] \times 1$, since $y_1 = +1$

$P(y_2|x_2) = [P(y_2 = +1|x_2)]^{0.5-1/2} \times [P(y_2 = -1|x_2)]^{0.5+1/2} = 1 \times [P(y_2 = -1|x_2)]$, since $y_2 = -1$

Training a logistic regression classifier

We want to maximize the probability for a given training set. Suppose we have two points $\{(x_1, y_1 = +1), (x_2, y_2 = -1)\}$

$(\alpha, \beta)^* = \arg\max_{(\alpha,\beta)} \prod_{i=1}^{2} [P(y_i|x_i)]$

where

$P(y_i|x_i) = [P(y_i = +1|x_i)]^{0.5+y_i/2} \times [P(y_i = -1|x_i)]^{0.5-y_i/2}$

We then have the form:

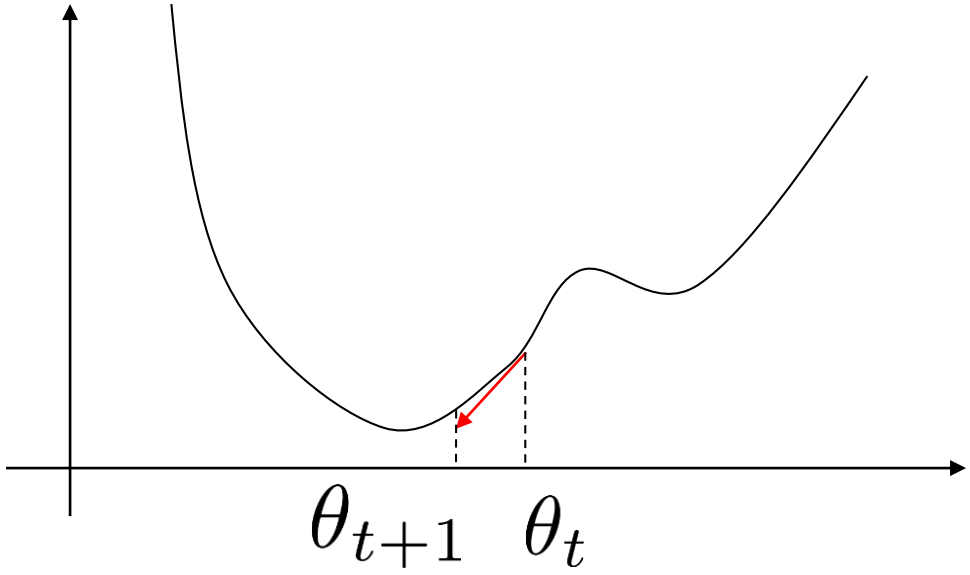
$(\alpha, \beta)^* = \arg\max_{(\alpha,\beta)} \prod_{i=1}^{n} [P(y_i = +1|x_i)]^{0.5+y_i/2} \times [P(y_i = -1|x_i)]^{0.5-y_i/2}$,

which can be simplified to:

$(\alpha, \beta)^* = \arg\max_{(\alpha,\beta)} \prod_{i=1}^{n} \frac{1}{1+e^{-y_i(\alpha+\beta x_i)}}$

if we have: $P(y|x) = \frac{1}{1+e^{-y(\alpha+\beta x)}}$

Gradient descent (ascent)



Gradient Descent Direction

(a) Pick a direction $\nabla g(x_t)$

(b) Pick a step size λ_t

(c) $\theta_{t+1} = \theta_t - \lambda_t \times \nabla g(\theta_t)$ such that function decreases

(d) Repeat

Training a logistic regression classifier

$$P(y_i|x_i) = \frac{1}{1+e^{-y_i(\alpha+\beta x_i)}}$$

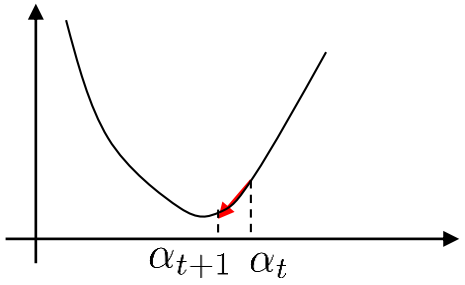
$$x_i \in R, i = 1..n$$

$$y_i \in \{-1, +1\}, i = 1..n$$

Train a logistic regression classifier $f(x)$:

$$(\alpha, \beta)^* = \arg\min_{(\alpha,\beta)} g(\alpha, \beta)$$

$$g(\alpha, \beta) = \sum_{i=1}^{n} \ln(1 + e^{-y_i(\alpha+\beta x_i)})$$



$$\nabla_\alpha g(\alpha, \beta) = \sum_i \frac{-y_i e^{-y_i(\alpha+\beta x_i)}}{1+e^{-y_i(\alpha+\beta x_i)}} = \sum_i -y_i(1 - P(y_i|x_i))$$

$$\nabla_\beta g(\alpha, \beta) = \sum_i \frac{-y_i x_i e^{-y_i(\alpha+\beta x_i)}}{1+e^{-y_i(\alpha+\beta x_i)}} = \sum_i -y_i x_i(1 - P(y_i|x_i))$$

$$\alpha_{t+1} = \alpha_t - \lambda_t \times \nabla_\alpha g(\alpha_t, \beta_t)$$

$$\beta_{t+1} = \beta_t - \lambda_t \times \nabla_\beta g(\alpha_t, \beta_t)$$

Sometimes we also use {0, 1} for y

$S_{training} = \{(-1.1, 0), (3.2, 1), (2.5, 0), (5.0, 1), (4.3, 1)\}$

$x_i \in R, i = 1..n$

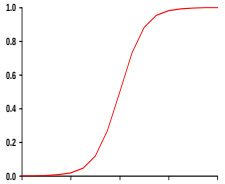
$y_i \in \{0, 1\}, i = 1..n$

Train a logistic regression classifier $f(x)$:

$$P(y = 1|x) = \frac{1}{1+e^{-(\alpha+\beta x)}}$$

$$P(y = 0|x) = \frac{e^{-(\alpha+\beta x)}}{1+e^{-(\alpha+\beta x)}} = \frac{1}{1+e^{(\alpha+\beta x)}}$$

$$f(x) = \begin{cases} 1 & if \ \frac{1}{1+e^{-(\alpha+\beta x)}} \geq 0.5 \\ 0 & otherwise \end{cases}$$



$$(\alpha, \beta)^* = \arg\max_{(\alpha,\beta)} \prod_{i=1}^{n} [P(y_i = 1|x_i)]_i^y \times [P(y_i = 0|x_i)]^{1-y_i}$$

$$P(y_i|x_i) = \frac{1}{1+e^{-(2y_i-1)(\alpha+\beta x_i)}}$$

$$(\alpha, \beta)^* = \arg\max_{(\alpha,\beta)} \prod_{i=1}^{n} \frac{1}{1+e^{-(2y_i-1)(\alpha+\beta x_i)}}$$

Sometimes we also use {0, 1} for y

$S_{training} = \{(-1.1, 0), (3.2, 1), (2.5, 0), (5.0, 1), (4.3, 1)\}$

$x_i \in R, i = 1..n$

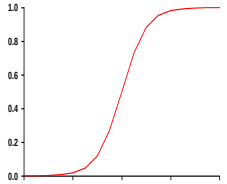
$y_i \in \{0, 1\}, i = 1..n$

Train a logistic regression classifier $f(x)$:

$$P(y = 1|x) = \frac{1}{1+e^{-(\alpha+\beta x)}}$$

$$P(y = 0|x) = \frac{e^{-(\alpha+\beta x)}}{1+e^{-(\alpha+\beta x)}} = \frac{1}{1+e^{(\alpha+\beta x)}}$$

$$f(x) = \begin{cases} 1 & if \ \frac{1}{1+e^{-(\alpha+\beta x)}} \geq 0.5 \\ 0 & otherwise \end{cases}$$



$$P(y_i|x_i) = \frac{1}{1+e^{-(2y_i-1)(\alpha+\beta x_i)}}$$

$$(\alpha, \beta)^* = \arg\max_{(\alpha,\beta)} \prod_{i=1}^{n} \frac{1}{1+e^{-(2y_i-1)(\alpha+\beta x_i)}}$$

$$(\alpha, \beta)^* = \arg\min_{(\alpha,\beta)} -\sum_{i=1}^{n} \ln\left(\frac{1}{1 + e^{-(2y_i-1)(\alpha+\beta x_i)}}\right) = \arg\min_{(\alpha,\beta)} \sum_{i=1}^{n} \ln(1 + e^{-(2y_i-1)(\alpha+\beta x_i)})$$

$$(\alpha, \beta)^* = \arg\min_{(\alpha,\beta)} [\ln(1 + e^{(\alpha-1.1\beta)} + \ln(1 + e^{-(\alpha+3.2\beta)} +$$

$$\ln(1 + e^{(\alpha+2.5\beta)} + \ln(1 + e^{-(\alpha+5.0\beta)} + \ln(1 + e^{-(\alpha+4.3\beta)}]$$

Multivariate input

$$P(y_i|\mathbf{x}_i) = \frac{1}{1+e^{-y_i(b+\ w^T\mathbf{x}_i)}}$$

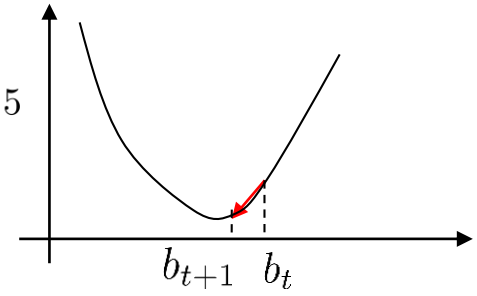
$$\mathbf{x}_i \in R^m, i = 1..n$$

$$y_i \in \{-1, +1\}, i = 1..n$$

Train a logistic regression classifier $f(\mathbf{x})$:

$$(b, \mathbf{w})^* = \arg\min_{(b,\mathbf{w})} g(b, \mathbf{w}) \quad f(\mathbf{x}) = \begin{cases} 1 & if\ \frac{1}{1+e^{-(b+\mathbf{w}^T\mathbf{x})}} \geq 0.5 \\ -1 & otherwise \end{cases}$$

$$g(b, \mathbf{w}) = \sum_{i=1}^{n} \ln(1 + e^{-y_i(b+\mathbf{w}^T\mathbf{x}_i)})$$



$$\nabla_b g(b, \mathbf{w}) = \sum_i \frac{-y_i e^{-y_i(b+\mathbf{w}^T\mathbf{x}_i)}}{1+e^{-y_i(b+\mathbf{w}^T\mathbf{x}_i)}} = \sum_i -y_i(1 - P(y_i|\mathbf{x}_i))$$

$$\nabla_{\mathbf{w}} g(b, \mathbf{w}) = \sum_i \frac{-y_i\mathbf{x}_i e^{-y_i(b+\mathbf{w}^T\mathbf{x}_i)}}{1+e^{-y_i(b+\mathbf{w}^T\mathbf{x}_i)}} = \sum_i -y_i\mathbf{x}_i(1 - P(y_i|\mathbf{x}_i))$$

$$b_{t+1} = b_t - \lambda_t \times \nabla_b g(b_t, \mathbf{w}_t)$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda_t \times \nabla_{\mathbf{w}} g(b_t, \mathbf{w}_t)$$

Multivariate input and y {0, 1}

$$P(y_i|\mathbf{x}_i) = \frac{1}{1+e^{-(2y_i-1)(b+\ w^T\mathbf{x}_i)}}$$

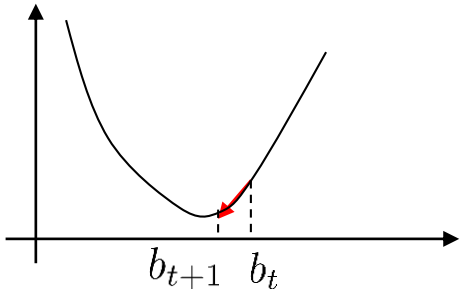
$$\mathbf{x}_i \in R^m, i = 1..n$$

$$y_i \in \{0, 1\}, i = 1..n$$

Train a logistic regression classifier $f(\mathbf{x})$:

$$(b, \mathbf{w})^* = \arg\min_{(b,\mathbf{w})} g(b, \mathbf{w}) \quad f(\mathbf{x}) = \begin{cases} 1 & if\ \frac{1}{1+e^{-(b+\mathbf{w}^T\mathbf{x})}} \geq 0.5 \\ 0 & otherwise \end{cases}$$

$$g(b, \mathbf{w}) = \sum_{i=1}^{n} \ln(1 + e^{-(2y_i-1)(b+\mathbf{w}^T\mathbf{x}_i)})$$



$$\nabla_b g(b, \mathbf{w}) = \sum_i \frac{-(2y_i-1)e^{-(2y_i-1)(b+\mathbf{w}^T\mathbf{x}_i)}}{1+e^{-(2y_i-1)(b+\mathbf{w}^T\mathbf{x}_i)}} = \sum_i -(2y_i - 1)(1 - P(y_i|\mathbf{x}_i))$$

$$\nabla_{\mathbf{w}} g(b, \mathbf{w}) = \sum_i \frac{-(2y_i-1)\mathbf{x}_i e^{-(2y_i-1)(b+\mathbf{w}^T\mathbf{x}_i)}}{1+e^{-(2y_i-1)(b+\mathbf{w}^T\mathbf{x}_i)}} = \sum_i -(2y_i - 1)\mathbf{x}_i(1 - P(y_i|\mathbf{x}_i))$$

$$b_{t+1} = b_t - \lambda_t \times \nabla_b g(b_t, \mathbf{w}_t)$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda_t \times \nabla_{\mathbf{w}} g(b_t, \mathbf{w}_t)$$

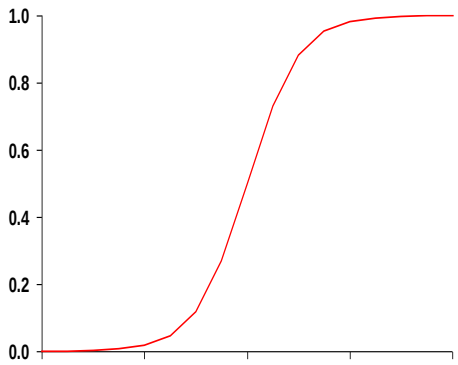
Logistic regression classifier

$$P(y_i|\mathbf{x}_i) = \frac{1}{1+e^{-y_i(b+\ \mathbf{w}^T\mathbf{x}_i)}}$$

$$\mathbf{x} \in R^m$$

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

$$f(\mathbf{x}) = \begin{cases} 1 & if\ \frac{1}{1+e^{-(b+\mathbf{w}^T\mathbf{x})}} \geq 0.5 \\ 0 & otherwise \end{cases}$$



Pros:

1. It is well-normalized.
2. Easy to turn into probability.
3. Easy to implement.

Cons:

1. Indirect loss function.
2. Dependent on good feature set.
3. Weak on feature selection.