COGS 181, Fall 2017

Neural Networks and Deep Learning

Lecture 4: Regression and Classification

ONNX

https://research.fb.com/facebook-and-microsoft-introduce-new-open-ecosystem-for-interchangeable-ai-frameworks

SEPTEMBER 7, 2017

Facebook and Microsoft introduce new open ecosystem for interchangeable AI frameworks

By: Joaquin Quinonero Candela

Facebook and Microsoft are today introducing Open Neural Network Exchange (ONNX) format, a standard for representing deep learning models that enables models to be transferred between frameworks. ONNX is the first step toward an open ecosystem where AI developers can easily move between state-of-the-art tools and choose the combination that is best for them.

When developing learning models, engineers and researchers have many AI frameworks to choose from. At the outset of a project, developers have to choose features and commit to a framework. Many times, the features chosen when

Related

PUBLICATION

ICCV 2017 / OCTOBER 22, 2017

Learning Visual N-Grams from Web Data

Ang Li, Allan Jabri, Armand Joulin,

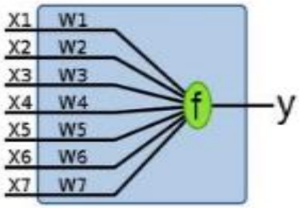
Three key things you are learning in this class:

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

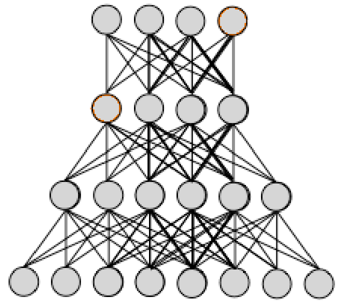
Evaluation: The ideal strategy is always to aim at your target directly (take non-stop flight as opposed to having multiple stops).

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

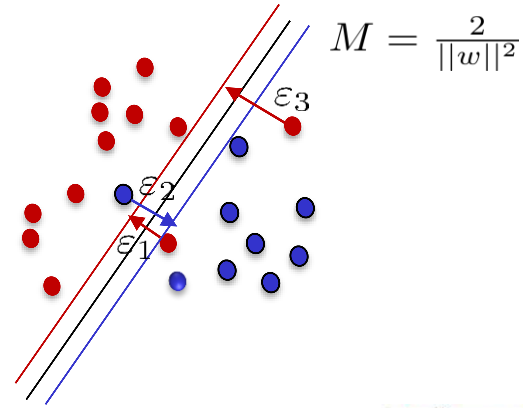
Representation (methods + features)



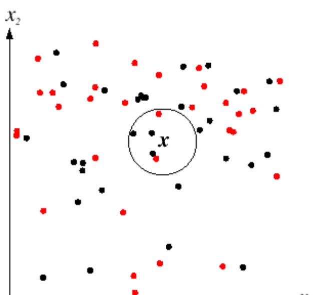
perceptron



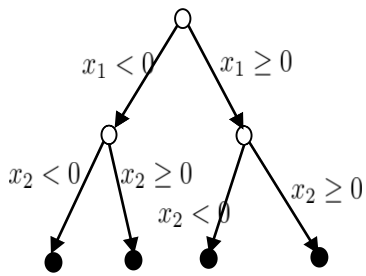
neural networks



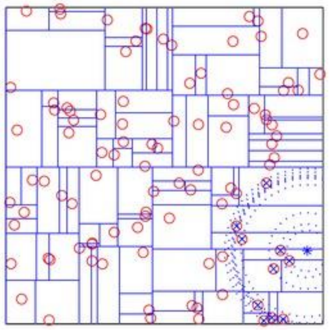
support vector machine



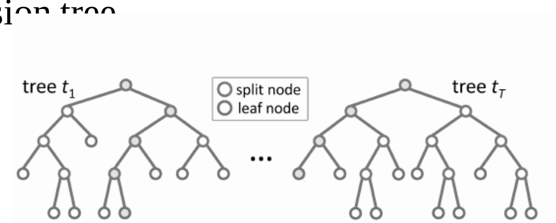
nearest neighborhood



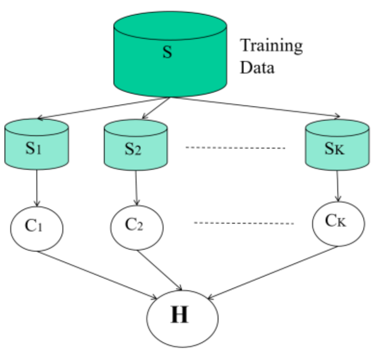
decision tree



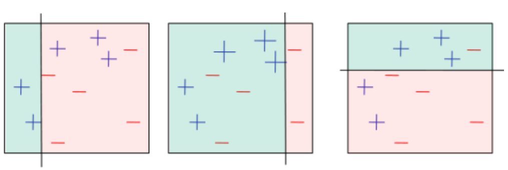
K-D tree



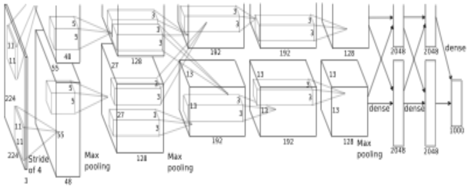
random forests



bagging



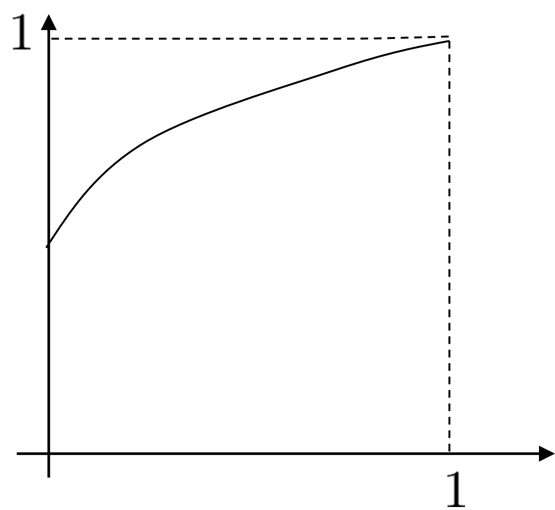
boosting



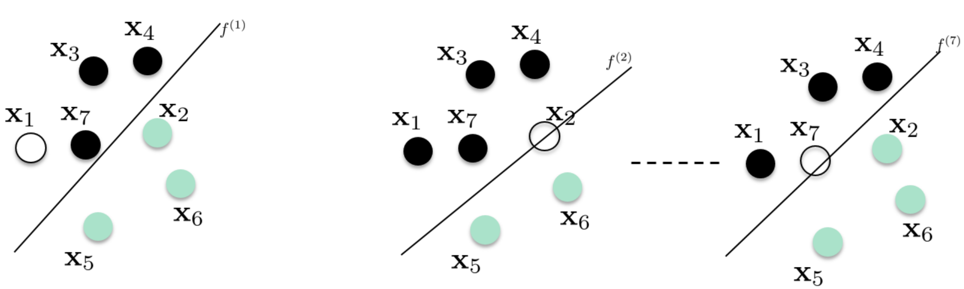
convolutional neural networks

Evaluation

		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−				

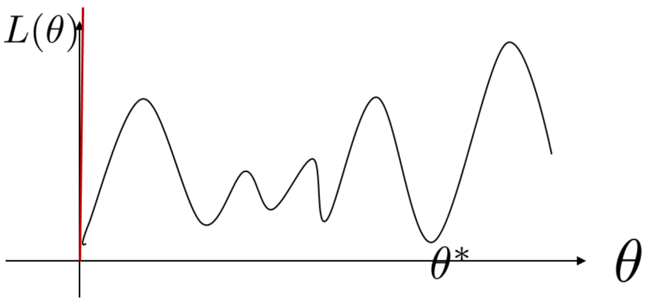


$$e_{testing} = e_{training} + generalization(f)$$

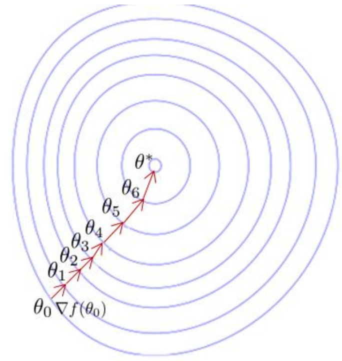


cross-validation

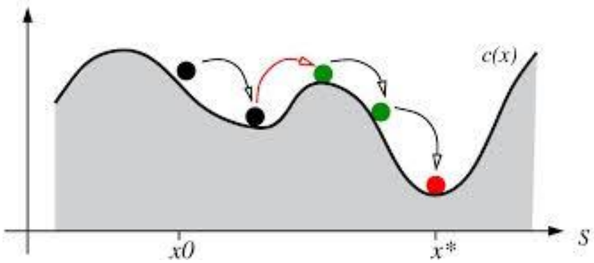
Optimization



exhaustive search



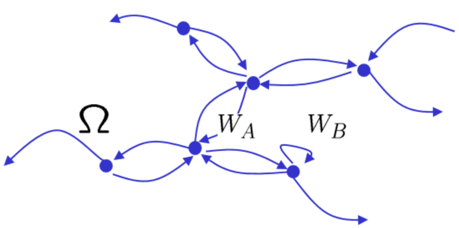
gradient descent



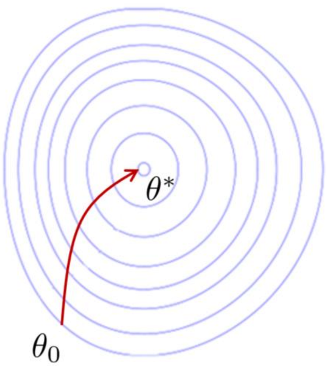
simulated annealing

Markov Chain: $MC = (\Omega, K, \nu)$

To design transition kernel: $p \bullet K = p$



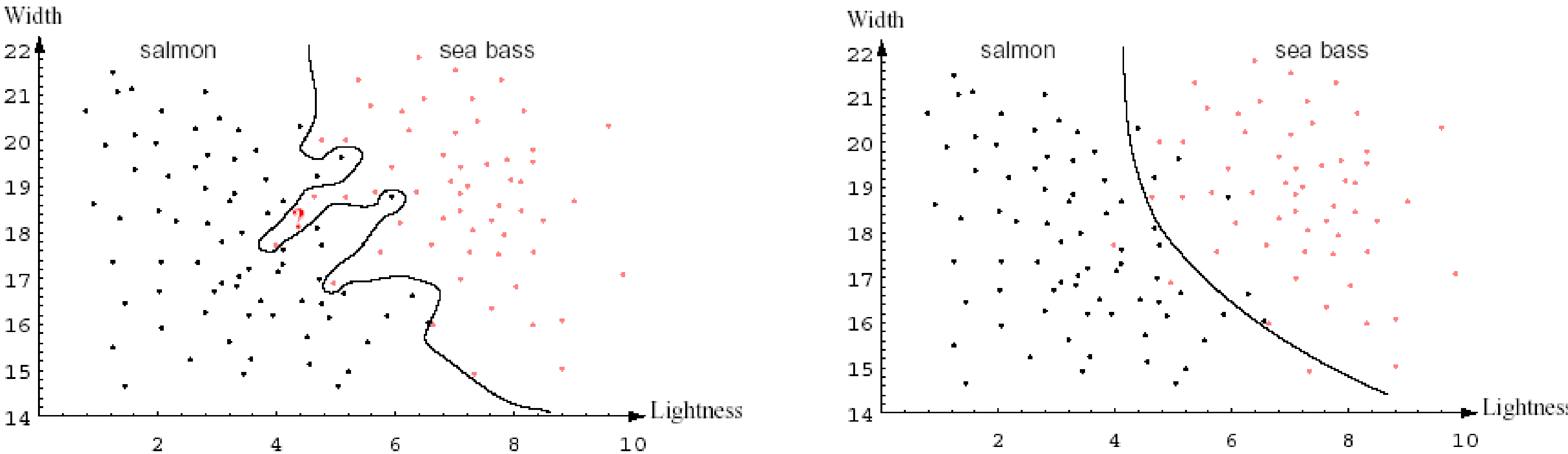
Markov chain Monte Carlo



Newton’s method

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:



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

$$\mathbf{1}(z) = \begin{cases} 1 & if\ z = TRUE \\ 0 & otherwise \end{cases}$$

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

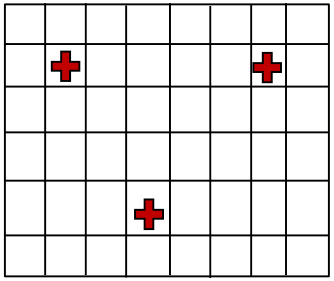
Error metric

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

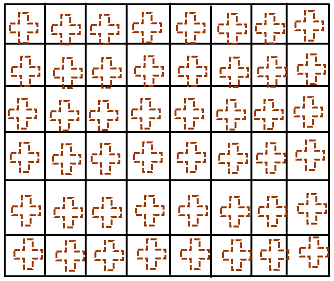
$$\text{recall} = P(hit|target) = \frac{\#(target, hit)}{\#(target)}$$

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

6×8 possible locations



3 targets to hit

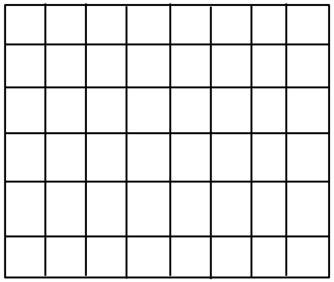


all hit

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

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

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

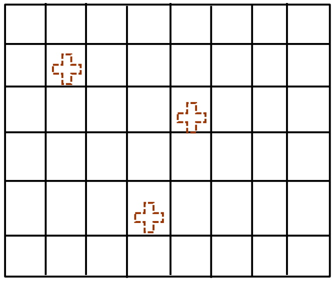


zero hit

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

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

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



miss one

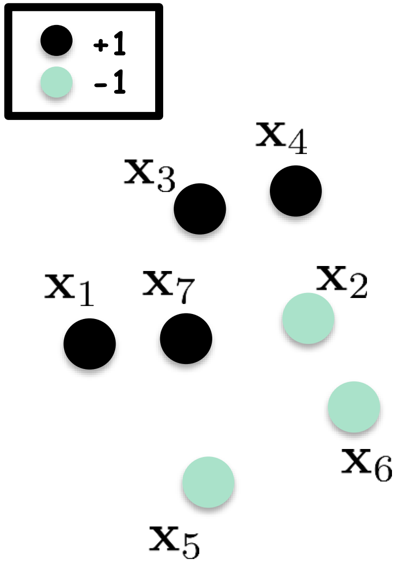
$\text{precision} = \frac{2}{3}$

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

$\text{F-value} \approx 0.667$

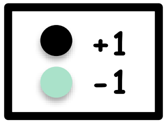
Cross-validation

(works for both regression and classification)

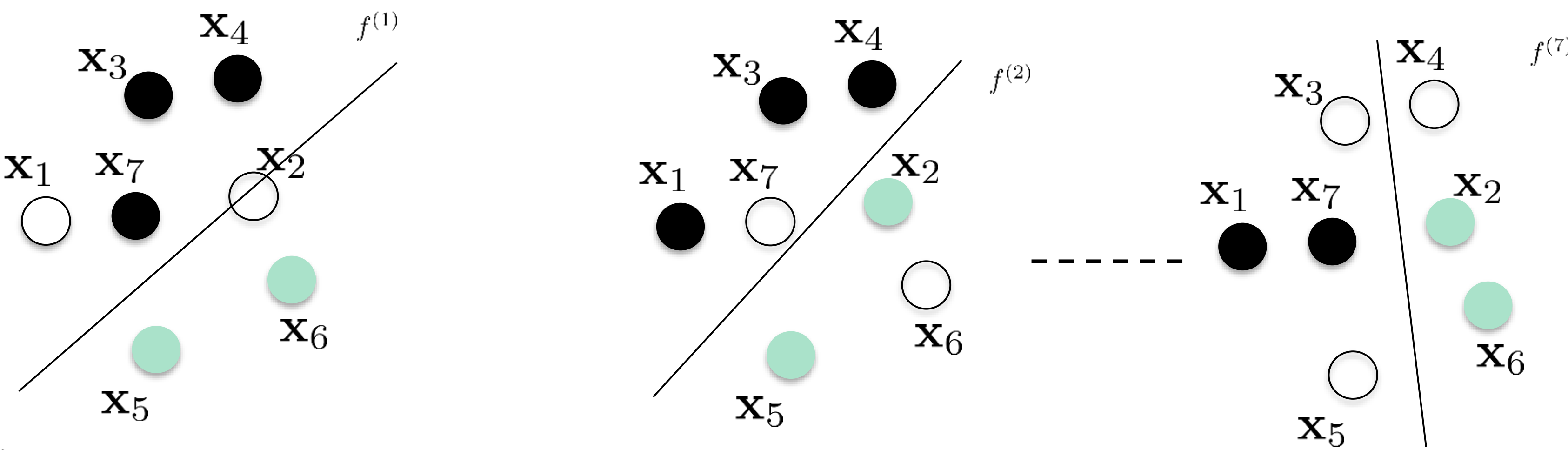


$$S_{training} =$$

$$\{(\mathbf{x}_1, +1), (\mathbf{x}_2, -1), (\mathbf{x}_3, +1), (\mathbf{x}_4, +1), (\mathbf{x}_5, -1), (\mathbf{x}_6, -1), (\mathbf{x}_7, +1)\}$$



Cross-validation
(works for both regression and classification)



$S^{(1)}_{training} = \{(\mathbf{x}_3, +1), (\mathbf{x}_4, +1), (\mathbf{x}_5, -1), (\mathbf{x}_6, -1), (\mathbf{x}_7, +1)\}$

Perform training to obtain $f^{(1)}$

$S^{(1)}_{testing} = \{(\mathbf{x}_1, +1), (\mathbf{x}_2, -1), \}$

$e_{testing}(f^{(1)}) = \frac{1}{2}[\mathbf{1}(y_1 \neq f^{(1)}(\mathbf{x}_1)) + \mathbf{1}(y_2 \neq f^{(1)}(\mathbf{x}_2))]$

$S^{(2)}_{training} = \{(\mathbf{x}_1, +1), (\mathbf{x}_3, +1), (\mathbf{x}_4, +1), (\mathbf{x}_5, -1), (\mathbf{x}_2, -1\}$

Perform training to obtain $f^{(2)}$

$S^{(2)}_{testing} = \{(\mathbf{x}_6, -1), (\mathbf{x}_7, +1)\}$

$e_{testing}(f^{(2)})$

$S^{(3)}_{training} = \{(\mathbf{x}_1, +1), (\mathbf{x}_2, -1), (\mathbf{x}_7, +1), (\mathbf{x}_6, -1)\}$

Perform training to obtain $f^{(k)}$

$S^{(k)}_{testing} = \{(\mathbf{x}_3, +1), (\mathbf{x}_4, +1), (\mathbf{x}_5, -1)\}$

$e_{testing}(f^{(k)})$

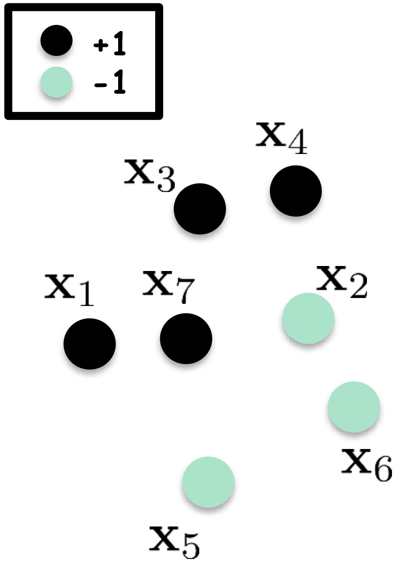
We compute the cross-validation error by

$\bar{e} = \frac{1}{k}\sum_i e_{testing}(f^{(i)})$

$var = \frac{1}{k}\sum_i (e_{testing}(f^{(i)}) - \bar{e})^2$

K-fold cross-validation

(works for both regression and classification)



$$S_{training} =$$

$$\{(\mathbf{x}_1, +1), (\mathbf{x}_2, -1), (\mathbf{x}_3, +1), (\mathbf{x}_4, +1), (\mathbf{x}_5, -1), (\mathbf{x}_6, -1), (\mathbf{x}_7, +1)\}$$

$$\Downarrow$$

$$S_{training} = \{Sub_1, Sub_2, Sub_3\}$$

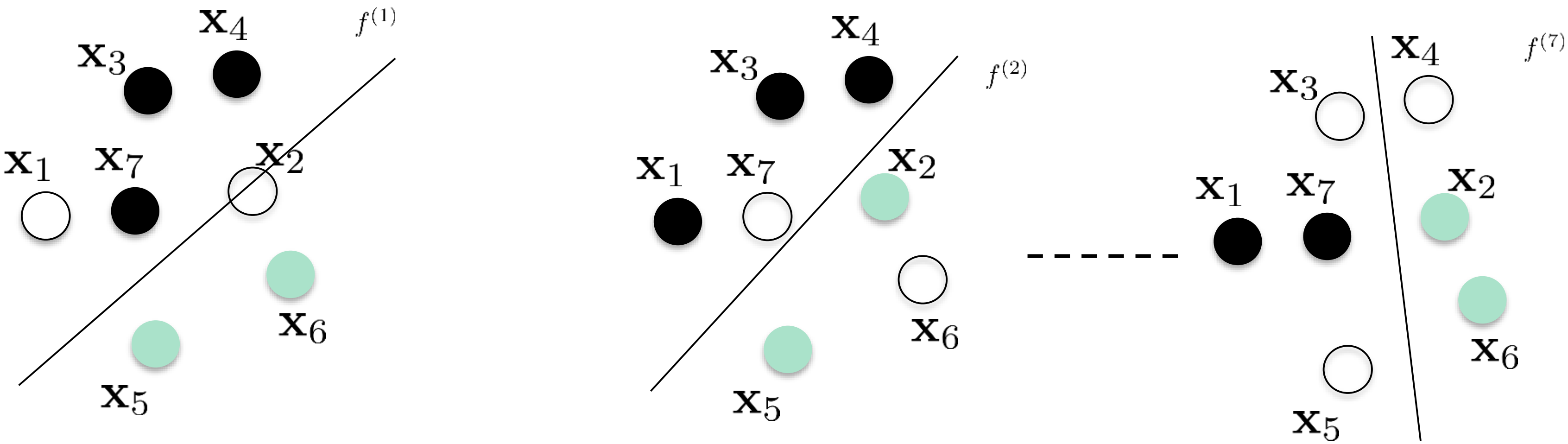
$$Sub_1 = \{(\mathbf{x}_1, +1), (\mathbf{x}_2, -1)\}$$

$$Sub_2 = \{(\mathbf{x}_6, -1), (\mathbf{x}_7, +1)\}$$

$$Sub_3 = \{(\mathbf{x}_3, +1), (\mathbf{x}_4, +1), (\mathbf{x}_5, -1)\}$$

For i=1 to k
Train classifier $f^{(i)}$ on a set that includes all the subsets but Sub_i.
Compute the testing error $e(f^{(i)})$ on Sub_i.
Fine-tune the model and hyper-parameter to minimize: $\bar{e} = \frac{1}{k} \sum_i e(f^{(i)})$.

K-fold Cross-validation
(works for both regression and classification)



We use $\bar{e} = \frac{1}{k}\sum_i e(f^{(i)})$ and var to decide on:

- Which model (linear or nonlinear ones) we should use?
- How to fine-tune the hyper-parameter?
- Have we collected enough data for training?
- Is our hypothesis valid statistically significant?

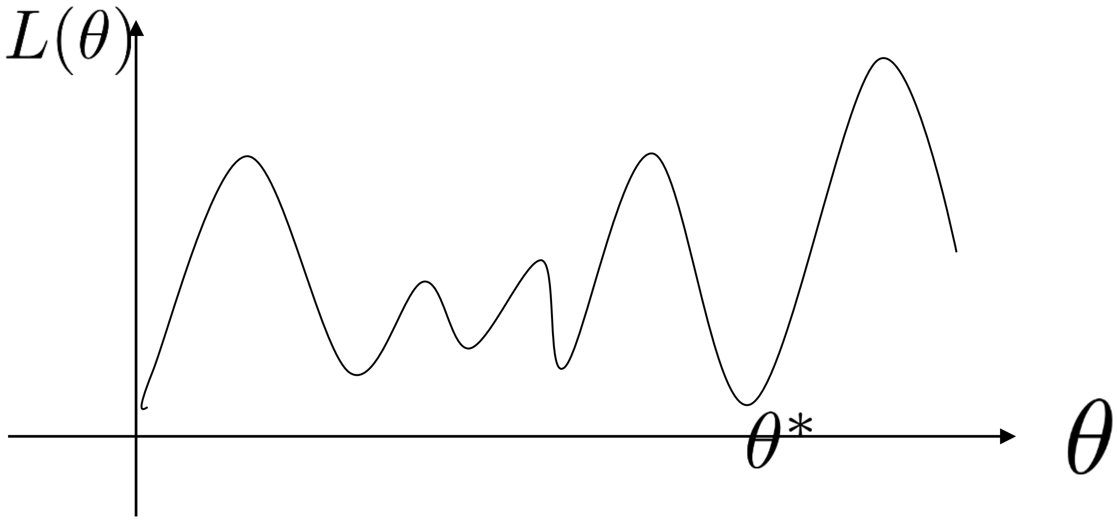
We are supposed to have small values for both $\bar{e}$ and var if our hypothsis is statistically significant.

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 $L(\theta)$

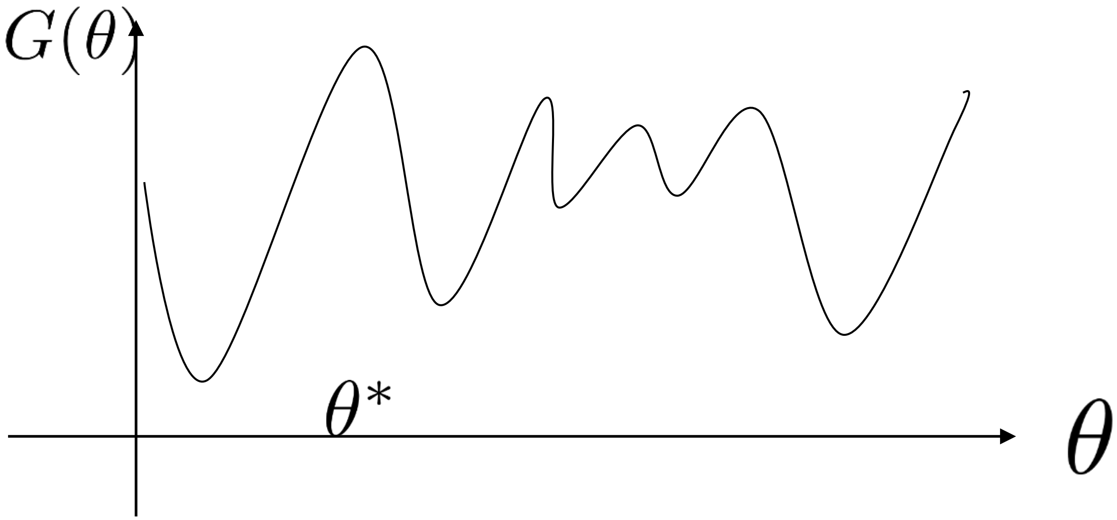


Optimization: argmax

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

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:

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

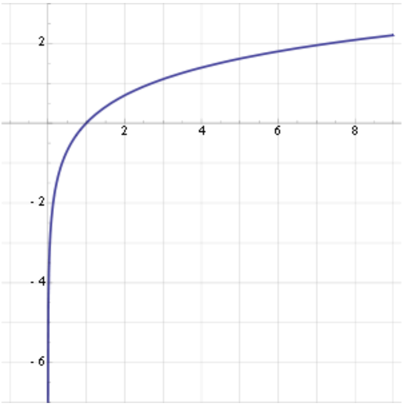
For example,

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 $X^T\theta$.

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:

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

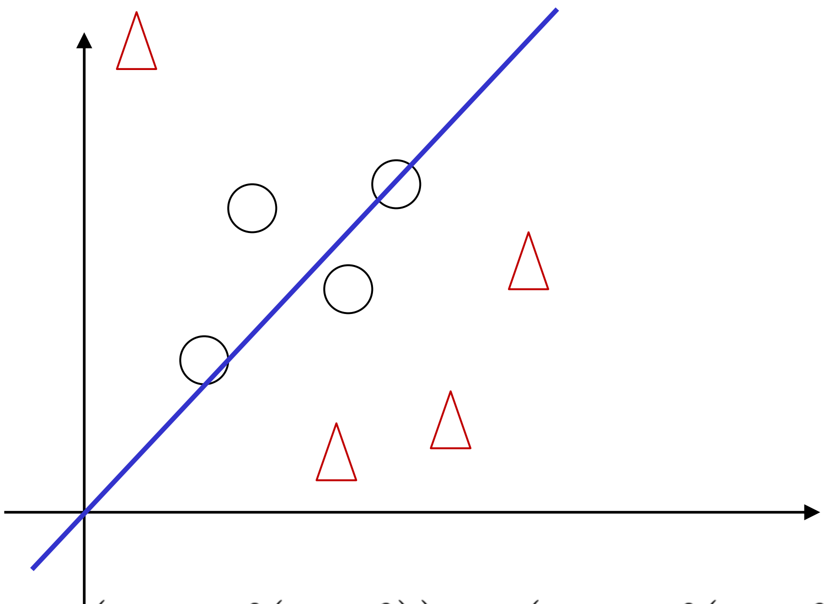
Estimation and optimization

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

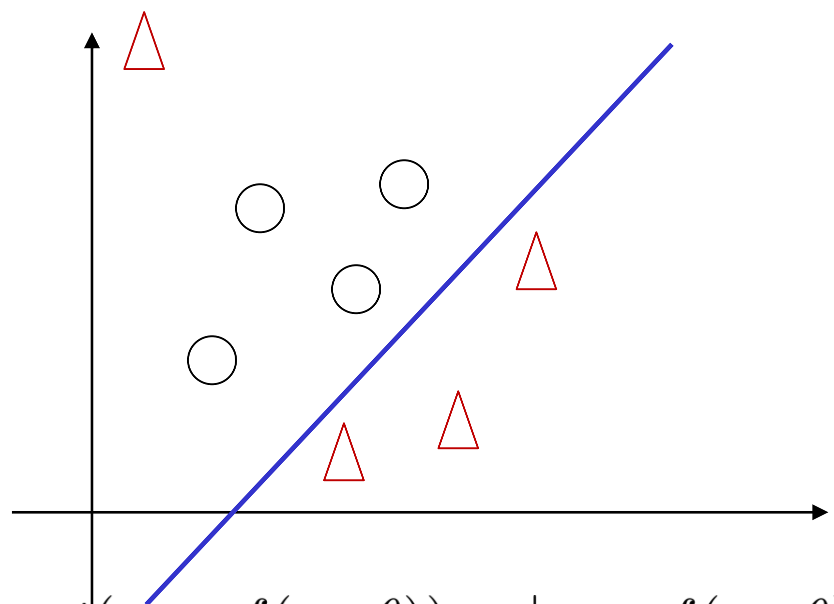
$$\mathbf{x} = [X_1, ..., X_k], X_1 \in \mathcal{R}, \quad \mathbf{x} \in \mathcal{R}^k \qquad \mathbf{x} \sim p(\mathbf{x})$$

A general form:

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



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



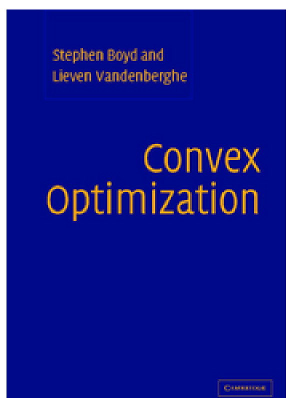
$$cost(y_i - f(\mathbf{x}_i; \theta)) = |y_i - f(\mathbf{x}_i; \theta)|$$

Estimation and optimization

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

Learning and estimation with convex functions:

http://stanford.edu/~boyd/

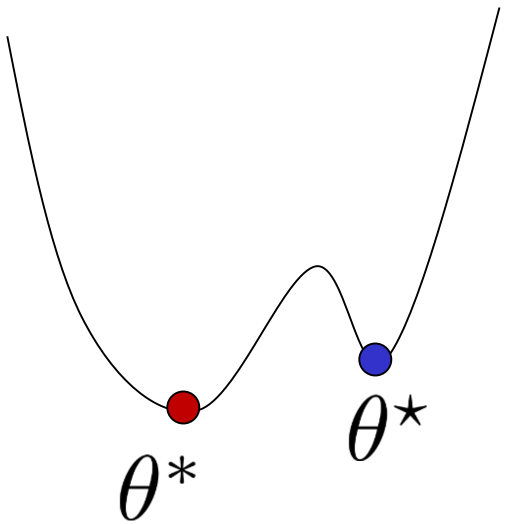


Convex Optimization
Stephen Boyd and Lieven Vandenberghe

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:

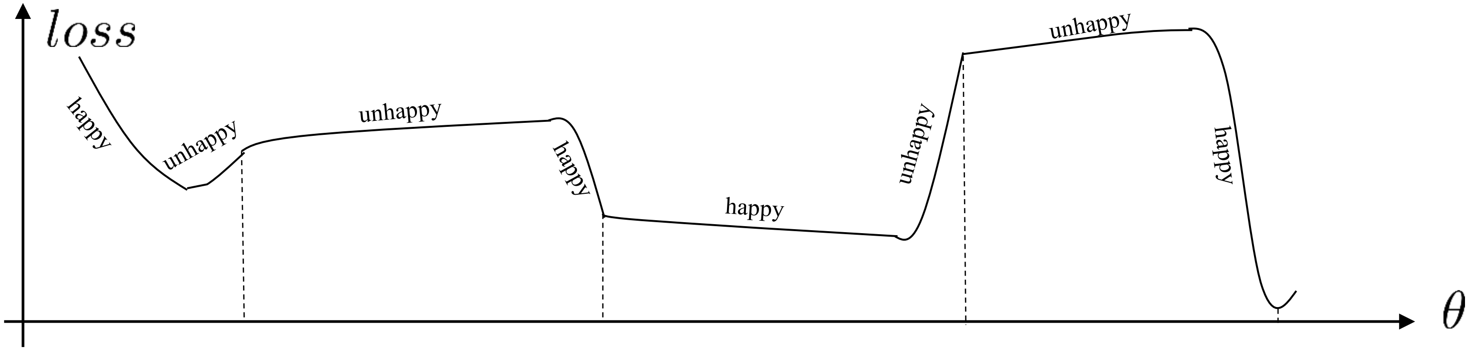
1. $\forall\theta$, check if $\theta \in \Omega$?
2. For $\forall\theta$, computing $g(\theta)$, $\nabla g(\theta)$, $\nabla^2 g(\theta)$.

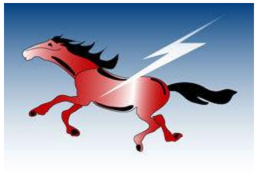
Definition:

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/





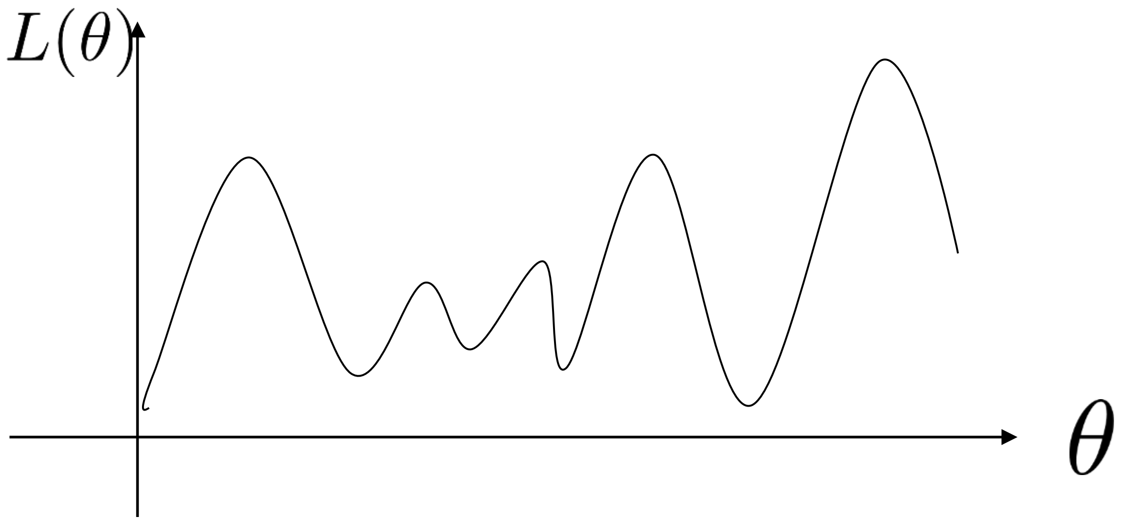
An analogy:
we want to find the lowest point in the figure.

http://menpiao.daiwoqu.com/

Learning

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

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



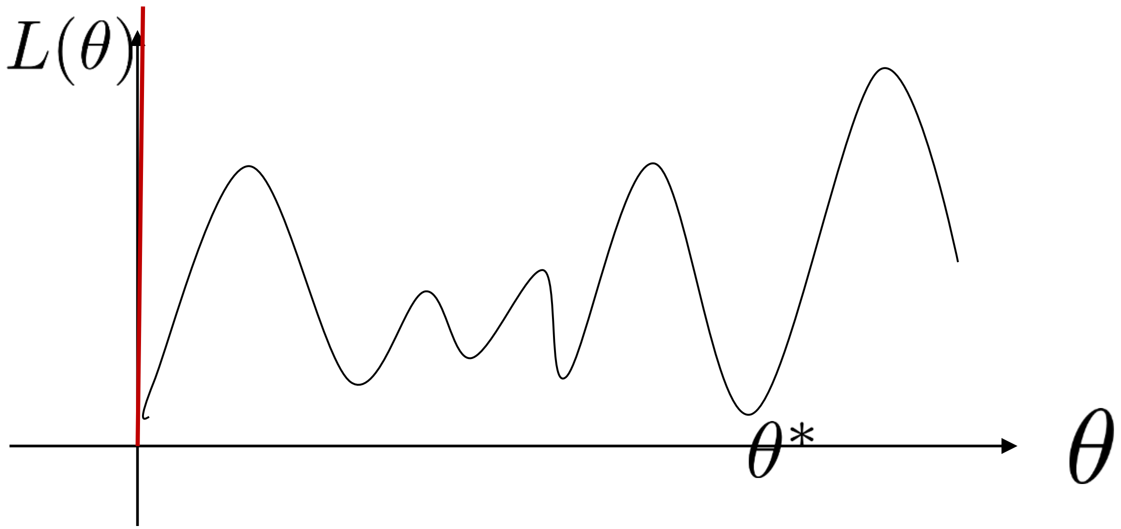
Given a θ, you can always evaluate the $L(\theta)$.

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.

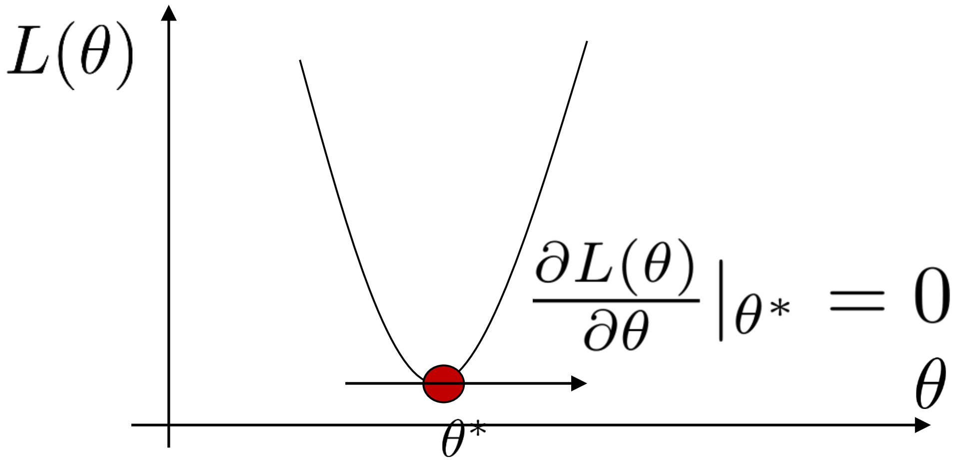
You are guaranteed to find the optimal θ^*,

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:

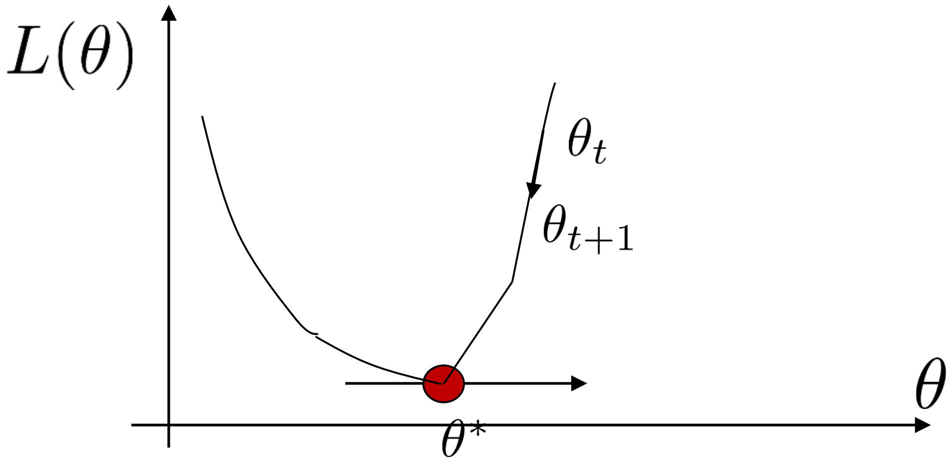
$$\text{Set: } \frac{\partial L(\theta)}{\partial \theta} = 0$$

θ^* 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:

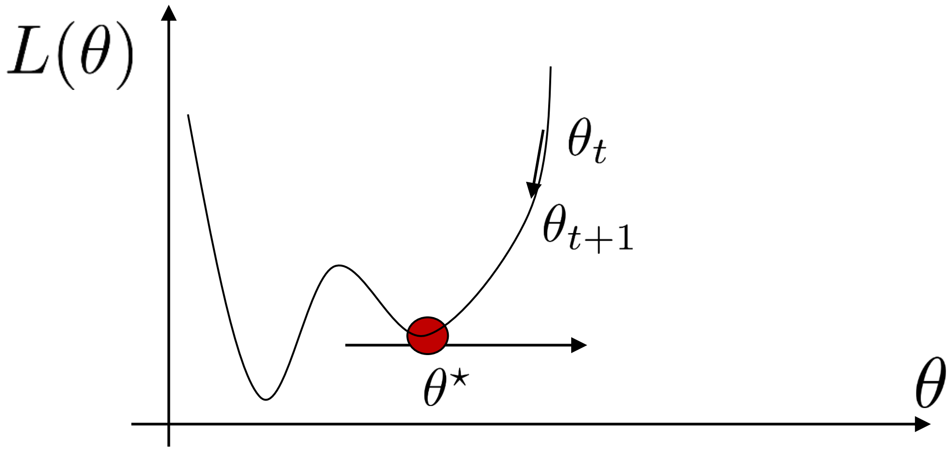
$$\text{Search for } \theta^* \text{ such that } \frac{\partial L(\theta)}{\partial \theta}\Big|_{\theta^*} = 0$$

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

Optimization: gradient descent

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

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



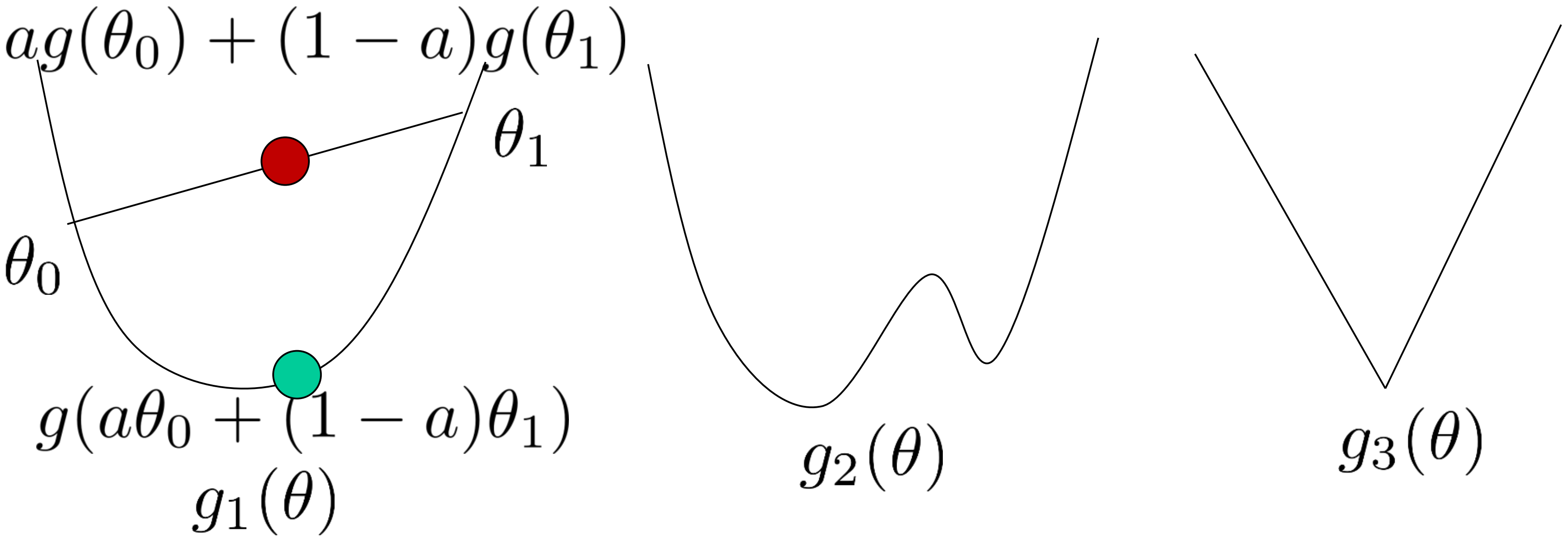
If you $L(\theta)$ is non-convex:

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

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

Convex functions

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



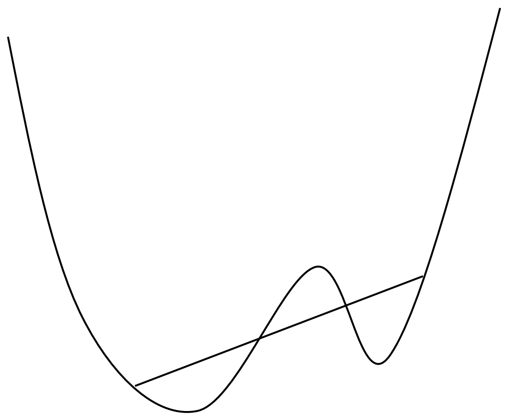
Definition:

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

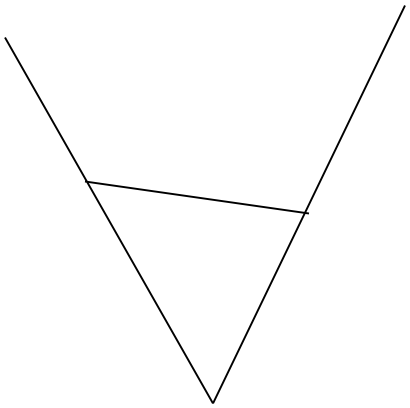
$$ag(\theta_0) + (1-a)g(\theta_1) \geq g(a\theta_0 + (1-a)\theta_1)$$

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):

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

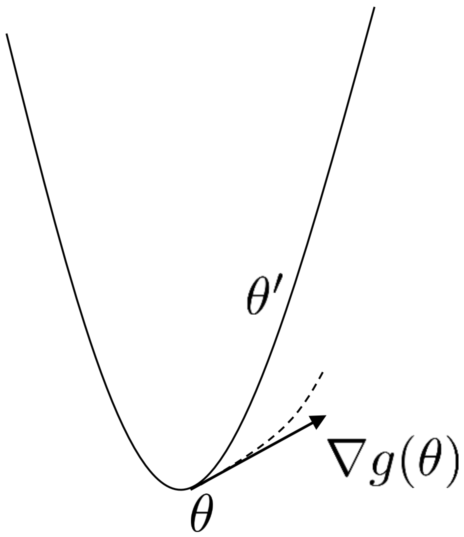
Strongly convex

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:

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



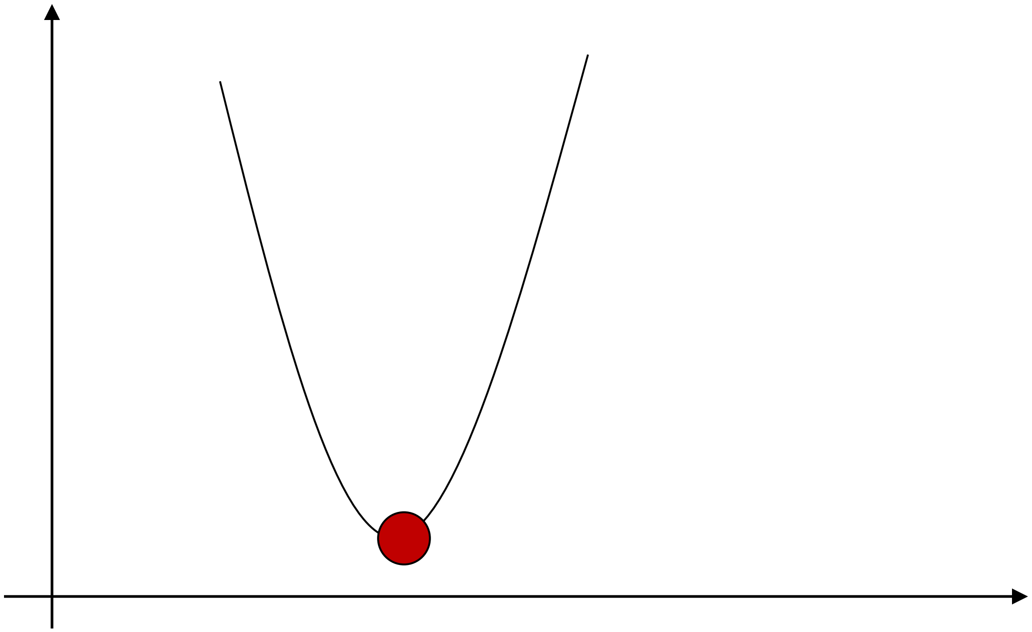
A strongly convex function is convex.

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

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)
- ….

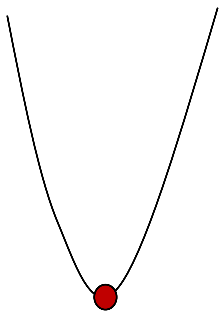
Parameters

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

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

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

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

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

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

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

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

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

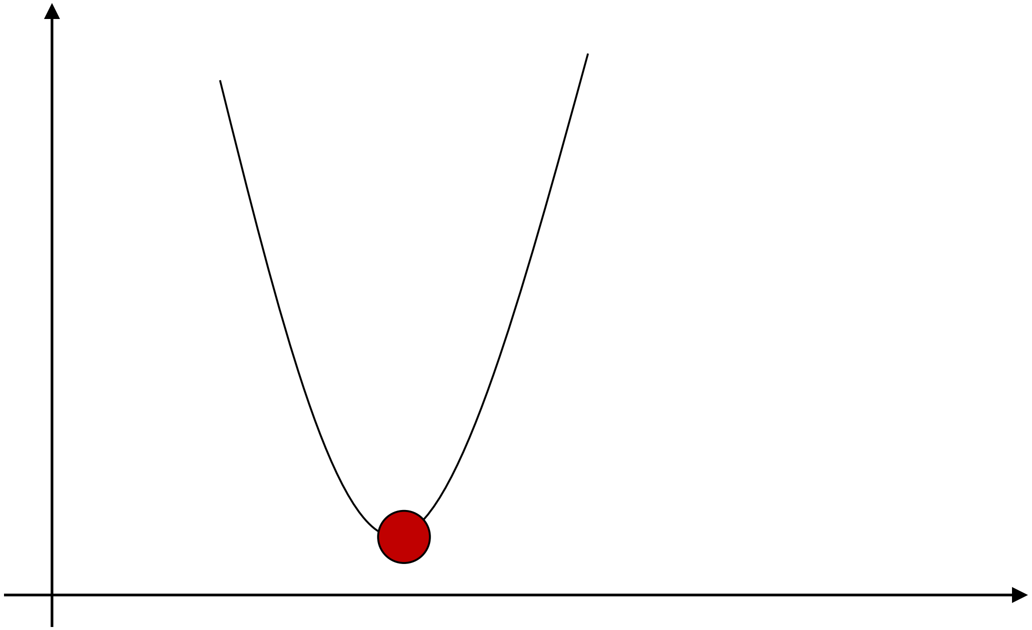
In matlab, you can simply call $X \setminus Y$ or $pinv(X) * Y$

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



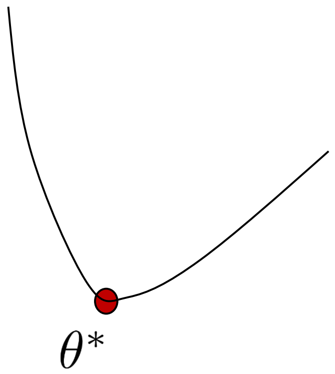
$$\theta^* = \arg min_\theta g(\theta)$$

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

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

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

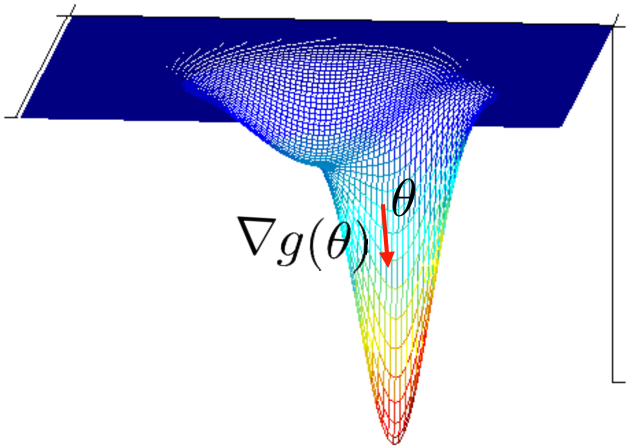
Convex and differentiable: gradient descent



Definition:

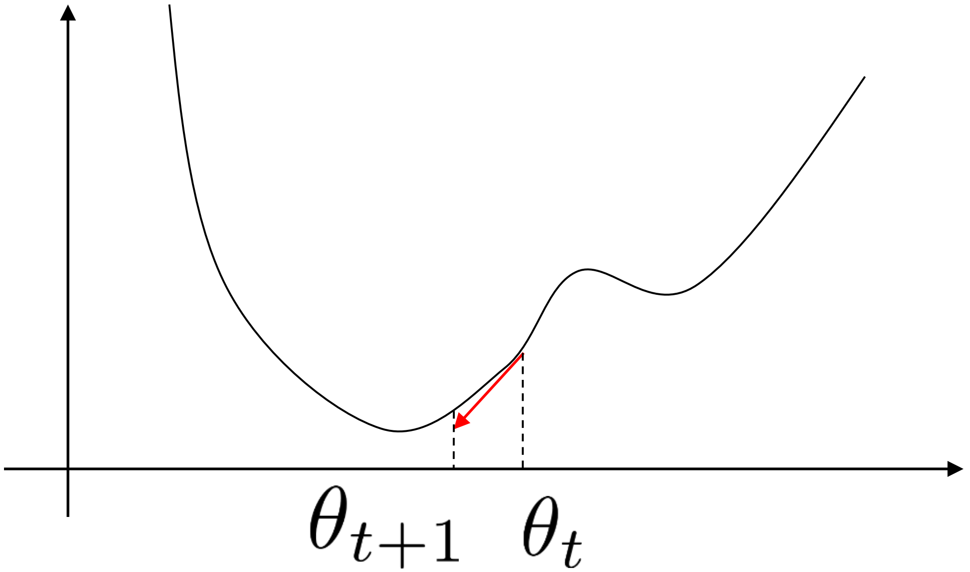
1. x^* is a globally optimal solution for $\theta^* \in \Omega$ and $g(\theta^*) \leq g(\theta) \forall \theta \in \Omega$
2. $x^\star$ is a locally optimal solution if there is a neighborhood $\mathcal{N}$ around x such that $\theta^\star \in \Omega$, $g(\theta^\star) \leq g(\theta)$, $\forall \theta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

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

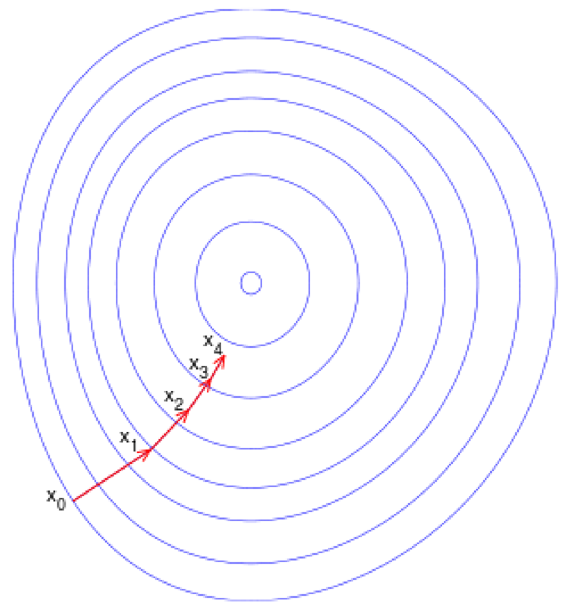
(a) Pick a direction v

(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

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

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

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

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

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

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

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

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

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

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