Decision and Optimization

ΕΠΛ 428: IOT PROGRAMMING

Dr. Panayiotis Kolios
Assistant Professor, Dept. Computer Science,
KIOS CoE for Intelligent Systems and Networks
Office: FST 01, 116
Telephone: +357 22893450 / 22892695
Web: https://www.kios.ucy.ac.cy/pkolios/

Πανεπιστήμιο Κύπρου

- Decision Theory
  - Bayesian Statistics
  - AUROC curve
  - Precision Recall Curve / F1 score
- Information Theory
  - Entropy
  - KL divergence (Relative entropy)
  - Joint Entropy
  - Mutual Information
- Optimization
  - Parameter estimation (model fitting)
  - Model identification (regularization)

- Describes the probability of an event, based on prior knowledge of conditions that might be related to the event

Prior distribution

Likelihood

$$p(H = h|Y = y) = \frac{p(H = h)p(Y = y|H = h)}{p(Y = y)}$$

$$\text{posterior} \propto \text{prior} \times \text{likelihood}$$

Posterior distribution / new belief state

Marginal Likelihood

- Assume the decision maker, or agent, has a set of possible actions, A, to choose from
- Every action has cost and benefits depending on underlying state of nature
- Encode this information to loss function $\ell(h, a)$, loss when we incur if action $a \in A$ is taken at state of nature, $h \in \mathcal{H}$
- Compute posterior expected loss or risk for each possible action

$$R(a|\boldsymbol{x}) \triangleq \mathbb{E}_{p(h|\boldsymbol{x})}\left[\ell(h,a)\right] = \sum_{h \in \mathcal{H}} \ell(h,a) p(h|\boldsymbol{x})$$

$$\pi^*(\boldsymbol{x}) = \underset{a \in \mathcal{A}}{\operatorname{argmin}}\, \mathbb{E}_{p(h|\boldsymbol{x})}\left[\ell(h,a)\right]$$

- Use Bayesian decision theory to decide the optimal class label to predict given an observed input $x \in X$
  - Suppose the states of nature correspond to class labels, so $H = Y = \{1, \dots, C\}$
  - the actions correspond to class labels, so $A = Y$
- In this setting, a very commonly used loss function is the zero-one loss, $\ell_{01}(y^*, \hat{y})$, simply counts how many mistakes a hypothesis would make on training data

	$\hat{y}=0$	$\hat{y}=1$
$y^*=0$	0	1
$y^*=1$	1	0

$$\ell_{01}(y^*, \hat{y}) = \mathbb{I}(y^* \neq \hat{y})$$

In this case, the posterior expected loss is

$$R(\hat{y}|x) = p(\hat{y} \neq y^*|x) = 1 - p(y^* = \hat{y}|x)$$

Mode of the posterior distribution or maximum a posteriori or MAP estimate

$$\pi(x) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, p(y|x)$$

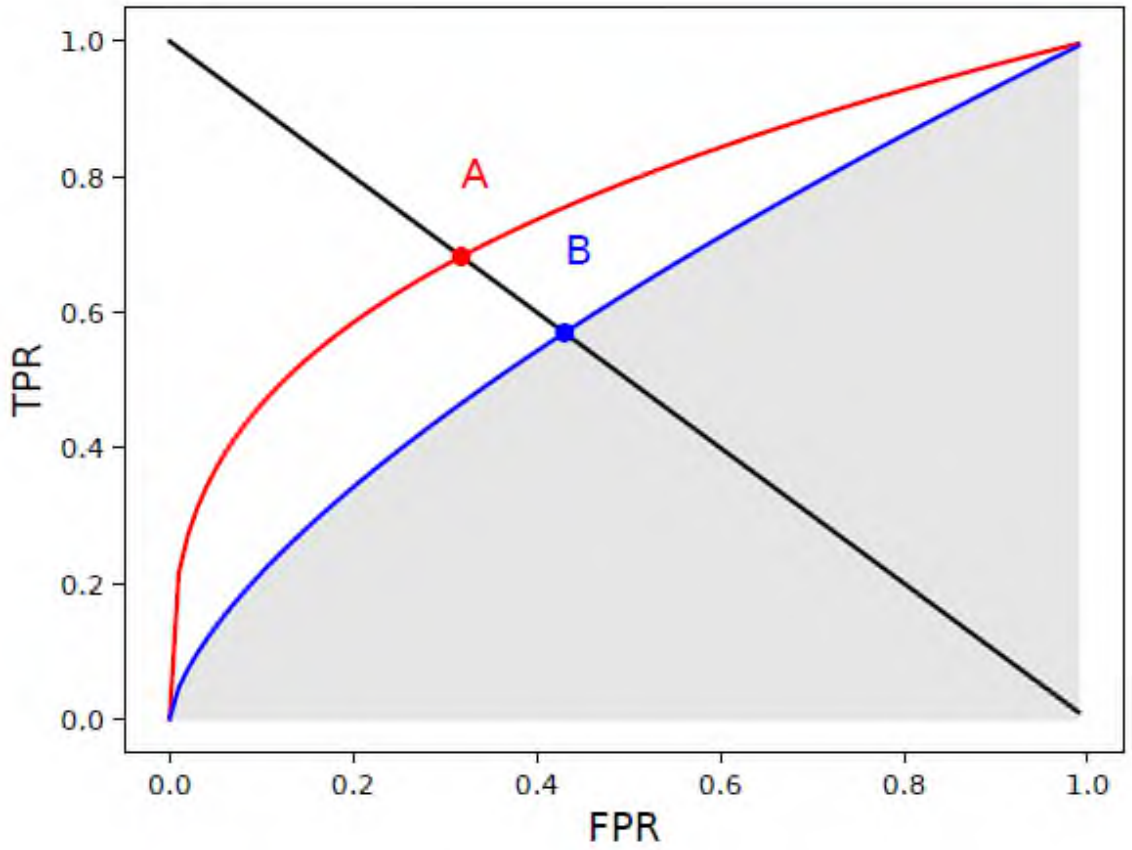
Accuracy	(TP + TN) / (TP + FN + FP + TN)
Recall / True Positive Rate (TPR) = predict how many disease correctly in disease cohort	TPR = TP / (TP + FN)
False Positive Rate (FPR) = predict how many disease in healthy cohort	FPR = FP / (FP + TN)

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection

If we have imbalanced testing data (class 0 = 100, class 1 = 10), if the model predict everything as class 0, accuracy = 90%

Reliable?? No

Πανεπιστήμιο Κύπρου

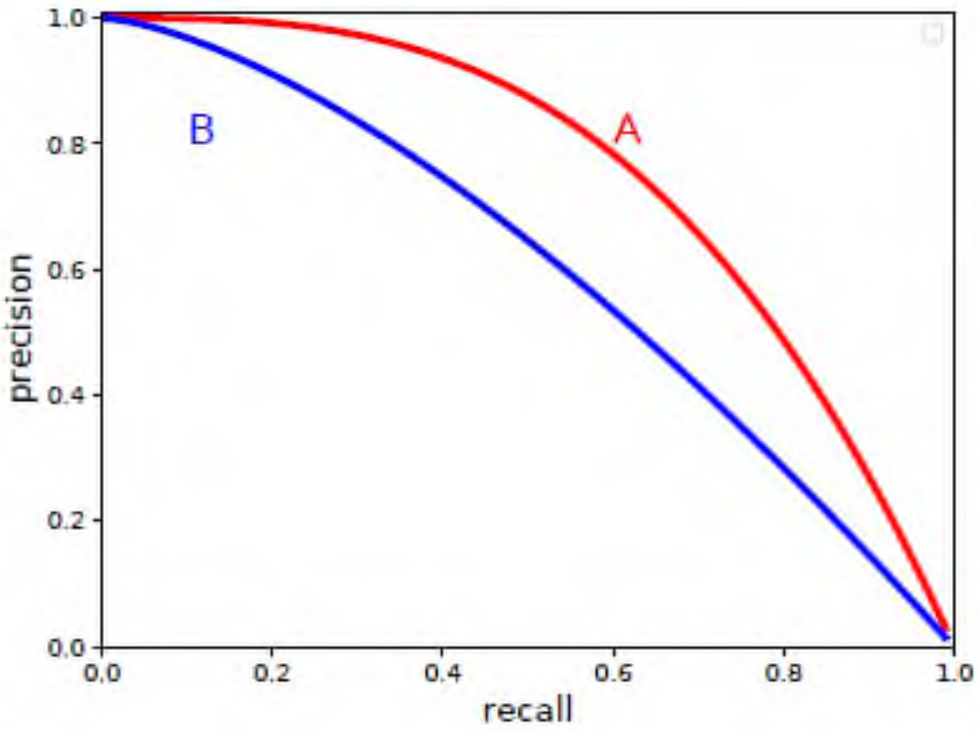


- Area under ROC curve is computed
- The higher the AUROC, the better the classifier

F1 – score

Recall / True Positive Rate (TPR) = predict how many positive correctly for both positive and false negative	TPR = TP / (TP + FN)
Precision = Positive Predictive Value (PPV) predict how many positive correctly if prediction is positive	PPV = TP / (TP +FP)

		Predicted condition	
	Total population = P + N	**Positive (PP)**	**Negative (PN)**
Actual condition	**Positive (P)**	**True positive (TP)**, hit	**False negative (FN)**, type II error, miss, underestimation
	Negative (N)	**False positive (FP)**, type I error, false alarm, overestimation	**True negative (TN)**, correct rejection



- Area under PR curve is computed
- The higher the PR AUC, the better the classifier
  - F1-score

- All of machine learning revolves around **optimization**
- Find values for a set of variables the minimize loss function
- Regression and model selection aim to provide models for the available data
- Regression → Curve fitting
    - polynomial / exponential fitting
- Result in **set** of **equations** modelling (non)linear systems
- Linear example:
    - $\boldsymbol{Ax} = \boldsymbol{b}$, need to estimate the parameters of matrix A
    - $\|\boldsymbol{Ax} - \boldsymbol{b}\|_2$ find the best model for the given data
- Regularization is when a penalty term is introduced in the optimization process to produce a solution out of the many possible alternatives

- Simple function that describes a trend by minimizing the **sum-square error** between the selected function f(·) and its fit to the data
- Consider a set of n data points
  - $(x_1, y_1), (x_2, y_2), (x_3, y_3), \cdots, (x_n, y_n)$
- Further, assume that we would like to find a best fit line through these points
- We can approximate the line by the function
  - $f(x) = \beta_1 x + \beta_2$

  where the constants β1 and β2 are chosen to minimize some error
  - $y_k = f(x_k) + E_k$ where y_k is the data value, that is approximated by $f(x_k)$ and the residual error E_k

- Various error metrics can be used to fit the model
  - minimize the error (norm)

L2 loss

$$\ell_2(h,a) = (h-a)^2$$

- Sensitive to outlier

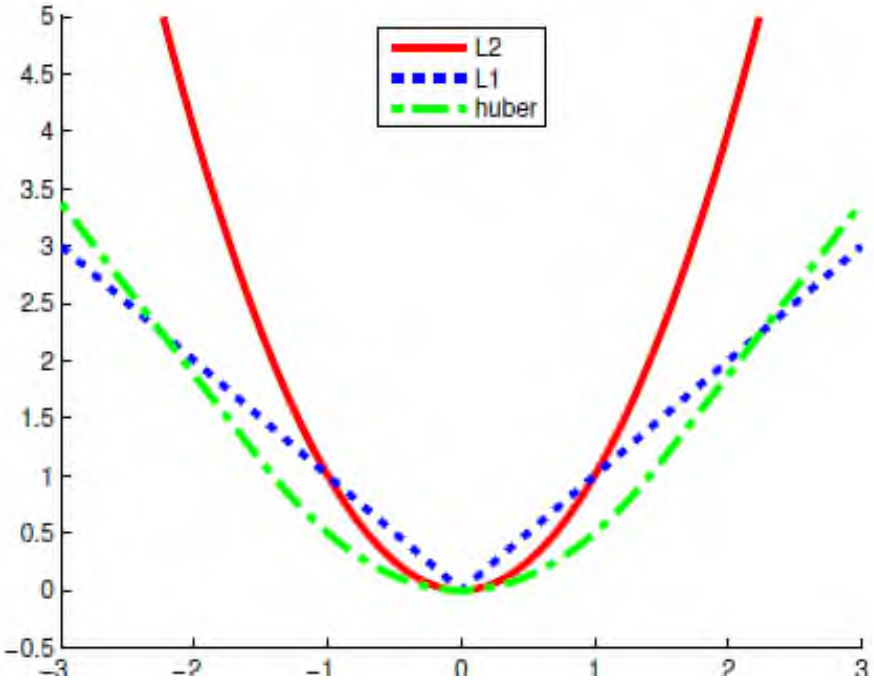
L1 loss

$$\ell_1(h,a) = |h-a|$$

Huber loss

$$\ell_\delta(h,a) = \begin{cases} r^2/2 & \text{if } |r| \le \delta \\ \delta|r| - \delta^2/2 & \text{if } |r| > \delta \end{cases}$$

where $r = h - a$.



Πανεπιστήμιο Κύπρου

Quantifying information is the foundation of the field of information theory.

The intuition behind quantifying information is the idea of measuring how much surprise there is in an event. Those events that are rare (low probability) are more surprising and therefore have more information than those events that are common (high probability).

- **Low Probability Event**: High Information (*surprising*).
- **High Probability Event**: Low Information (*unsurprising*).

Rare events are more uncertain or more surprising and require more information to represent them than common events.

We can calculate the amount of information there is in an event using the probability of the event. This is called "*Shannon information*," "*self-information*," or simply the "*information*," and can be calculated for a discrete event *x* as follows:

- information(x) = -log(p(x))

Where *log()* is the base-2 logarithm and *p(x)* is the probability of the event *x*.

The choice of the base-2 logarithm means that the units of the information measure is in bits (binary digits). This can be directly interpreted in the information processing sense as the number of bits required to represent the event.

The calculation of information is often written as *h()*; for example:

- h(x) = -log(p(x))

The negative sign ensures that the result is always positive or zero.

- Calculates the number of bits required to represent or transmit an average event from one distribution compared to another distribution
- The intuition for this definition comes if we consider a target or underlying probability distribution P and an approximation of the target distribution Q. Then the cross-entropy of Q from P is average bits of information needed to identify an event drawn from the estimated probability distribution q, rather than the true distribution p

The cross-entropy between two probability distributions, such as Q from P, can be stated formally as:

- H(P, Q)

$$H(P,Q) = -\sum^{\{x \in X\}} P(x) \times \log(Q(x))$$

Where H() is the cross-entropy function, P may be the target distribution and Q is the approximation of the target distribution.

- Kullback Leibler divergence, or KL divergence is a measure of how one probability distribution is different from a second, reference probability distribution (relative entropy)

$$\mathbb{KL}\left(p\|q\right) \triangleq \sum_{y\in\mathcal{Y}} p(y)\log\frac{p(y)}{q(y)}$$

$$\begin{aligned}\mathbb{KL}\left(p\|q\right) &= \sum_{y\in\mathcal{Y}} p(y)\log p(y) - \sum_{y\in\mathcal{Y}} p(y)\log q(y)\\ &= -\mathbb{H}(p) + \mathbb{H}(p,q)\\ \mathbb{H}(p) &\triangleq -\sum_{y} p(y)\log p(y)\\ \mathbb{H}(p,q) &\triangleq -\sum_{y} p(y)\log q(y)\end{aligned}$$

Entropy;
- Max if the distribution p is uniform
- Zero if p is a delta function

Cross-Entropy

If KL = 0 correctly predict the probabilities of all possible future events

Πανεπιστήμιο Κύπρου

$$\mathbb{H}(p) + \mathbb{KL}(p\|q) = \mathbb{H}(p,q)$$

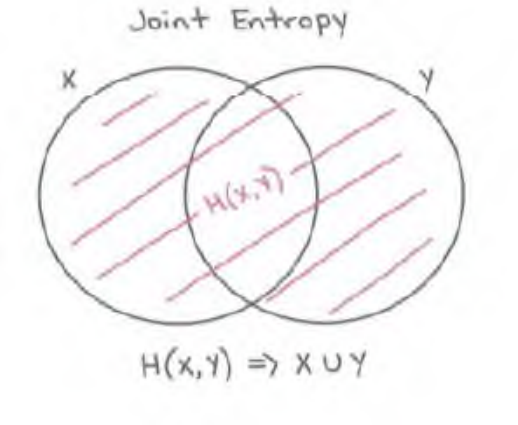
average number of extra bits required when compressing data coming from p if your code is designed based on q distribution

average bits of information needed to identify an event drawn from the estimated probability distribution q, rather than the true distribution p

Πανεπιστήμιο Κύπρου

The joint entropy of two random variables X and Y is defined as

$$\mathbb{H}(X,Y) = -\sum_{x,y} p(x,y) \log_2 p(x,y)$$

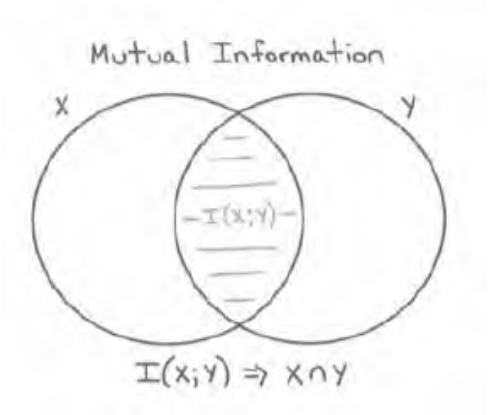


Πανεπιστήμιο Κύπρου

How similar two distributions were

The mutual information between rv's X and Y is defined as follows:

$$\mathbb{I}(X;Y) \triangleq \mathbb{KL}\left(p(x,y) \| p(x)p(y)\right) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)\,p(y)}$$



Used in Clustering – to measure cluster quality

$$NMI(X,Y) = \frac{\mathbb{I}(X;Y)}{\min\left(\mathbb{H}(X), \mathbb{H}(Y)\right)} \leq 1$$

NMI = Normalized Mutual Information

OPTIMIZATION

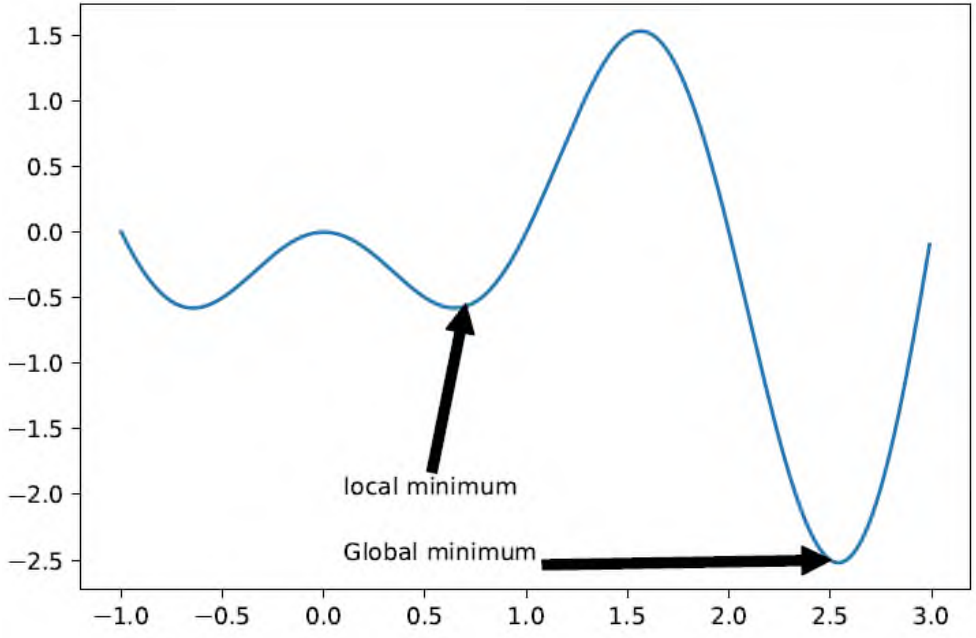
- The core problem in machine learning is parameter estimation (aka model fitting).

- This requires solving an **optimization problem**, where we try to find the values for a set of variables, $\theta \in \Theta$, that minimize a scalar-valued loss function or cost function, $\mathcal{L}(\theta)$

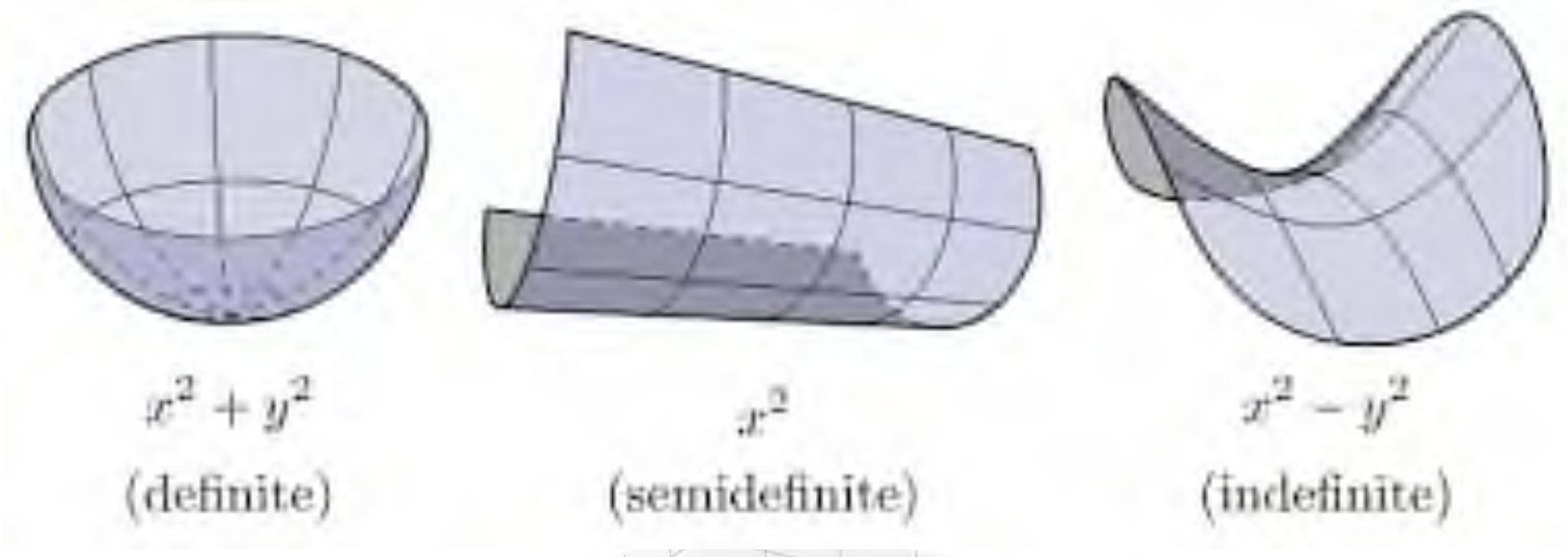
$$\theta^* \in \underset{\theta \in \Theta}{\operatorname{argmin}} \mathcal{L}(\theta)$$

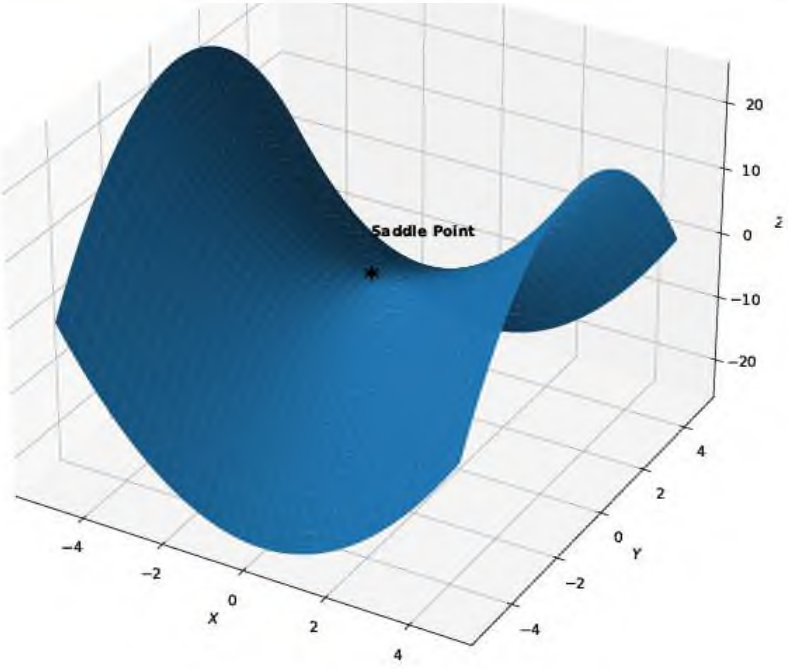
- The **objective function** is what we want to maximize or minimize
- The algorithm that can find an optimum of an objective function is often called a **solver**

- A set of values in $\theta \in \Theta$ that satisfies $\underset{\theta \in \Theta}{\operatorname{argmin}} \mathcal{L}(\theta)$ is called a **global optimum**
- In general, finding global optima is computationally intractable
- Alternatively, try to find a **local optimum**
  - Set of values θ which has lower (or equal) cost than "nearby" points



- Let $\mathrm{g}(\theta) = \nabla \mathcal{L}(\theta)$ be the gradient vector
- Let $\mathrm{H}(\theta) = \nabla^2 \mathcal{L}(\theta)$ be the Hessian matrix
- Consider a point $\theta^* \in \mathbb{R}^D$
  - let $g^* = g(\theta)\big|\theta^*$ be the gradient at that point
  - $H^* = H(\theta)\big|\theta^*$ be the corresponding Hessian
  - Every local minimum:
    - Necessary condition: $g^* = 0$ and H^* must be positive semi-definite
    - Sufficient condition: If $g^* = 0$ and H^* is positive definite, then θ^* is a local optimum
- A zero gradient is not sufficient, since a stationary point could be a local minimum, maximum or **saddle point**, which is a point where some directions point downhill, and some uphill





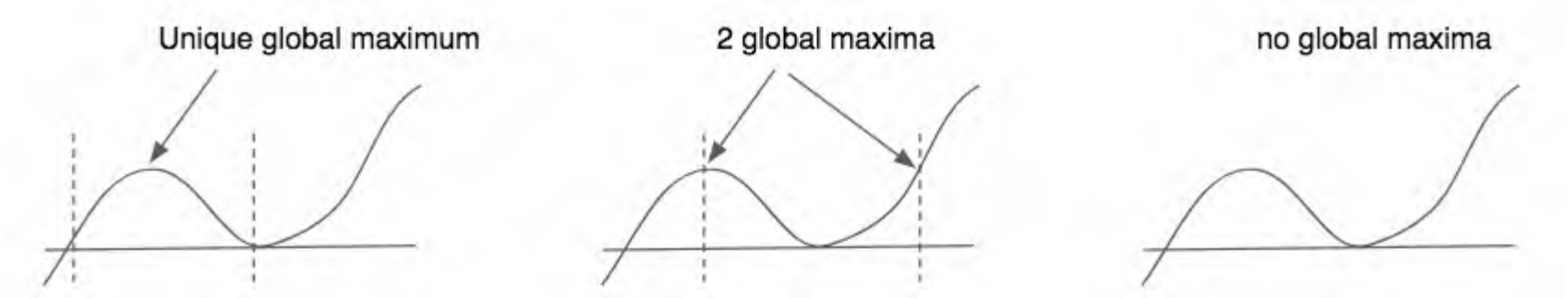
Πανεπιστήμιο Κύπρου

- In addition to the objective function we often have a set of **constraints** on the allowable values
- Set of constraints C separated into
  - **inequality constraints**, $g_j(\theta) \leq 0$ for $j \in J$
  - **equality constraints**, $h_k(\theta) = 0$ for $k \in K$

$$\theta^* \in \underset{\theta \in C}{\operatorname{argmin}} \mathcal{L}(\theta)$$

- A common strategy for solving constrained problems is to create penalty terms that measure how much we violate each constraint
- We then add these terms to the objective function and solve an unconstrained optimization problem
- The Lagrangian is one such combined objective

- If too many constrained → empty feasible sets



- The task of finding any point (regardless of its cost) in the feasible set is called a feasibility problem; this can be a hard subproblem in itself

- In **convex** optimization, we require the objective to be a convex function defined over a convex set
- A convex set S includes $x, x' \in S$

$$\lambda x + (1 - \lambda)x' \in S, \forall \lambda \in [0,1]$$

- That is, if we draw a line from x to x′, all points on the line lie inside the set

Convex

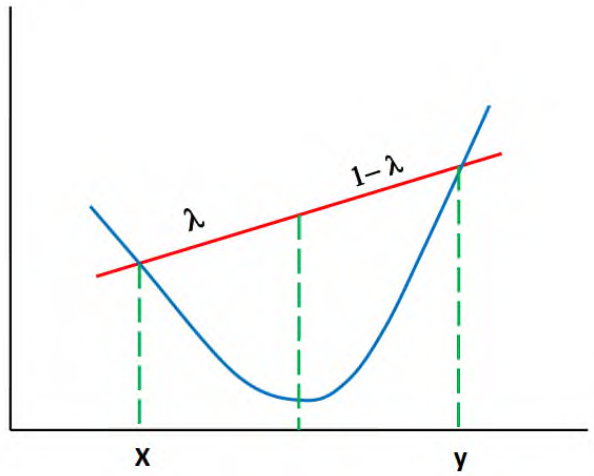
Not Convex

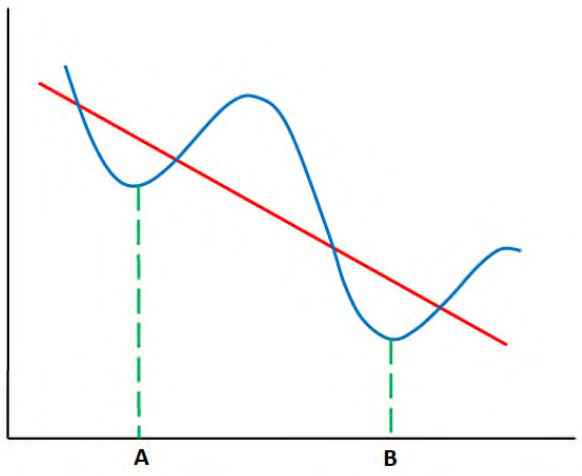
- We say f is a convex function if its epigraph (the set of points above the function) defines a convex set
- Function $f(x)$ is convex if it is defined on a convex set and if for any x, y and for any λ we have:

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(\mathrm{y})$$

Y-axis value that fell between the domain from x and y

Straight line between (x, f(x)) and (y, f(y)) as a function of λ

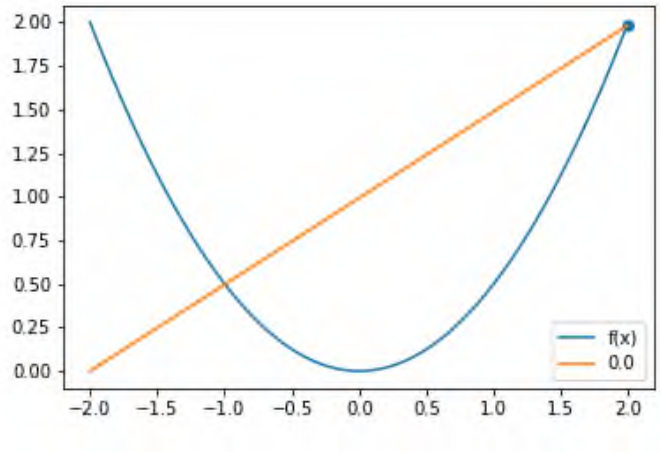




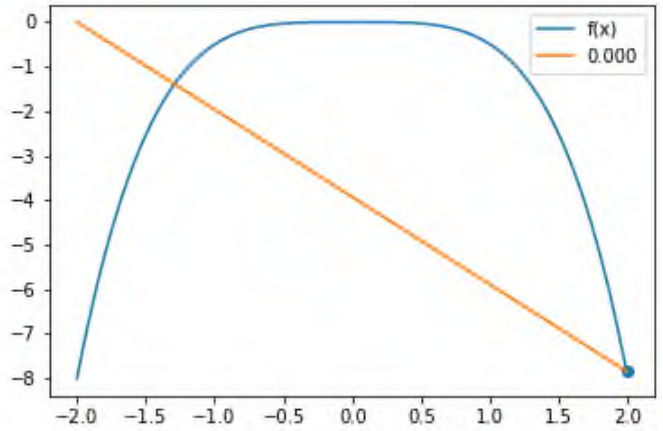
Πανεπιστήμιο Κύπρου

- A function $f(x)$ is concave if $-f(x)$ is convex

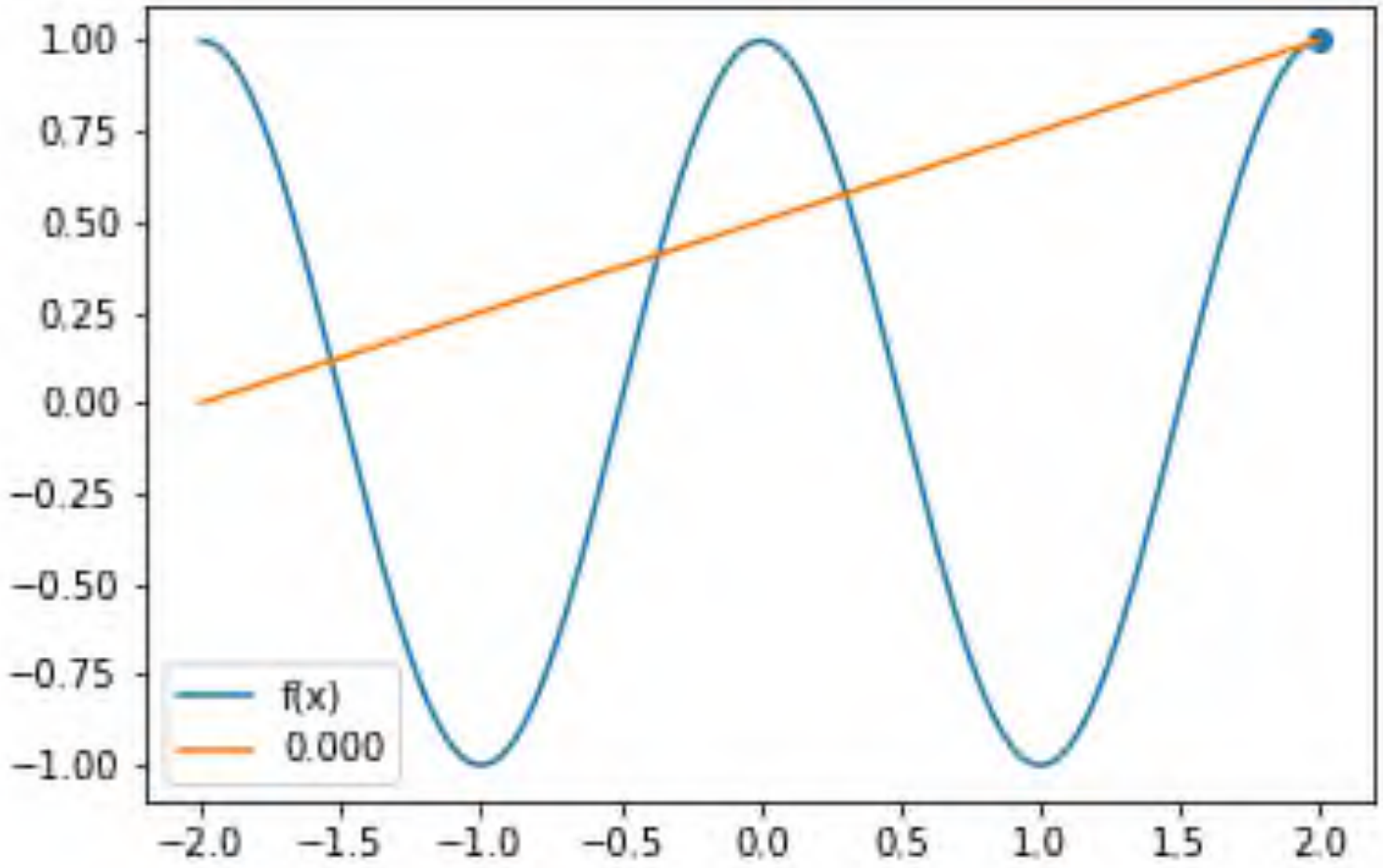
$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(\mathrm{y})$$



Convex



Concave


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
f(x)
0.000

Strictly convex

Strictly concave

Convex but not strictly

*N*either convex nor concave → saddle point

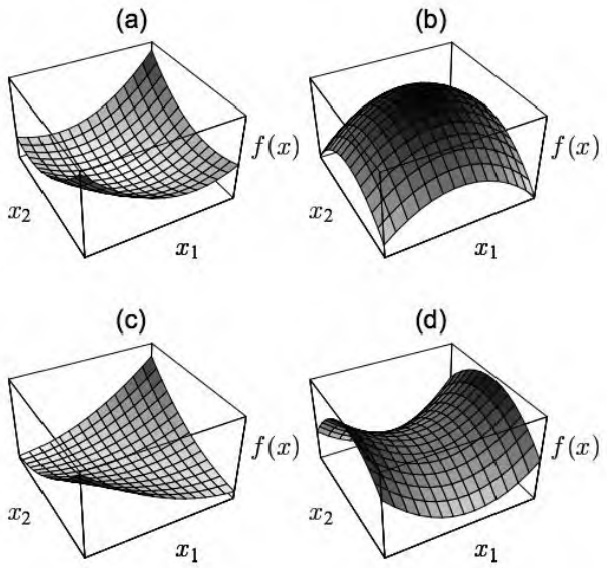
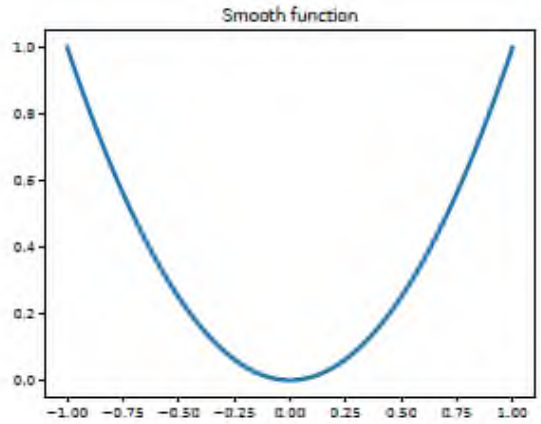
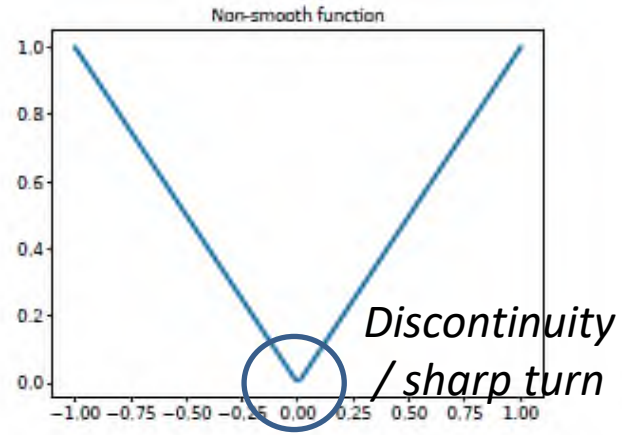


Figure 8.6: The quadratic form $f(\boldsymbol{x}) = \boldsymbol{x}^{\mathsf{T}}\mathbf{A}\boldsymbol{x}$ in 2d. (a) $\mathbf{A}$ is positive definite, so f is convex. (b) $\mathbf{A}$ is negative definite, so f is concave. (c) $\mathbf{A}$ is positive semidefinite but singular, so f is convex, but not strictly. Notice the valley of constant height in the middle. (d) $\mathbf{A}$ is indefinite, so f is neither convex nor concave. The stationary point in the middle of the surface is a saddle point. From Figure 5 of [She94].

Πανεπιστήμιο Κύπρου





$$|f(x_1) - f(x_2)| \leq L|x_1 - x_2|$$

Lipschitz constant – quantify
the degree of smoothness

- Nonsmooth function with at least some points where the gradient of the objective function or the constraints is not well-defined
- Could partition the objective in 2 parts: one with the smooth (differentiable) objective and a part with the nonsmooth terms
- Composite objective:

$$\mathcal{L}(\theta)=\mathcal{L}_s(\theta) + \mathcal{L}_r(\theta)$$

- Gradient descent
- Step size/ learning rate
- Convergence rate
- Momentum Method

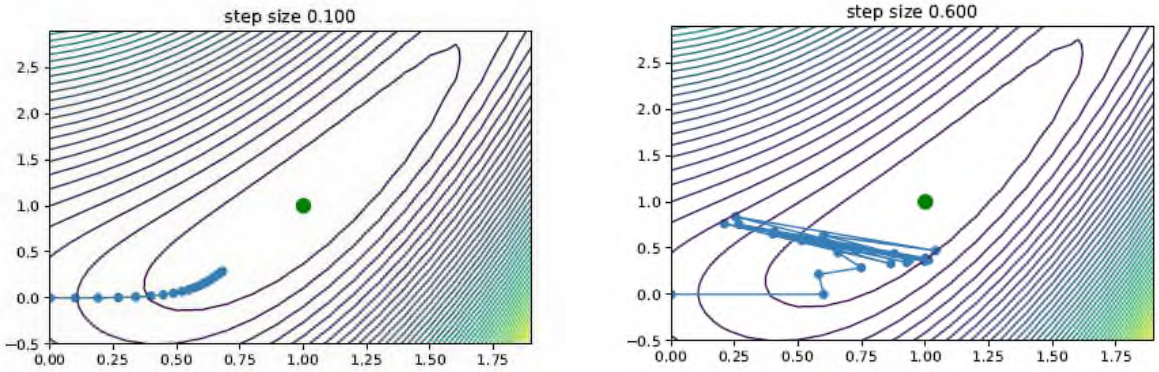
- Iterative optimization methods that leverage first-order derivatives of the objective function, i.e., they compute which directions point “downhill”, but they ignore curvature information

$$\theta_{t+1} = \theta_t + \rho_t d_t$$

Updated parameter

Step size

Descent direction



- Steepest descent will have global convergence iff the step size satisfies

$$\rho < \frac{2}{\lambda_{\max}(\mathbf{A})}$$

$$\mathcal{L}(\theta) = \tfrac{1}{2}\theta^{\mathsf{T}}\mathbf{A}\theta + b^{\mathsf{T}}\theta + c \text{ with } \mathbf{A} \succeq 0.$$

λmax = max eigenvalue

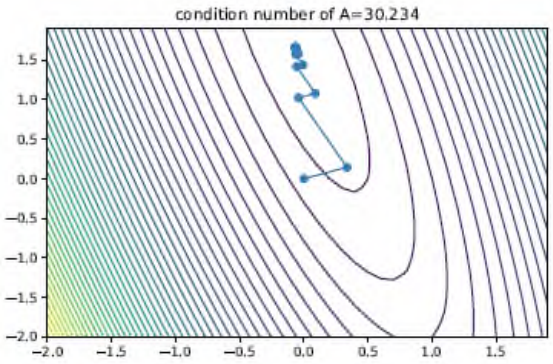
- Consider a quadratic objective:

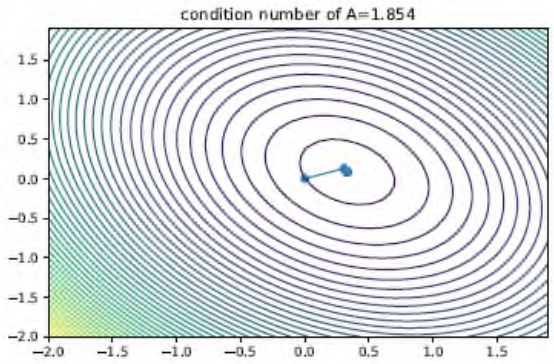
$$\mathcal{L}(\theta) = \frac{1}{2}\theta^{\mathsf{T}}\mathbf{A}\theta + b^{\mathsf{T}}\theta + c \text{ with } \mathbf{A} \succeq 0.$$

$$\mu = \left(\frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}\right)^2$$

$$\mu = \left(\frac{\kappa-1}{\kappa+1}\right)^2, \text{ where } \kappa = \frac{\lambda_{\max}}{\lambda_{\min}} \text{ is the condition number of } \mathbf{A}.$$

- The condition number measures how “skewed” the space is, in the sense of being far from a symmetrical “bowl”





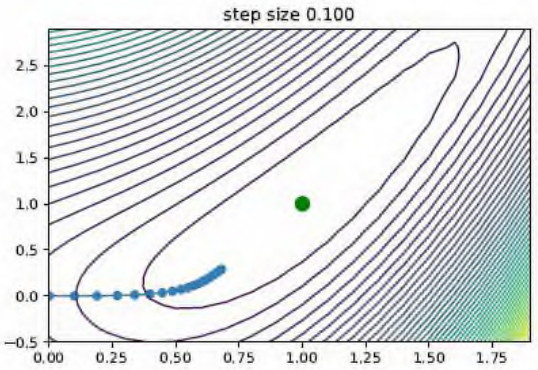
- Gradient descent can move very slowly along flat regions of the loss landscape
- A solution to this problem is using **momentum**
  - move faster along directions that were previously good, and to slow down along directions where the gradient has suddenly changed

Thinking like a ball rolling downward. At flat surface, it roll down slowly. At sharp region, roll down faster.

momentum

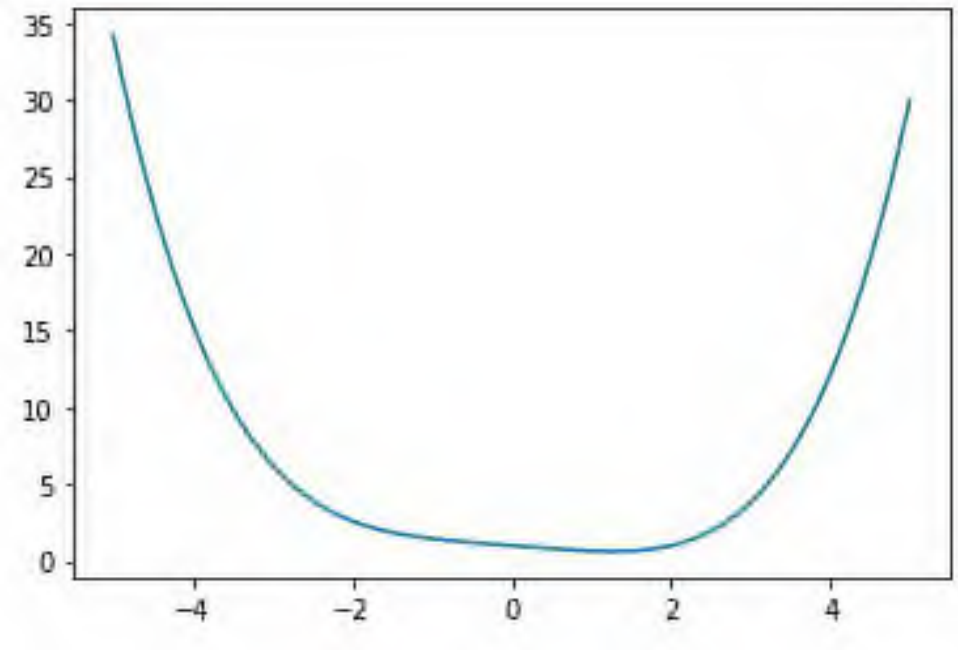
$$m_t = \beta m_{t-1} + g_{t-1}$$

$$\theta_t = \theta_{t-1} - \rho_t m_t$$



- Normally β=0.9
- If β = 0 → gradient descent

- Consider we want to minimize this loss function, $f(\theta) = \frac{1}{20}\theta^4 - \frac{2}{5}\theta + 1$
- We know the min point is at θ * = 1.5 (compute f'(θ) = 0)
- What if we use line search (iterative method to find θ *)



- Start with random number, θ_t

Compute the gradient at θ_k

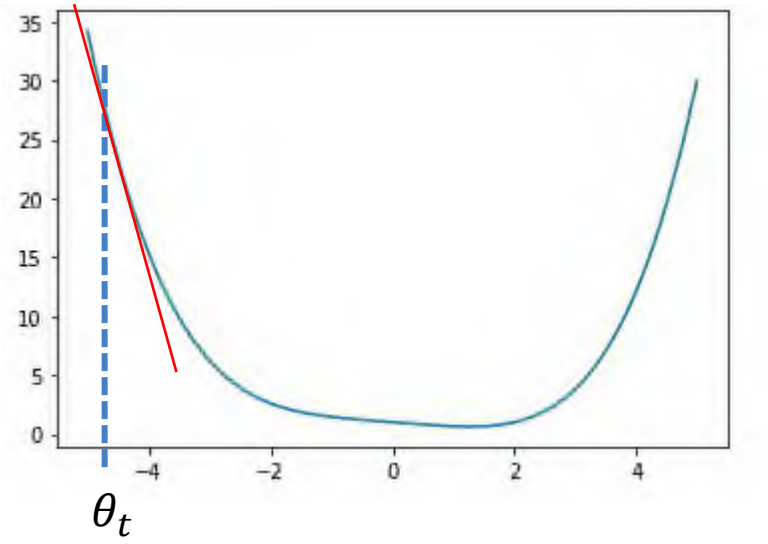
$$f'(\theta_t) = \frac{f(\theta) - f(\theta_t)}{\theta - \theta_t}$$

If $f'(\theta_t)$ is negative, move θ_t to the right
If $f'(\theta_t)$ is positive, move θ_t to the left

$$\theta_{t+1} = \theta_t - \alpha f'(\theta_t)$$

Large step size, α will overshoot
Small step size, α will be very slow

Slow to converge



Approximate with non linear graph

Compute the second derivative at θ_k

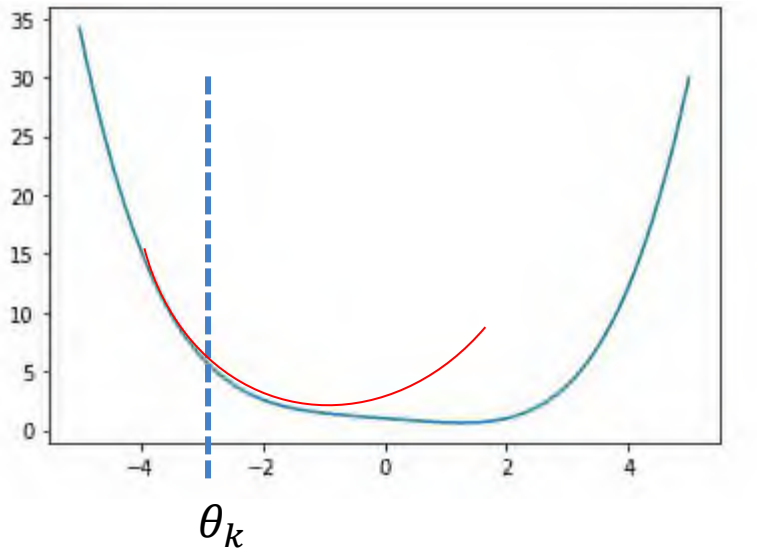
$$f''(\theta_k) = \frac{f(\theta) - f(\theta_k)}{\theta - \theta_k}$$

$$\theta_{t+1} = \theta_k - \alpha f'(\theta_k)$$

Using some algebra, the update formula is

$$\theta_{t+1} = \theta_k - \alpha \frac{1}{f''(\theta_k)} f'(\theta_k)$$

- Faster convergence



- Second-order optimization methods incorporate curvature in various ways (e.g., via the Hessian), which may yield faster convergence
- Newton's method:

$$\theta_{t+1} = \theta_t - \rho_t \mathbf{H}_t^{-1} g_t$$

H = Hessian matrix
ρ = step size
g_t = gradient

where

$$\mathbf{H}_t \triangleq \nabla^2 \mathcal{L}(\theta)|_{\boldsymbol{\theta}_t} = \nabla^2 \mathcal{L}(\theta_t) = \mathbf{H}(\theta_t)$$

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

Batch Gradient Descent

$$f(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\theta_{t+1} = \theta_t - \alpha\, f'(\theta)$$

Need to compute for all data,
if you have sample size, m = 1million
Very slow to update θ

Stochastic Gradient Descent

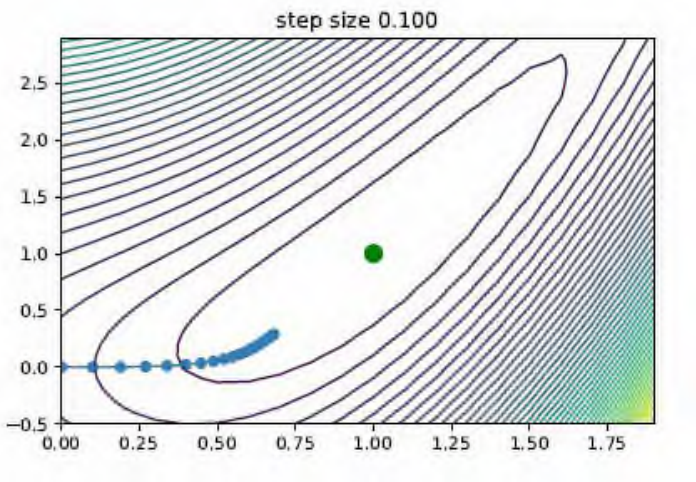
$$cost\left(\theta, (x^{(i)}, y^{(i)})\right) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$$

1. Random Shuffle Data
2. Repeat {
   for j = 1, ... , m {
   $$\theta_{t+1} = \theta_t - \alpha \frac{\partial}{\partial \theta_t} cost\left(\theta, (x^{(i)}, y^{(i)})\right)$$
   }
   }

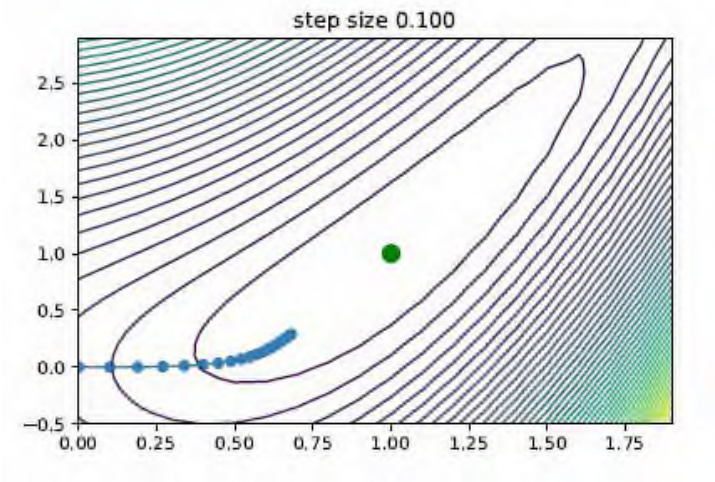
Update θ using only one data point.
Faster

Πανεπιστήμιο Κύπρου

Batch Gradient Descent



Stochastic Gradient Descent



- All of machine learning revolves around optimization
- Regression -> curve fitting
    - polynomial & exponential fitting
    - Solving $\boldsymbol{Ax} = \boldsymbol{b}$
- Model selection frameworks
    - Underdetermined (no sol.) or overdetermined (infinite number of solutions)
    - Solving $\|\boldsymbol{Ax} - \boldsymbol{b}\|_2 - \lambda g(x)$
    - Penalty $g(x)$ used to find a solution (regularization)
- Regression on non-linear models can be solved using gradient descent

- Regression attempts to estimate the relationship among variables using a variety of statistical tools
- Specifically, one can consider the general relationship between independent variables X, dependent variables Y, and some unknown parameters β:

$$\mathrm{Y} = f(X, \beta)$$

- Minimize the sum-square error between the selected function f (·) and its fit to the data

- Consider a set of n data points

$$(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_n, y_n)$$

- Need to find a best fit line through these points
- Approximate the line by the function

$$f(x) = \beta_1 x + \beta_2$$

- where the parameters β_1 and β_2 are chosen to minimize some error associated with the fit
- Linear regression model

$$f(x_k) = y_k + E_k$$

- where y_k is the data value and E_k is the error of the fit

- Various error metrics can be minimized when approximating with a given function f(x)
- The choice of error metric, or norm, used to compute a goodness-of-fit ℓ_2 (least-squares), ℓ_1 , and ℓ_∞ norms

$$E_\infty(f) = \max_{1<k<n} |f(x_k) - y_k| \qquad \text{Maximum Error } (\ell_\infty)$$

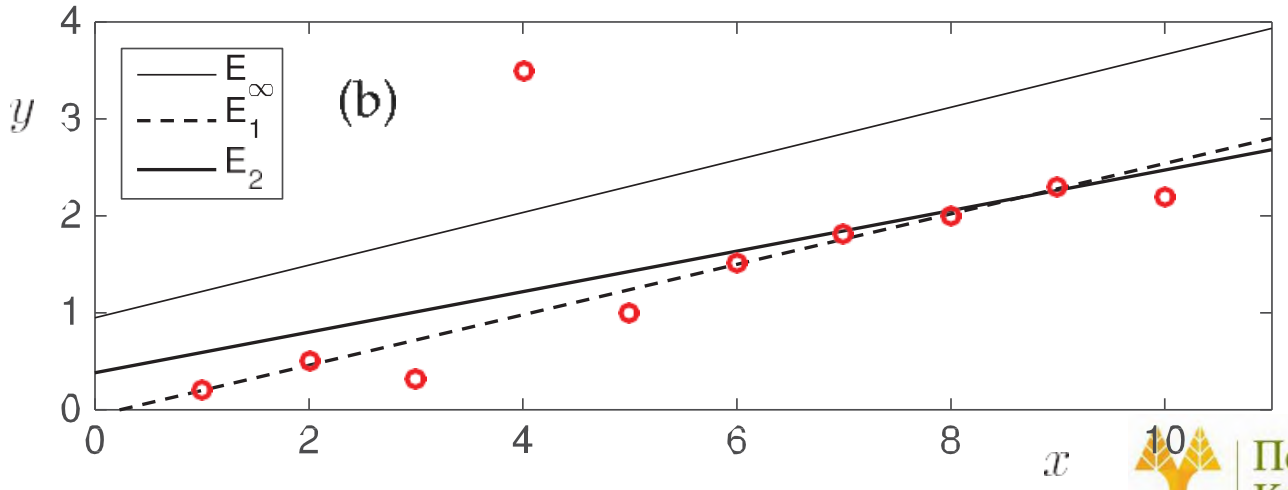
$$E_1(f) = \frac{1}{n}\sum_{k=1}^{n} |f(x_k) - y_k| \qquad \text{Mean Absolute Error } (\ell_1)$$

$$E_2(f) = \left(\frac{1}{n}\sum_{k=1}^{n} |f(x_k) - y_k|^2\right)^{1/2} \qquad \text{Least-squares Error } (\ell_2).$$

- In general, ℓ_P-norm

$$E_P(f) = \left(\frac{1}{n}\sum_{k=1}^{n} |f(x_k) - y_k|^P\right)^{\frac{1}{P}}$$

- When fitting a curve to a set of data, the root-mean square (RMS) error is often chosen to be minimized
- This is called a least-squares fit
- Example depicting fits that minimize errors E_∞, E_2, E_1
  - E_∞ error line fit is strongly influenced by the one data point which does not fit the trend
  - E_2, E_1 line fit nicely through the bulk of the data, although their slopes are quite different in comparison to when the data has no outliers



- Least-squares fitting to linear models has critical advantages over other norms and metrics.
- Specifically, the optimization is inexpensive, since the error can be computed analytically
- e.g. for E_2 $$f(x) = \beta_1 x + \beta_2$$

$$E_2(f) = \left(\frac{1}{n} \sum_{k=1}^{n} |f(x_k) - y_k|^2 \right)^{\frac{1}{2}} = \sum_{k=1}^{n} |\beta_1 x_k + \beta_2 - y_k|^2$$

- Minimizing this sum requires differentiation

$$\frac{\partial E_2}{\partial \beta_1} = 0 \qquad \frac{\partial E_2}{\partial \beta_2} = 0$$

$$\frac{\partial E_2}{\partial \beta_1} = 0 \qquad \sum_{k=1}^{n} 2(\beta_1 x_k + \beta_2 - y_k) x_k = 0$$

$$\frac{\partial E_2}{\partial \beta_2} = 0 \qquad \sum_{k=1}^{n} 2(\beta_1 x_k + \beta_2 - y_k) = 0$$

$$\begin{pmatrix} \sum_{k=1}^{n} {x_k}^2 & \sum_{k=1}^{n} x_k \\ \sum_{k=1}^{n} x_k & n \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{n} x_k y_k \\ \sum_{k=1}^{n} y_k \end{pmatrix}$$

- General theory of nonlinear regression assumes that the fitting function takes the general form

$$f(x) = f(x, \beta), \beta \in \mathbb{R}^{\mathrm{m}}$$

- as before β coefficients used to minimize error

$$E_2(\beta) = \sum_{k=1}^{n} |f(x_k, \beta) - y_k|^2 \qquad \frac{\partial E_2}{\partial \beta_j} = 0, j = 1, \dots, m$$

$$\sum_{k=1}^{n} (f(x_k, \beta) - y_k) \frac{\partial f}{\partial \beta_j} = 0$$

- no general methods available for solving such nonlinear systems
- and thus gradient descent methods are employed