# Optimization for Machine Learning

Xiao Wang

Shanghai University of Finance and Economics

April 15, 2021

# Contents

# Neural Network

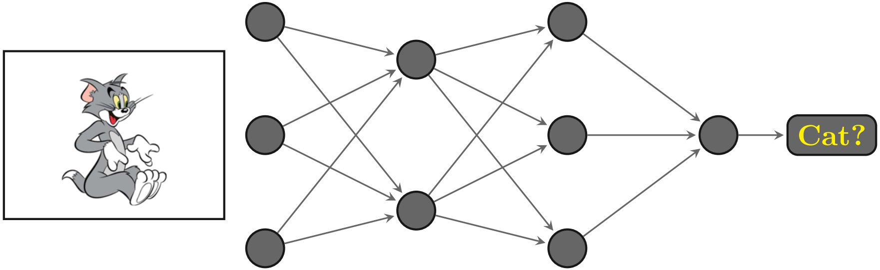
## Supervised Learning Problem:

Let $\mathcal{K}$ be a set (e.g. images, texts), with $F : \mathcal{K} \to \mathbb{R}^m$ a function. Suppose that the values of $F$ are known only on a proper finite set $\mathcal{S} \subset \mathcal{K}$. Can we predict the values of $F(x)$ for $x \in \mathcal{K} \setminus \mathcal{S}$?

# Neural Network

## Supervised Learning Problem:

Let $\mathcal{K}$ be a set (e.g. images, texts), with $F : \mathcal{K} \to \mathbb{R}^m$ a function. Suppose that the values of $F$ are known only on a proper finite set $\mathcal{S} \subset \mathcal{K}$. Can we predict the values of $F(x)$ for $x \in \mathcal{K} \setminus \mathcal{S}$?



# Neural Network

## Definition

The function $\phi : \mathbb{R}^{n_0} \to \mathbb{R}^{n_q}$

$$\phi : \mathbb{R}^{n_0} \xrightarrow{T_1} \mathbb{R}^{n_1} \xrightarrow{h_1} \mathbb{R}^{n_1} \xrightarrow{T_2} \cdots \xrightarrow{T_q} \mathbb{R}^{n_q} \xrightarrow{h_q} \mathbb{R}^{n_q}$$

is called a Feed-forward Neural Network.

# Neural Network

## Definition

The function $\phi : \mathbb{R}^{n_0} \to \mathbb{R}^{n_q}$

$$\phi : \mathbb{R}^{n_0} \xrightarrow{T_1} \mathbb{R}^{n_1} \xrightarrow{h_1} \mathbb{R}^{n_1} \xrightarrow{T_2} \cdots \xrightarrow{T_q} \mathbb{R}^{n_q} \xrightarrow{h_q} \mathbb{R}^{n_q}$$

is called a Feed-forward Neural Network.

- $T_j$ are affine maps of the form

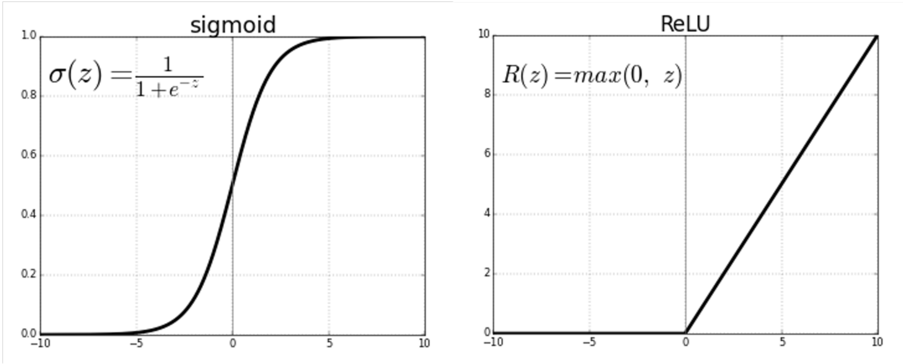
$$T_j(x) = A_j x + b_j,$$

  The weight matrix $A_j$ and bias vector $b_j$ represent the map from one layer to another;

# Neural Network

- $h_j$ are the activation functions that are typically chosen from

$$\left\{ \max\{0, x\}, \left(1 + e^{-x}\right)^{-1} \right\}$$



# Optimization for NNW

## Training

- For a neural network $\phi$ with fixed architecture, $\phi$ is determined by the parameters $\mathbf{A} = (A_1, ..., A_q), \mathbf{b} = (b_1, ..., b_q)$, provided a given set of activation functions;

# Optimization for NNW

## Training

- For a neural network $\phi$ with fixed architecture, $\phi$ is determined by the parameters $\mathbf{A} = (A_1, ..., A_q), \mathbf{b} = (b_1, ..., b_q)$, provided a given set of activation functions;
- To train the network, we choose, for example, the mean-square loss function:

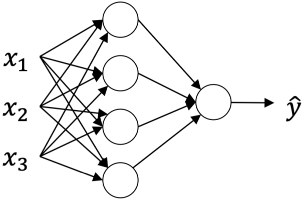
$$L(\mathbf{A}, \mathbf{b}) = \sum_{x \in \mathcal{S}} \| F(x) - \phi(x, \mathbf{A}, \mathbf{b}) \|^2$$

and minimize such $L$ using gradient descent.

# Optimization for NNW

## Example: ReLU network with one hidden layer

- Activation function: $h(x) = \max\{0, x\}$;
- Input dimension: 3;
- Width of hidden layer: 4;
- Weight matrix $W = \{w_i\}_{i=1}^4$: 4 is the number of neurons;
- Bias vector: $b = (b_1, ..., b_4)$
- $z_i = w_i^\top \mathbf{x} + b_i \to h(z_i) \to \hat{y} = (h(z_1), ..., h(z_4))$or $\sum_{i=1}^4 a_i h(z_i) \to$ Output.



# Optimization for NNW

## Cont. with example

Suppose that the NNW is of the form $\phi(\mathbf{x}) = \sum_{i=1}^{4} a_i h(w_i^\top \mathbf{x} + b_i)$, and we are given $n$ samples from observation: $\{(\mathbf{x}_k, y_k)\}_{k=1}^{n}$, where $y_i$'s are called "labels". By "Learning/Training", we mean to minimize the loss function

$$L(W, b, a) = \frac{1}{n} \sum_{k=1}^{n} (\phi(\mathbf{x}_k) - y_k)^2 = \frac{1}{n} \sum_{k=1}^{n} \left( \left( \sum_{i=1}^{4} a_i h(w_i^\top \mathbf{x}_k + b_i) \right) - y_k \right)^2$$

# Optimization for NNW

## Cont. with example

Suppose that the NNW is of the form $\phi(\mathbf{x}) = \sum_{i=1}^{4} a_i h(w_i^\top \mathbf{x} + b_i)$, and we are given $n$ samples from observation: $\{(\mathbf{x}_k, y_k)\}_{k=1}^{n}$, where $y_i$'s are called "labels". By "Learning/Training", we mean to minimize the loss function

$$L(W, b, a) = \frac{1}{n}\sum_{k=1}^{n}(\phi(\mathbf{x}_k) - y_k)^2 = \frac{1}{n}\sum_{k=1}^{n}\left(\left(\sum_{i=1}^{4} a_i h(w_i^\top \mathbf{x}_k + b_i)\right) - y_k\right)^2$$

Reference for one hidden layer NNW: https://cs230.stanford.edu/files/C1M3.pdf (by Andrew Ng).

# Gradient Descent

**Definition**

Let $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function, the Gradient Descent algorithm is

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha \nabla f(\mathbf{x}_t)$$

# Gradient Descent

## Definition

Let $f(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$ be a differentiable function, the Gradient Descent algorithm is

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha \nabla f(\mathbf{x}_t)$$

## Continuous time couterpart

$$\frac{d\mathbf{x}}{dt} = -\nabla f(\mathbf{x})$$

# Gradient Descent

## Definition

Let $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function, the Gradient Descent algorithm is

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha \nabla f(\mathbf{x}_t)$$

## Continuous time couterpart

$$\frac{d\mathbf{x}}{dt} = -\nabla f(\mathbf{x})$$

Both dynamical systems “stop” at $\mathbf{x}^*$ where $\nabla f(\mathbf{x}^*) = \mathbf{0}$.

# Convex Function

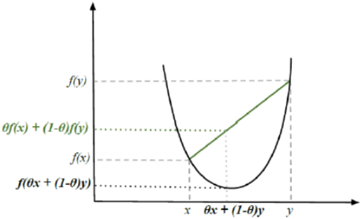
## Definition

$f(\mathbf{x})$ is convex if the domain is a convex set and for any $\mathbf{x}, \mathbf{y}$, $t \in [0, 1]$,

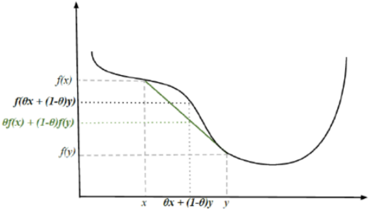
$$f(t\mathbf{x} + (1-t)\mathbf{y}) \leq tf(\mathbf{x}) + (1-t)f(\mathbf{y})$$

or

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x})$$




Convex function




Non-convex function

# Minimizing Convex Functions

### Lemma

*Let $f$ be differentiable and convex,* $\mathbf{x}^*$ *is a minimizer if and only if* $\nabla f(\mathbf{x}^*) = 0$.

A continuously differentiable function $f$ is $L$-smooth if its gradient is $L$-Lipschitz,

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L \|\mathbf{x} - \mathbf{y}\| .$$

It is often required that $\alpha \leq \frac{1}{L}$!!! The minimizer $\mathbf{x}^*$ is unique and gradient descent is guaranteed to converges to $\mathbf{x}^*$ if step size $\alpha < \frac{1}{L}$.

# Non-convex Function

Many functions are non-convex, e.g. the loss function of neural network. A non-convex function might have multiple local minima, local maxima, and saddle points.

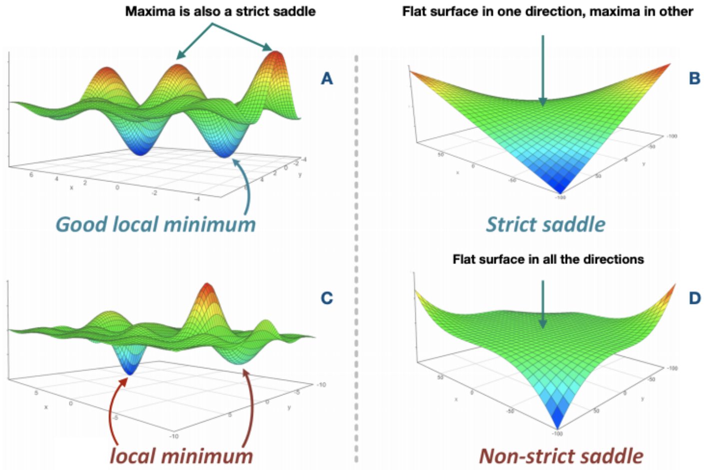
# Non-convex Function

Many functions are non-convex, e.g. the loss function of neural network. A non-convex function might have multiple local minima, local maxima, and saddle points.

**Definition**

- A point $\mathbf{x}^*$ is critical point of $f$ if $\nabla f(\mathbf{x}^*) = 0$;
- A critical point $\mathbf{x}^*$ of $f$ is a saddle point if for all neighborhood $U$ around $\mathbf{x}^*$ there are $\mathbf{y} \in U$ such that $f(\mathbf{z}) \leq f(\mathbf{x}^*) \leq f(\mathbf{y})$;
- A critical point $\mathbf{x}^*$ of $f$ is strict saddle if $\lambda_{\min}(\nabla^2 f(\mathbf{x}^*)) < 0$

# Non-convex Function




XW

# Non-convex Function

## Gradient Descent avoids strict saddle points

- With mild assumptions, the initial conditions s.t. GD converges to a saddle point lie on a set of measure zero. (Lee et al. 2016, 2019, Panageas and Piliouras 2017, Panageas et al. 2019)
- By adding perturbation properly,

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha(\nabla f(\mathbf{x}_t) + \xi_t), \quad \xi_t \sim \text{noise}$$

  GD can escape from saddle point efficiently. (Ge et al. 2015, Jin et al. 2017, 2019)

Reference: Lecture notes by Ioannis Panageas for course CS 295 at UC Irvine. https://panageas.github.io/teaching/

# Examples

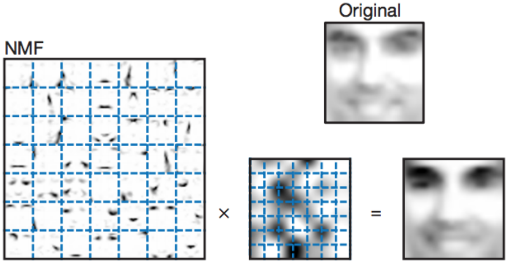
- Solving linear systems:
$$A\mathbf{x} = b$$
Find $\mathbf{x}^*$ such that $\|A\mathbf{x} - b\|^2$ is minimized.
- Matrix factorization:
$$V = WH$$
where $V$ is a given $n \times m$ matrix, $W$ is $n \times r$, and $H$ is $r \times m$. Find $W^*$ and $H^*$ so that the Frobenius norm $\|V - W^*H^*\|_F^2$ is minimized.

# Non-negative Matrix Factorization (NMF)

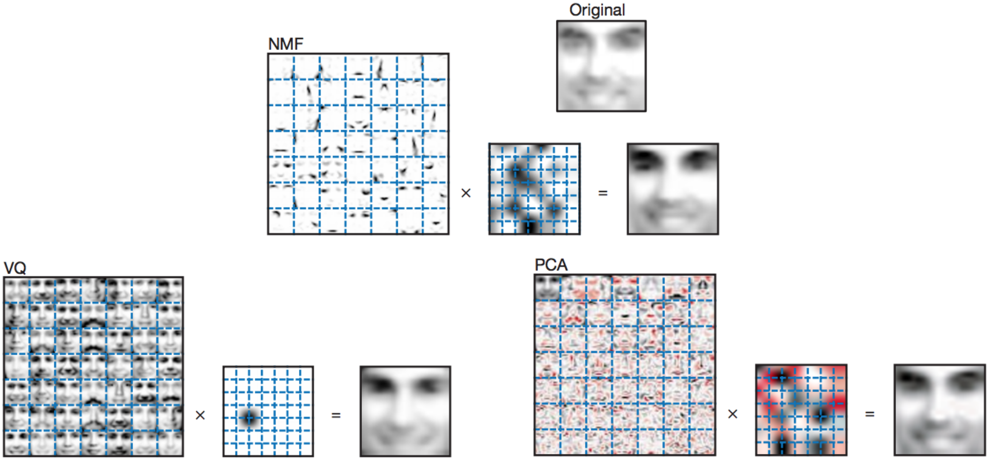
## Background

Lee and Seung, *Nature, 1999*:

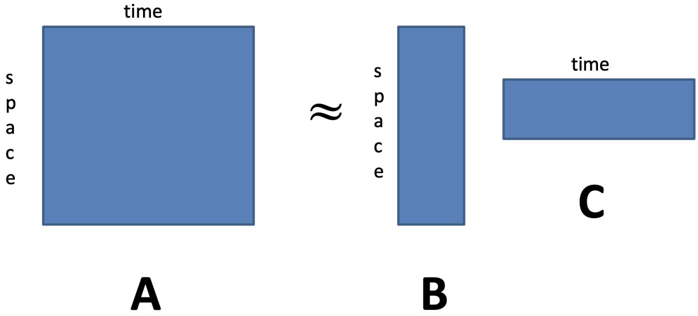
*"We have applied non-negative matrix factorization (NMF), together with principal component analysis (PCA) and vector quantization (VQ), to a database of facial images. ... The NMF basis is radically different: its images are localized features that correspond better with intuitive notions of the parts of faces."*



# NMF

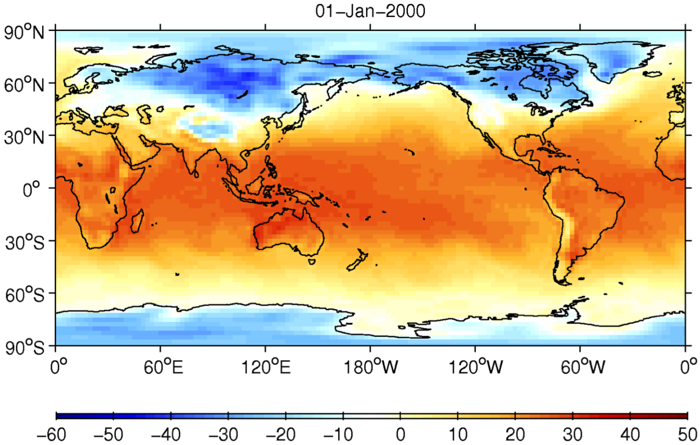


# NMF



# Global daily temperature

(10.512 points x 20.440 days)



# NMF

## Problem

In NMF, we are asked to decompose a non-negative data matrix $V \in \mathbb{R}_{+}^{n\times m}$ into the product of two non-negative matrix $W \in \mathbb{R}_{+}^{n\times r}$ and $H \in \mathbb{R}_{+}^{r\times m}$. In optimization viewpoint, we try to solve the minimization problem

$$\min_{W,H} F(W,H) = \|V - WH\|_F^2,$$

where $\|A\|_F^2 = \sum A_{ij}^2$ is the Frobenius norm.

- Vectorizing $W$ and $H$, $F(W,H)$ is a non-convex function;
- Non-negative constraint, i.e.,$W$ and $H$ are vectors in the non-negative orthant.

# Projected Gradient Descent

The previous session concerns about *unconstrained optimization problems*, i.e. the domain of $f$ is the whole $\mathbb{R}^n$. But there are problems like NMF which has constraints. In this subsection we discuss how to solve constrained optimization problem:

$$\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}),$$

with *Projected Gradient Descent*.

# Projected Gradient Descent

The previous session concerns about *unconstrained optimization problems*, i.e. the domain of $f$ is the whole $\mathbb{R}^n$. But there are problems like NMF which has constraints. In this subsection we discuss how to solve constrained optimization problem:

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}),$$

with *Projected Gradient Descent*.

## Definition

The projection of a point $\mathbf{y}$, onto a set $\mathcal{X}$ is defined as the nearest point in the set to $\mathbf{y}$.

$$P_{\mathcal{X}}(\mathbf{y}) = argmin_{\mathbf{x}\in\mathcal{X}} \|\mathbf{x}-\mathbf{y}\|^2.$$

Let $f:\mathbb{R}^n \to \mathbb{R}$ be a differentiable function in come convex set $\mathcal{X}$. The Projected gradient descent is given by

$$\mathbf{x}_{t+1} = P_{\mathcal{X}}(\mathbf{x}_t - \alpha\nabla f(\mathbf{x}_t)).$$

# Ending

## Further topics

- Stochastic Gradient Descent;
- MinMax Optimization;
- Game Theory;
- Optimization on Manifold.

## Reading materials

- Notes for "Optimization for Machine Learning" by Chi Jin (Princeton), https://sites.google.com/view/cjin/ee539cos512
- Game Theory, by Tim Roughgarden (Columbia), http://timroughgarden.org/
- Optimization on Manifold, by Nicolas Boumal (EPFL), http://sma.epfl.ch/ nboumal/

Thank You!