Workshop 7

COMP90051 Statistical Machine Learning

Semester 2, 2024

Learning Outcomes

By the end of this workshop you should be able to:

1. Understand and implement different optimization algorithms
2. Understand the Pros and Cons of different common used optimization algorithms
3. Be able to use optimizer in Pytorch
4. Understand and implement autoencoder

Optimization Algorithms

A common pipeline in ML is:

特例. ridge, linear ⇒ analytic solution

1. Define the model

不用 opt Alg 的

2. Define the loss function
3. Optimize the model parameter w.r.t the loss function

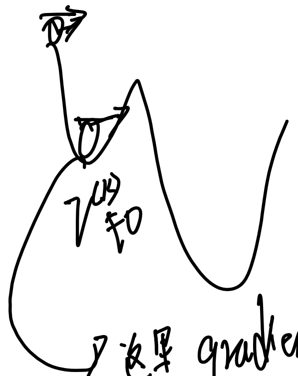
How to optimize/update the model parameters?

Optimization algorithms: GD, SGD, AdaGrad, Adam, …

adaptive

non convex

$V^{(0)}$ momentum base　　update momentum depend on gradient.



这里 gradient $=0$, but $V^{(k)} \neq 0$

momentum

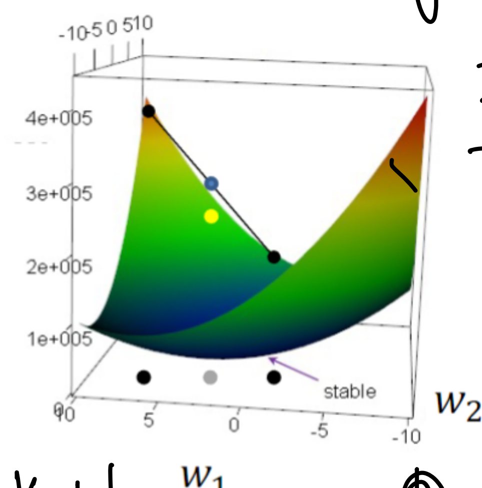
Messy Loss Function

In general, the loss function is complicated (not convex nor concave), especially for Deep Learning models.
How to update the parameter is important!

$\|y - Xw\|^2$ convex quadratic

最小化 指定

- Recall linear regression, convex 'Bowl shaped' objective
  - gradient descent finds a **global** optimum



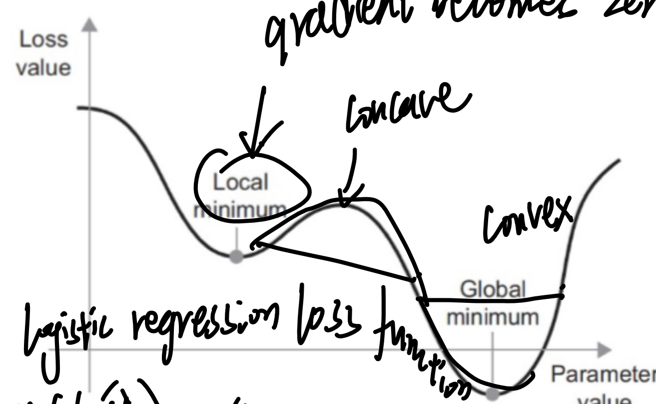
global optimum
minimum

- In contrast, most DNN objectives are not convex
  - gradient methods get trapped in local optima or saddle points

gradient becomes zero

gradient descent converges

stop



concave

convex

always convex for logistic regression loss function

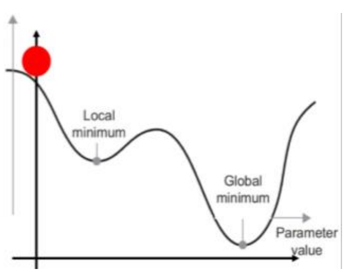
$\mathcal{L} = -y\log\hat{y} - (1-y)\log(1-\hat{y})$, $\hat{y} = \sigma(w^T x)$ → sigmoid

Better than GD/SGD?!

Momentum

"How to escape local minimum?"

Consider a ball with mass rolling down the loss function surface.



* $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \boldsymbol{v}^{(t)}$
* $\boldsymbol{v}^{(t)} = \alpha \boldsymbol{v}^{(t-1)} + \eta \nabla L(\boldsymbol{\theta}^{(t)})$
* α decays the velocity, e.g., 0.9

momentum decays over time

AdaGrad

"Why update all parameter with the same rate?"

Frequent features should update less, while infrequent features should update more.

梯度和

* $g_i^{(t)} = g_i^{(t-1)} + \nabla L(\boldsymbol{\theta}^{(t)})_i^2$
* $\theta_i^{(t+1)} = \theta_i^{(t)} - \frac{\eta}{\sqrt{g_i^{(t)} + \epsilon}} \nabla L(\boldsymbol{\theta}^{(t)})_i$

跳出local optimum

Adam

"How about we combine these two together?"

Combining elements from momentum and adaptive learning rate, most common used optimizer.

* $\boldsymbol{m}^{(t)} = \beta_1 \boldsymbol{m}^{(t-1)} + (1 - \beta_1) \nabla L(\boldsymbol{\theta}^{(t)})$
* $\boldsymbol{v}^{(t)} = \beta_2 \boldsymbol{v}^{(t-1)} + (1 - \beta_2) \nabla L(\boldsymbol{\theta}^{(t)})^2$
* $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \frac{\eta}{\sqrt{\boldsymbol{v}^{(t)} / _{1-\beta_2}} + \epsilon} \boldsymbol{m}^{(t)} / _{1-\beta_1}$
* $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$

g(t) 和 momentum 的结合、

前两种的结合

Autoencoder: Why reconstruct the data itself??

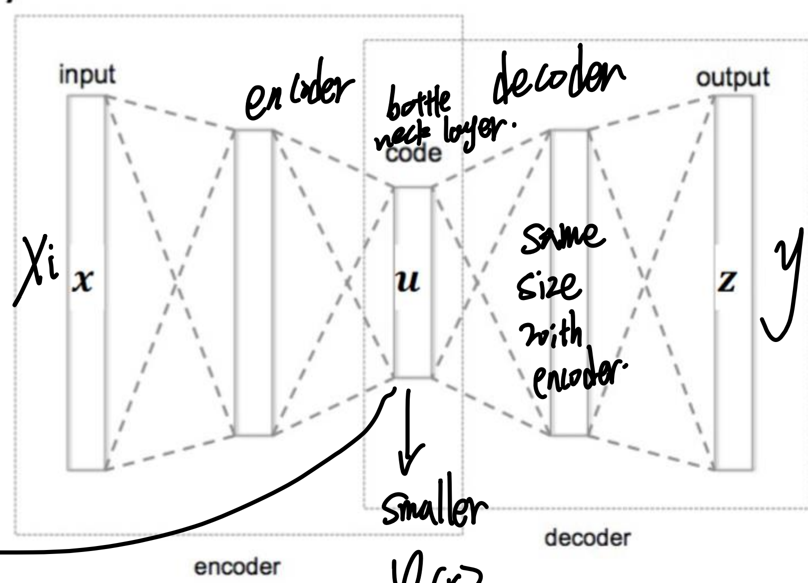
build encoder to shrink the data

*The encoder gives good **low-dimensional representation** of the data which could be used for learning tasks afterwards.*
*Autoencoder is **unsupervised**! No need for labelling, save time and money!*

- Given data without labels $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, set $\boldsymbol{y}_i \equiv \boldsymbol{x}_i$ and train a DNN to predict $\boldsymbol{z}(\boldsymbol{x}_i) \approx \boldsymbol{x}_i$
- Set bottleneck layer $\boldsymbol{u}$ in middle "thinner" than input, and/or
  - corrupt input $\boldsymbol{x}$ with noise
  - regularise s.t. $\boldsymbol{u}$ is sparse
  - regularise to contract inputs

autoencoder

类似PCA
去
降低维度还想让
他与原始数据有相同
表达能力.



adapted from Chervinskii at Wikimedia Commons (CC4)

$y_i = x_i$ $\|y - f(x)\|^2$

$= \sum_i \|y_i - f(x_i)\|^2$

Worksheet 7

$g^k = \nabla_w(L)$ gradient vector at iteration k

$(g^k)^T$

$$G^i = \left(\sum_{k=1}^{i} (g_1^k)^2, \sum_{k=1}^{i} (g_2^k)^2, \ldots\ldots\right)$$

G_i → diagonal version

sum of square of gradient

$$w^{i+1} = w^i - \frac{\eta}{\sqrt{G_i^i}} \cdot g^i$$

$g^k \times g^k$ ←

$$G^i = \begin{bmatrix} \vdots \\ g^k \\ \vdots \end{bmatrix} \begin{bmatrix} \cdots & g^k \end{bmatrix} = \begin{bmatrix} G \end{bmatrix}$$ → 只看对角线

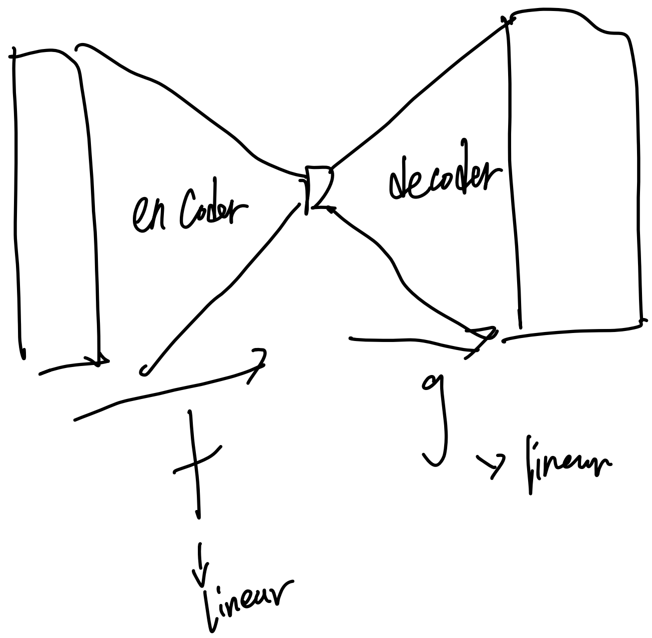
$\cancel{n\times 1 \quad 1\times n}$

matrix 只看

momentum

w is updated by $\begin{cases} \text{velocity} \\ \text{gradient} \end{cases}$ together.

Adam最好 用



$h(x) = \cancel{f \circ f(x)} \quad g \circ f(x)$