AI - FOUNDATION AND APPLICATION

Instructor:

Assoc. Prof. Dr. Truong Ngoc Son

Chapter 3
Optimazation of learning process

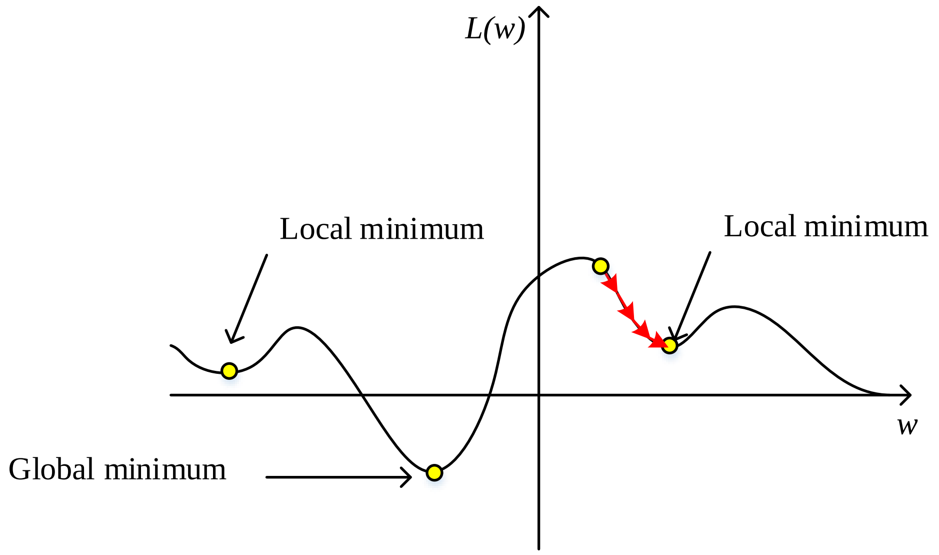
Outline

- ❑ The challenges in Deep learning
- ❑ Momentum
- ❑ ADAGRAD – Adaptive Gradient Descent
- ❑ RMSPROP (Root Mean Squared Propagation)
- ❑ ADAM
- ❑ Dropout

The challenge in Deep Learning

Local minima

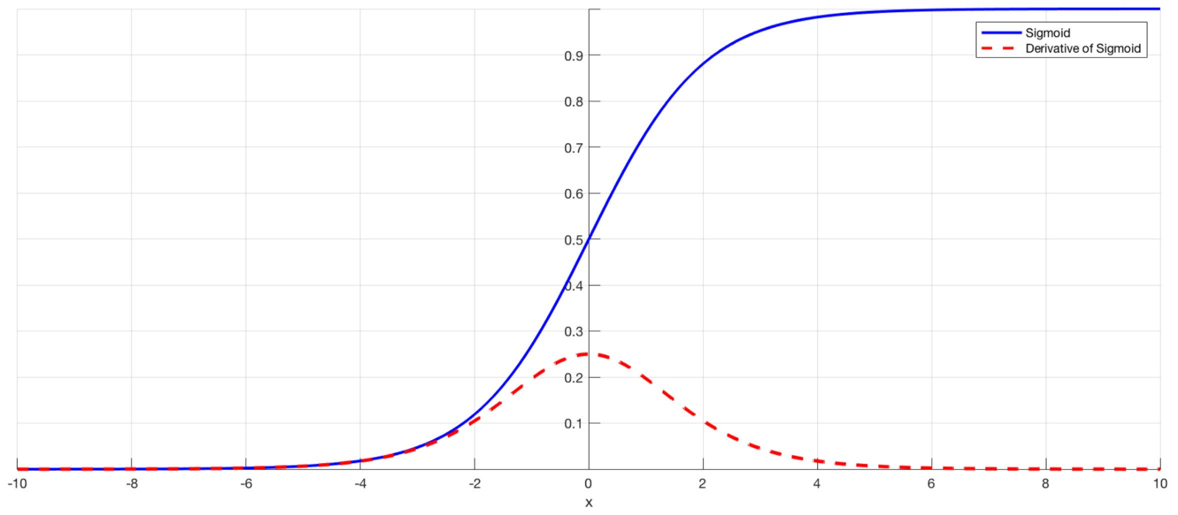
- ❑ The objective function of deep learning usually has many local minima
- ❑ When the numerical solution of an optimization problem is near the local optimum, the numerical solution obtained by the final iteration may only minimize the objective function locally, rather than globally, as the gradient of the objective function's solutions approaches or becomes zero



The challenge in Deep Learning

Vanishing Gradient

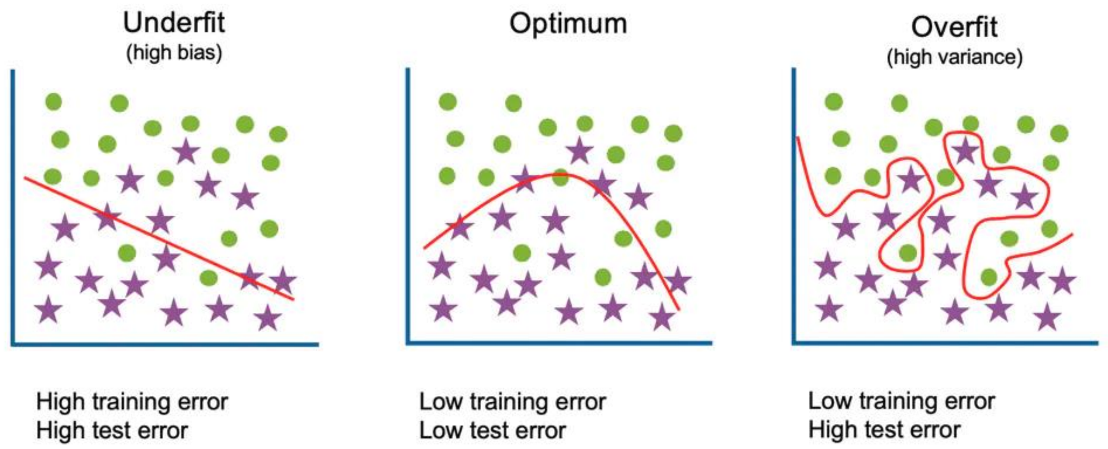
- ❑ As more layers using certain activation functions are added to neural networks, the gradients of the loss function approaches zero, making the network hard to train
- ❑ The simplest solution is to use other activation functions, such as ReLU, which doesn't cause a small derivative.
- ❑ Residual networks are another solution, as they provide residual connections straight to earlier layers



The challenge in Deep Learning

Over fitting and under fitting

- ❑ Overfitting is a modeling error in statistics that occurs when a function is too closely aligned to a limited set of data points. As a result, the model is useful in reference only to its initial data set, and not to any other data sets
- ❑ The model fit well the training data, but it not show the good performance with the testing data
- ❑ Underfitting is a scenario in data science where a data model is unable to capture the relationship between the input and output variables accurately, generating a high error rate on both the training set and unseen data



Momentum

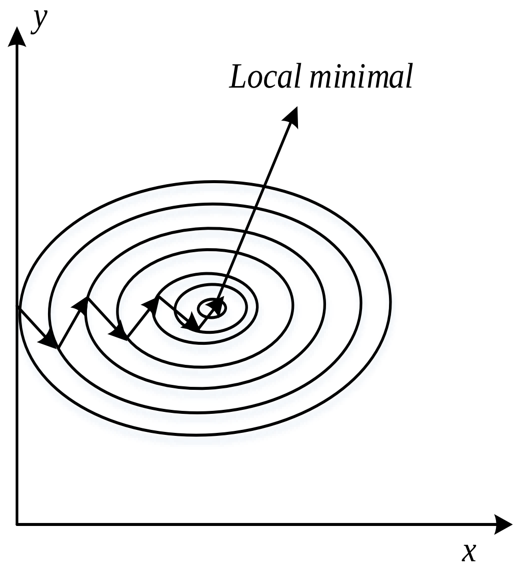
- The method of momentum is designed to accelerate learning.
- The momentum algorithm accumulates exponentially decaying moving average of past gradient and continues to move in their direction

Gradient descent

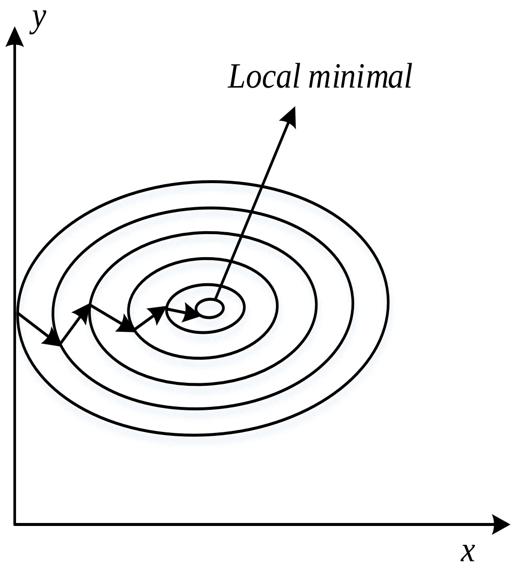
$$w_t = w_t - \Delta w_t$$
$$\Delta w_t = \eta \nabla \mathrm{C}(w)$$

Gradient descent with momentum

$$\Delta w_t = \eta \nabla \mathrm{C}(w) + \gamma \Delta w_{t-1}$$



(a) Không sử dụng momentum



(a) Sử dụng momentum

ADAGRAD – Adaptive Gradient Descent

- ❑ Decay the learning rate for parameters in proportion to their update history
- ❑ Adapts the learning rate to the parameters, performing smaller updates (low learning rates) for parameters associated with frequently occurring features, and larger updates (high learning rates) for parameters associated with infrequent features
- ❑ It is well-suited for dealing with sparse data
- ❑ Adagrad greatly improved the robustness of SGD and used it for training large-scale neural nets

$$w_t = w_t - \eta' \frac{\partial \mathrm{L}}{\partial w_{t-1}}$$

Where

$$\eta' = \frac{\eta}{\sqrt{\alpha_t + \varepsilon}}$$

$$\alpha_t = \sum_{i=1}^{t} \left(\frac{\partial \mathrm{L}}{\partial w_{t-1}} \right)^2$$

RMSProp (Root Mean Squared Propagation)

- ❑ Adapts the learning rate to the parameters
- ❑ Divide the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight

$$w_t = w_t - \eta' \frac{\partial L}{\partial w_{t-1}}$$

Where

$$\eta' = \frac{\eta}{\sqrt{\alpha_t + \varepsilon}}$$

$$\alpha_t = \beta \alpha_{t-1} + (1 - \beta)\left(\frac{\partial L}{\partial w_{t-1}}\right)^2$$

Adam — Adaptive Moment Estimation

- ADAM combines two stochastic gradient descent approaches, Adaptive Gradients, and Root Mean Square Propagation
- Adam also keeps an exponentially decaying average of past gradients similar to SGD with momentum

$$v_{dW} = \beta_1 v_{dW} + (1 - \beta_1) \frac{\partial \mathrm{L}}{\partial w}$$

$$s_{dW} = \beta_2 s_{dW} + (1 - \beta_2) \frac{\partial \mathrm{L}}{\partial w}$$

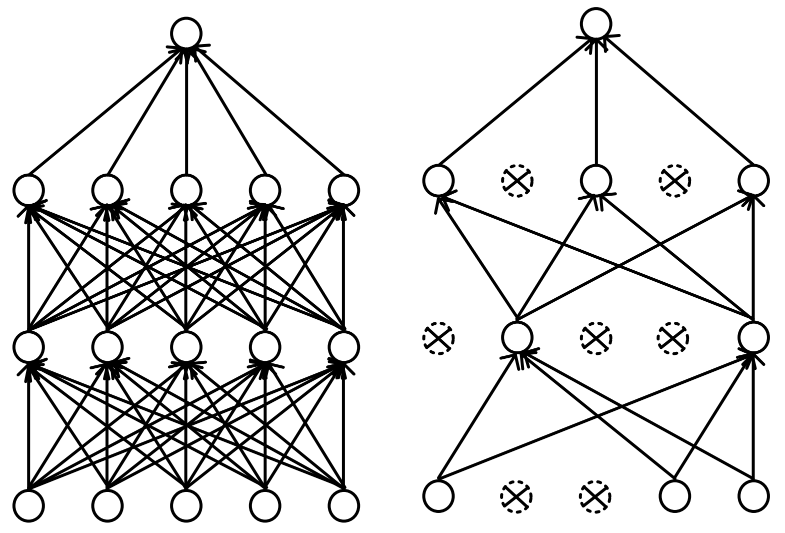
$$\hat{v}_{dW} = \frac{v_{dW}}{1 - (\beta_1)^t}$$

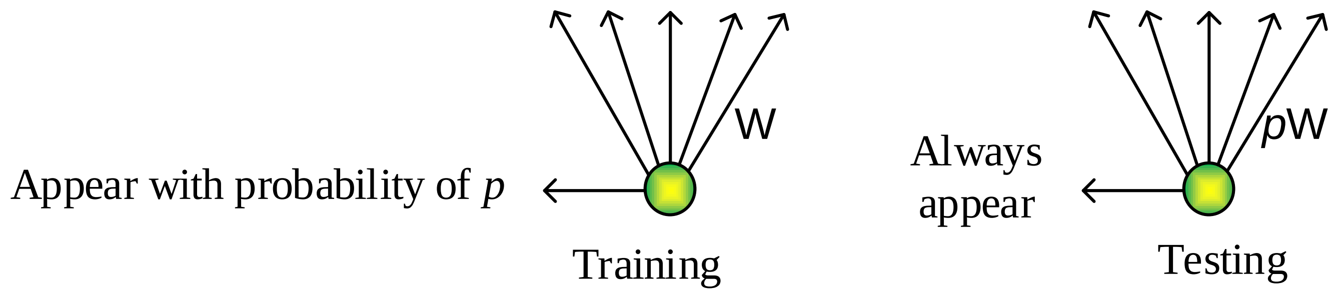
$$\hat{s}_{dW} = \frac{s_{dW}}{1 - (\beta_2)^t}$$

$$W = W - \eta \frac{\hat{v}_{dW}}{\sqrt{\hat{s}_{dW}} + \varepsilon}$$

Dropout

- ❑ Avoid overfitting problem
- ❑ Probabilistically dropping out nodes in the network is a simple and effective regularization method
- ❑ Dropout is implemented per-layer in a neural network
- ❑ A common value is a probability of 0.5 for retaining the output of each node in a hidden layer

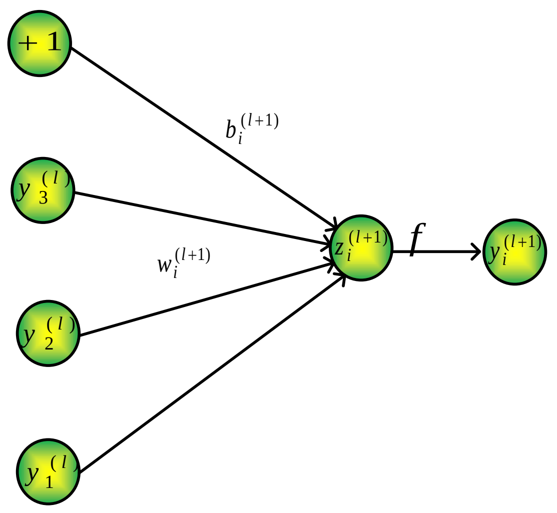




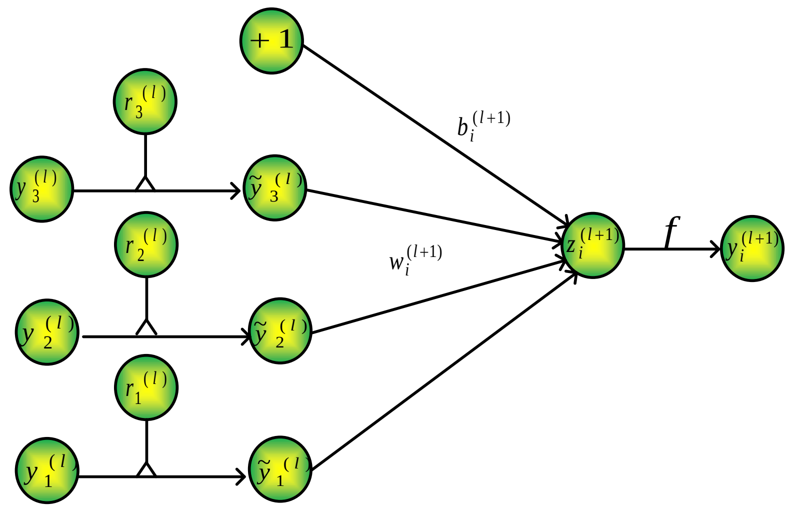
Dropout

- ❑ How to apply dropout

$$z_i^{(l+1)} = w_i^{(l+1)} y^l + b_i^{(l+1)}$$
$$y_i^{(l+1)} = f(z_i^{(l+1)})$$



Standard NN

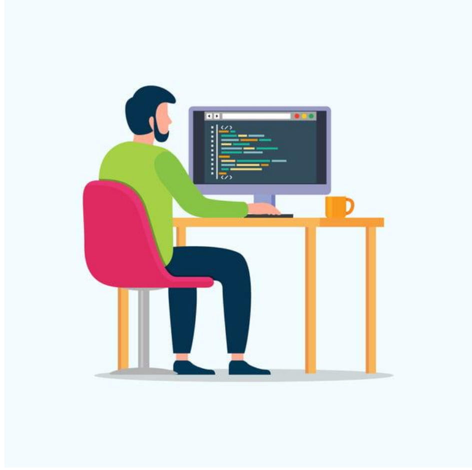


Dropout NN

$$r_j^{(l)} \sim Bernoulli(p)$$
$$\tilde{y}^{(l)} = r^{(l)} * y^{(l)}$$
$$z_i^{(l+1)} = w_i^{(l+1)} \tilde{y}^l + b_i^{(l+1)}$$
$$y_i^{(l+1)} = f(z_i^{(l+1)})$$

PYTHON CODE

Assignments



- ❑ Design a multilayer neural network, apply the optimizations of learning algorithms.
- ❑ (input layer, 2 hidden layers (sigmoid,ReLU) , output layer)
- ❑ Optimization: Momentum, Adagrad, Dropout (+ Momentum, Adagrad)
- ❑ Compare : Accuracy, (Converging time)
- ❑ Dataset: MNIST

Assignments

- ❑ Week 8: Submit assignment
- ❑ Week 9: Quiz, decision of final project
- ❑ Week 10-14: CNN
- ❑ Week 15-17: Final project present