ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

Lecture 2
Principle of Learning in Neural Networks

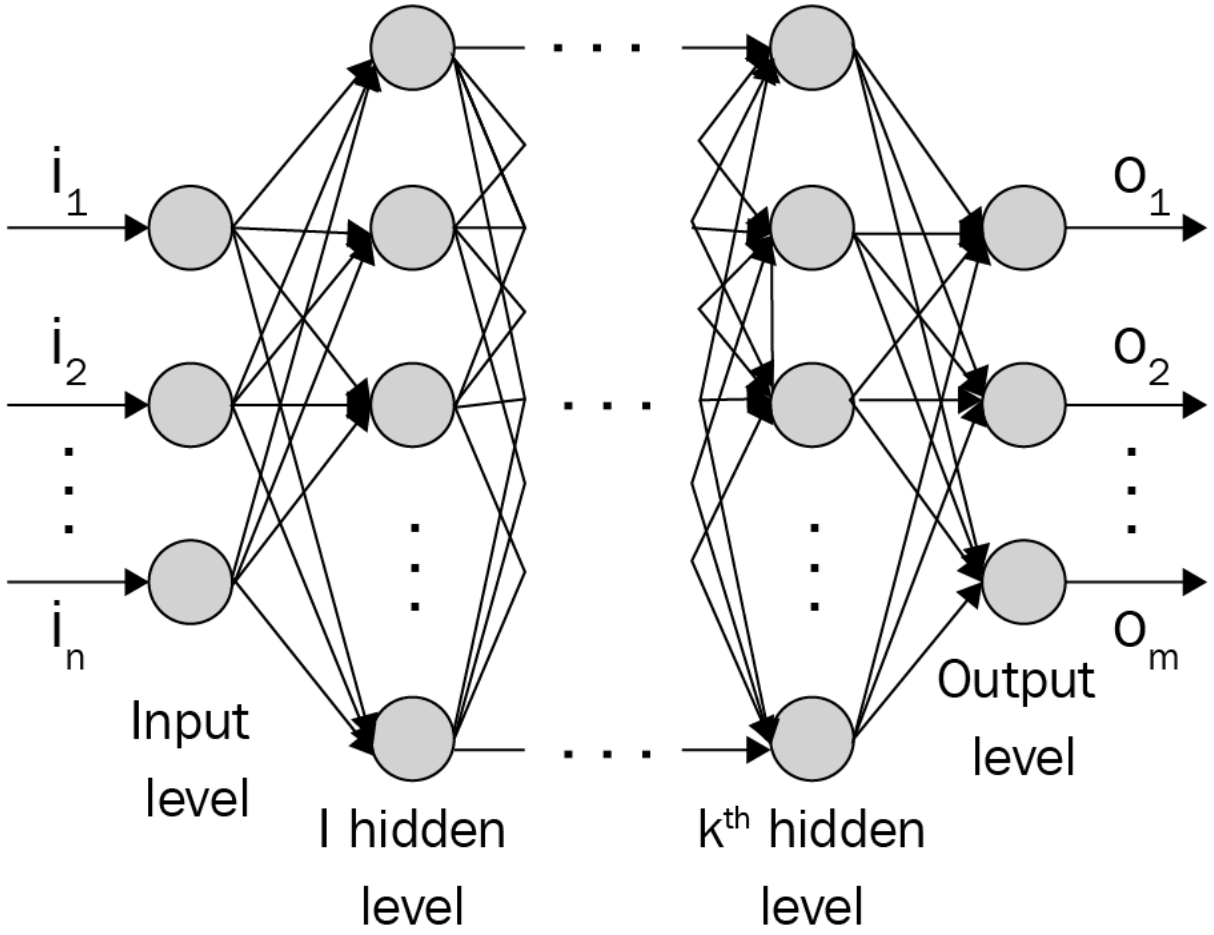
LÊ ANH CƯỜNG

Ton Duc Thang University

Outline

1. Objective of Learning
2. Classification and Regression problem
3. Loss functions and Minimize Empirical Risk
4. Gradient Descent Algorithms
5. Optimizers
6. Prevent overfitting

Feed Forward Neural Network



The General Model of Learning from Examples

- Suppose that there is a functional relationship between two sets of objects X and Y:

$$f: X \rightarrow Y$$

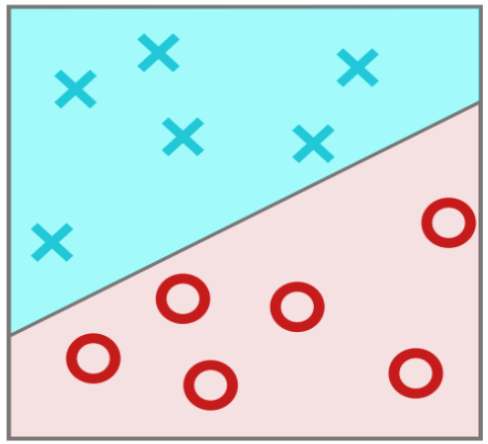
- Given a finite set of examples:

$$D = \{(\mathbf{x}_i, y_i) \mid i=1,2,\ldots,N\} \text{ , where } \mathbf{x}_i \in X \text{ and } y_i \in Y$$

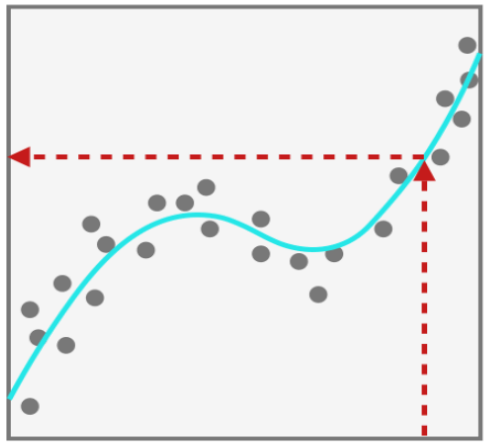
- The task here is to derive (i.e. to learn) the objective function f

Classification and Regression

- $y = f(\mathbf{x})$
- If y is the real value ie Y = R then we have a regression problem
- If y is a value in a given finite discrete set, then we have a classification problem



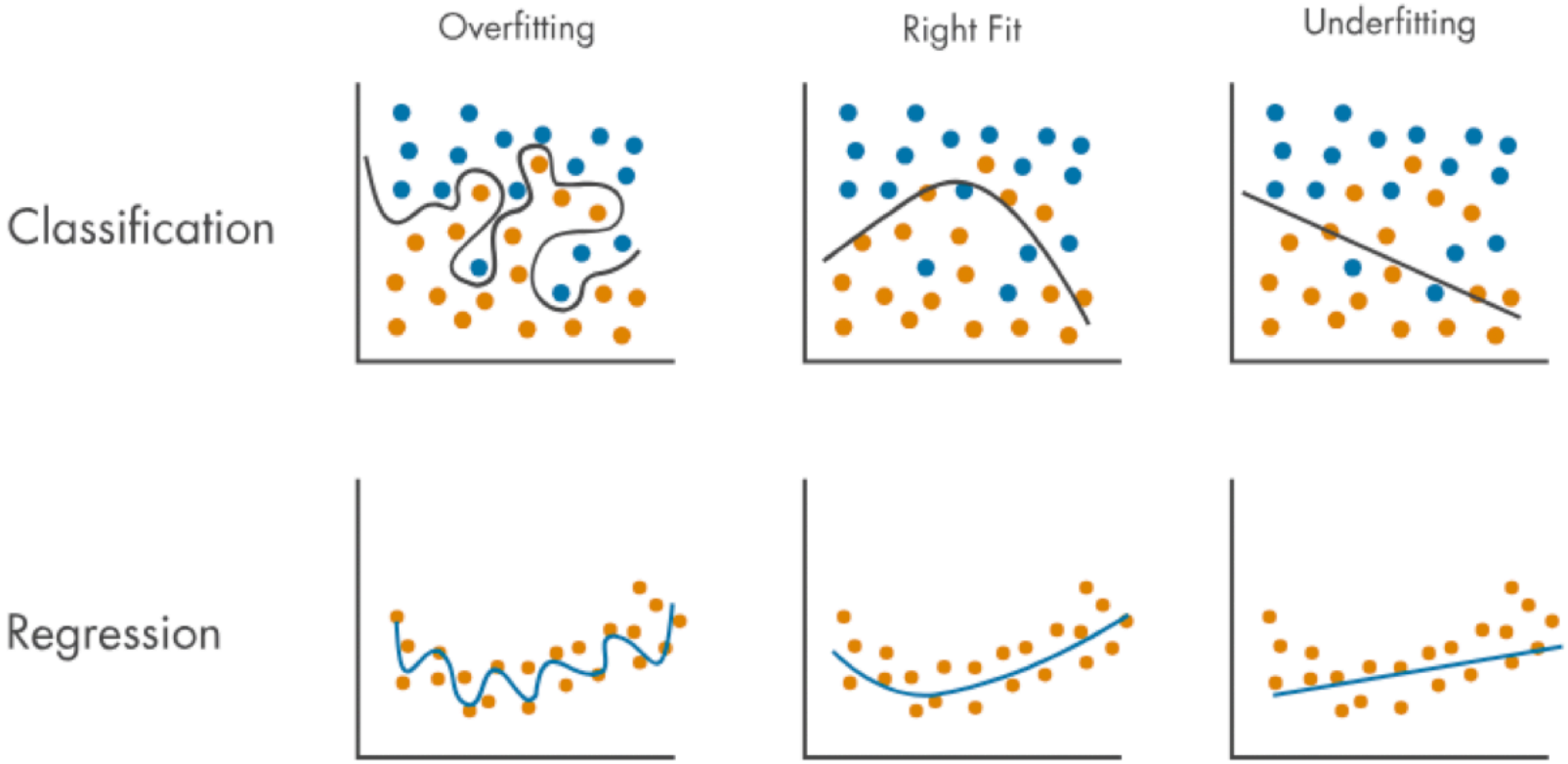
Classification



Regression

Objective of Learning

- Learn to generalize from a finite set of examples
- The learnt function then can predict output *y* given a new input **x**



Data Representation

- **x** is a vector of features

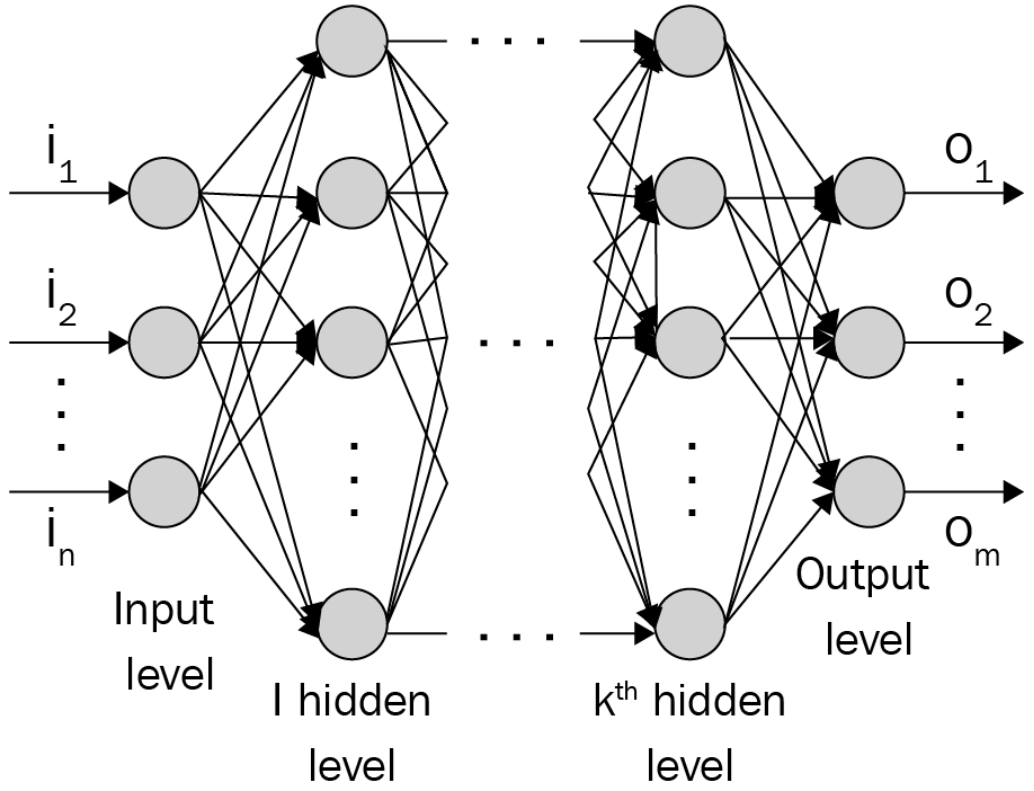
$$\mathbf{x} = (x_1, x_2, ..., x_d)$$

$$\mathbf{X} = R^d$$

- y is a real number in the regression problem
- y is in classification problem:
  - binary classification, y = {0,1} or {-1,+1}
  - multiple classes: y = {1, 2, ..., k} or one-hot vector (0,...0,1,0,...0)

One-hot vector

- Multiple classes: y = {1, 2, ..., k}
  or one-hot vector (0,...0,1,0,...0)



One-hot vector

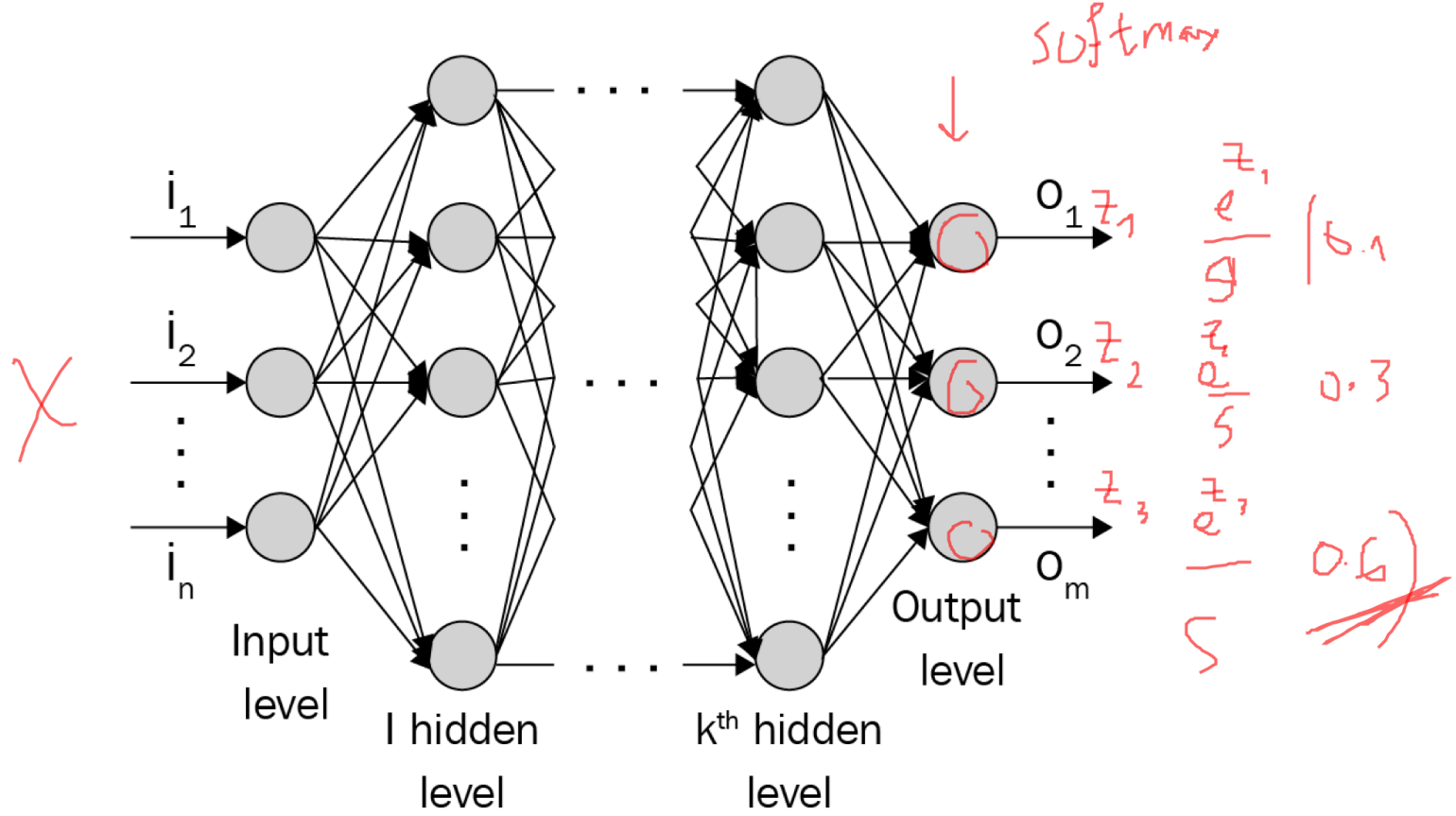
- Multiple classes: y = {1, 2, ..., k}
  or one-hot vector (0,...0,1,0,...0)

a b c

a : (1, 0, 0)

b : (0, 1, 0)

c (0, 0, 1)



Loss function

- Suppose that ($\boldsymbol{x}$,y) is an example. We want to find the difference between the ground true value y and the predicted value h($\boldsymbol{x}$)
- For regression:

$$L(y,h(x)) = \left(y - h(x)\right)^2$$

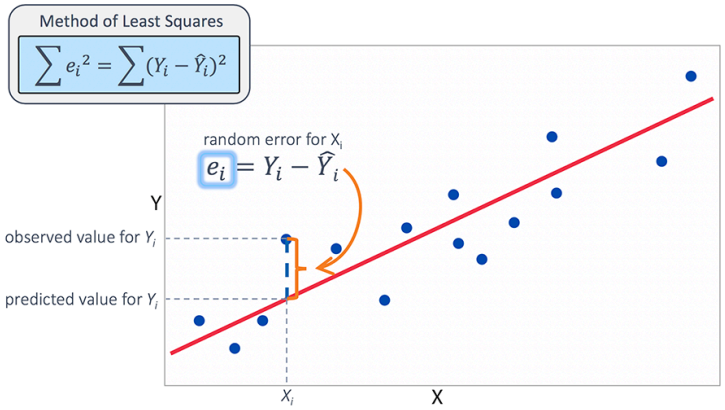
- For classification:

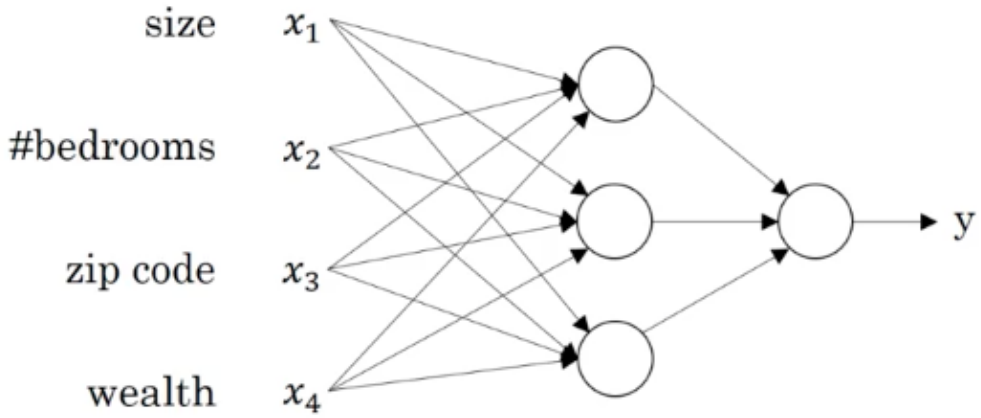
$$L(y,h(x)) = \begin{cases} 0 & \text{if } y = h(x) \\ 1 & \text{if } y \neq h(x) \end{cases}$$

MSE for loss function

MSE: Mean Square Error

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$





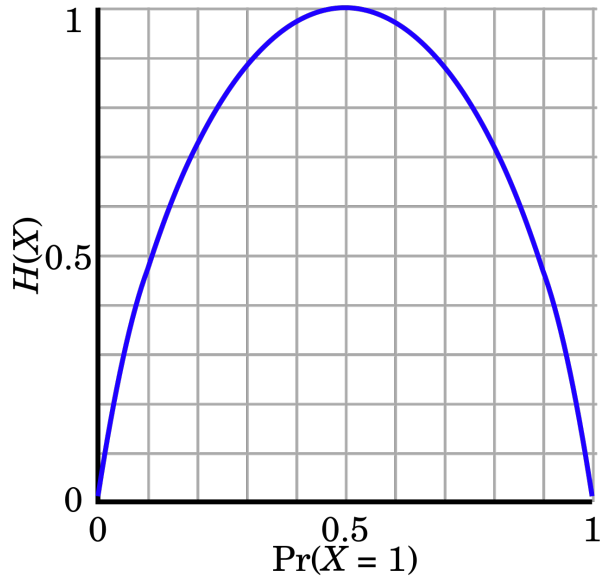
Cross Entropy for multi-class classification

- Entropy
- Kullback–Leibler Divergence
- Cross Entropy

Entropy

Entropy is a **measure** of the unpredictability of the state, or equivalently, of its average **information** content.

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i)$$



p=0.7, then

$$\begin{aligned}
\mathrm{H}(X) &= -p \log_2(p) - q \log_2(q) \\
&= -0.7 \log_2(0.7) - 0.3 \log_2(0.3) \\
&\approx -0.7 \cdot (-0.515) - 0.3 \cdot (-1.737) \\
&= 0.8816 < 1
\end{aligned}$$

$$\begin{aligned}
\mathrm{H}(X) &= -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i) \\
&= -\sum_{i=1}^{2} \frac{1}{2} \log_2 \frac{1}{2} \\
&= -\sum_{i=1}^{2} \frac{1}{2} \cdot (-1) = 1
\end{aligned}$$

Kullback–Leibler Divergence

x	0	1	2
Distribution *P*(x)	0.36	0.48	0.16
Distribution *Q*(x)	0.333	0.333	0.333

$$
\begin{aligned}
D_{\mathrm{KL}}(P \parallel Q) &= \sum_{x \in \mathcal{X}} P(x) \ln\left(\frac{P(x)}{Q(x)}\right) \\
&= 0.36 \ln\left(\frac{0.36}{0.333}\right) + 0.48 \ln\left(\frac{0.48}{0.333}\right) + 0.16 \ln\left(\frac{0.16}{0.333}\right) \\
&= 0.0852996
\end{aligned}
$$

Interpretations [edit]

The Kullback–Leibler divergence from Q to P is often denoted $D_{\mathrm{KL}}(P \parallel Q)$.

In the context of machine learning, $D_{\mathrm{KL}}(P \parallel Q)$ is often called the information gain achieved if Q is used instead of P. By analogy with information theory, it is also called the **relative entropy** of P with respect to Q. In the context of coding theory, $D_{\mathrm{KL}}(P \parallel Q)$ can be constructed by measuring the expected number of extra bits required to code samples from P using a code optimized for Q rather than the code optimized for P.

Cross Entropy for Loss Function

The cross entropy formula takes in two distributions, $p(x)$, the true distribution, and $q(x)$, the estimated distribution, defined over the discrete variable x and is given by

$$H(p, q) = -\sum_{\forall x} p(x) \log(q(x))$$

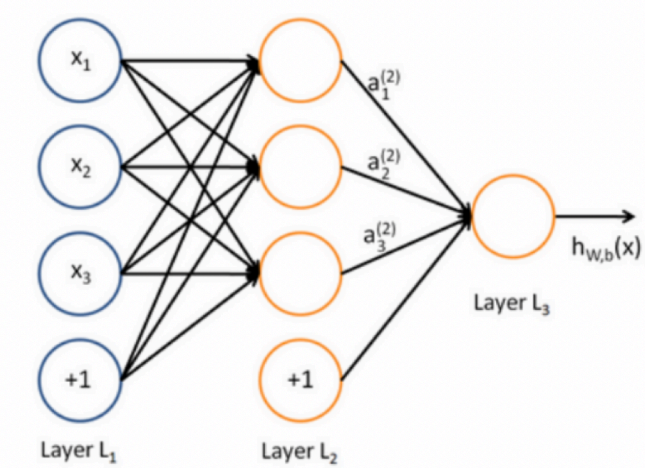
$$H(p, q) = H(p) + D_{\mathrm{KL}}(p \| q)$$

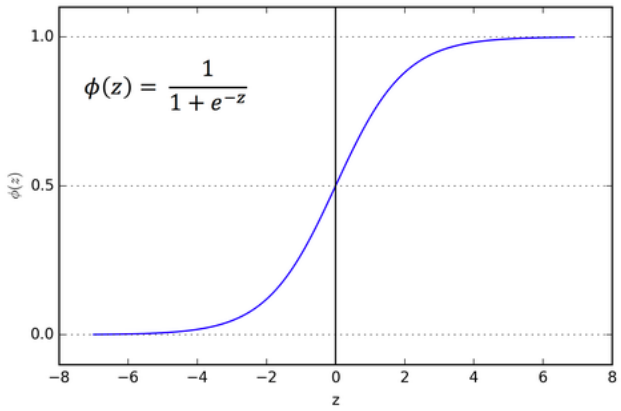
$$H(p, q) = -\sum_{\forall x} p(x) \log(q(x))$$

Cross Entropy for Loss Function

Having set up our notation, $p \in \{y, 1-y\}$ and $q \in \{\hat{y}, 1-\hat{y}\}$, we can use cross entropy to get a measure of dissimilarity between p and q:

$$H(p,q) = -\sum_i p_i \log q_i = -y \log \hat{y} - (1-y) \log(1-\hat{y})$$





Summary

- Objective of learning?
- Criteria for learning?
  - Loss function
  - Aims to minimize the loss function -> fit the model to training examples
- How to define the loss functions for:
  - Regression
  - Classification

Expected Risk and Empirical Risk

- Expected risk/loss is the mean of L(y,h(**x**)) over the whole space ***X*** x ***Y***

$$R(h) = \iint L(y, h(x))p(x, y)dxdy$$

- Empirical risk/loss is the mean of L(y,h(**x**)) over the training dataset D

$$R_{emp}(h) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, h(x_i))$$

Empirical Risk

- For regression:

$$R_{emp}(h) = \frac{1}{N}\sum_{i=1}^{N}(y_i - h(x_i))^2$$

- For classification:

$$R_{emp}(h) = \frac{1}{N}\sum_{i=1}^{N}\delta(y_i, h(x_i))$$

Empirical risk minimization inductive principle. (Nguyên lý quy nạp cực tiểu sai số thực nghiệm)

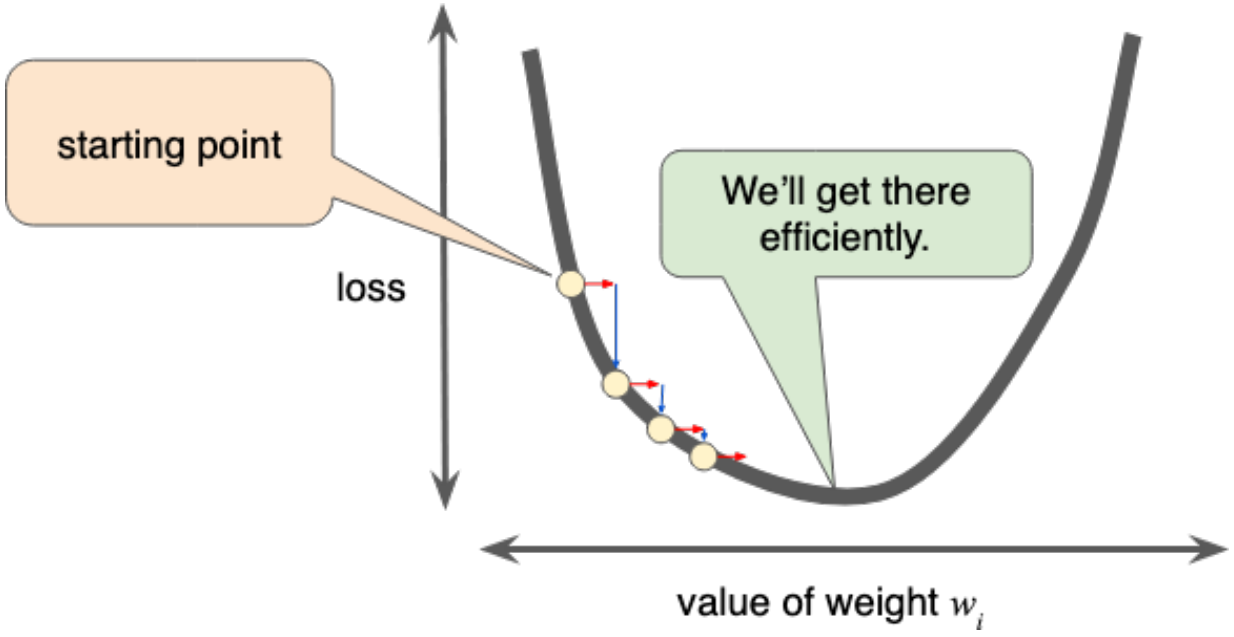
- We will consider the objective function f by the approximation function g as follows:

$$\mathbf{H} = \{h : X \to Y\}$$

$$g = \arg\min_{h} R_{emp}(h)$$

Gradient Descent Algorithm

Linear Regression: Learning by Gradient Descent



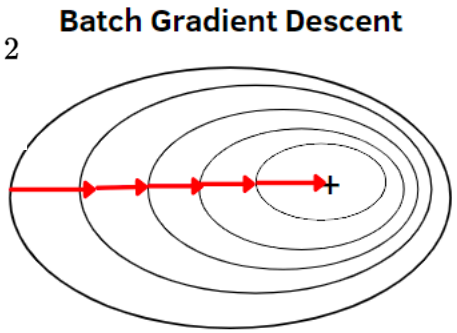
Update parameters:

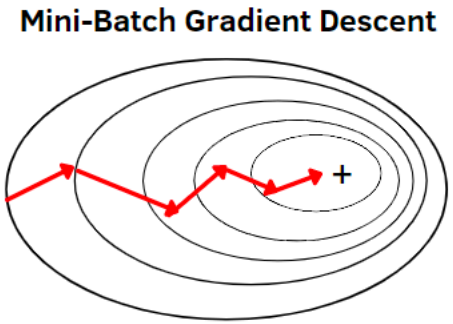
Learning rate

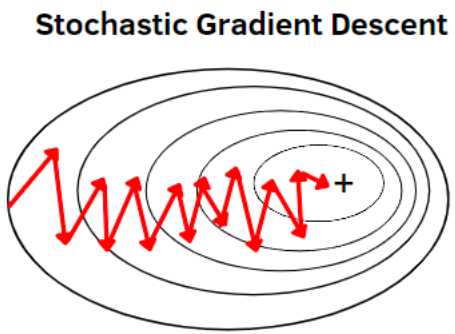
$$w_i = w_i - \mu \frac{\partial L}{\partial w_i}$$

Stochastic Gradient Descent (SGD)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

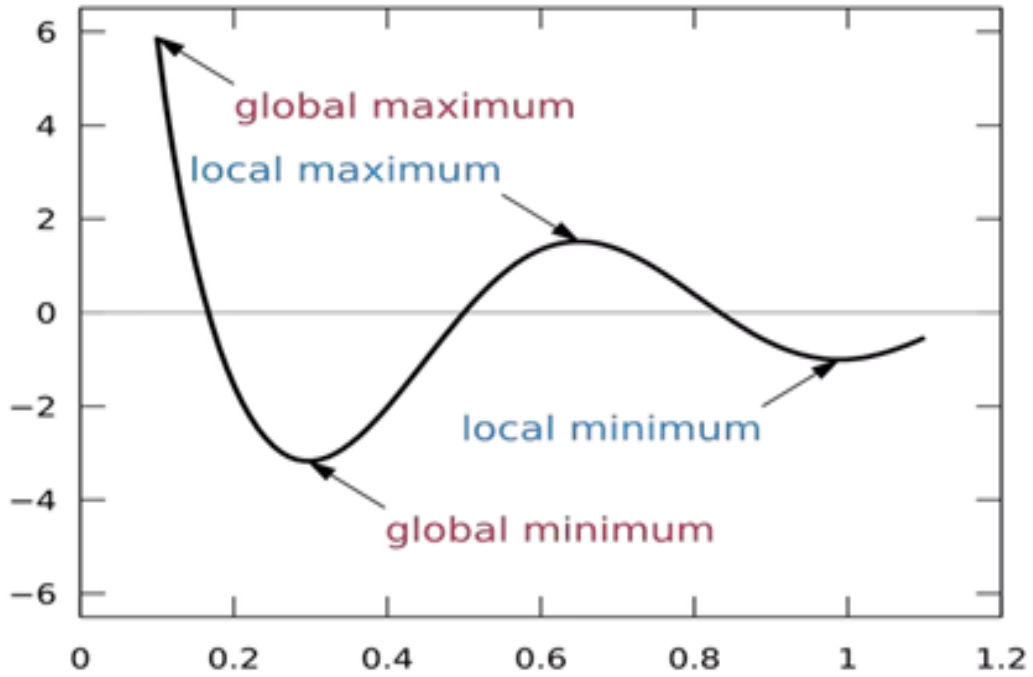






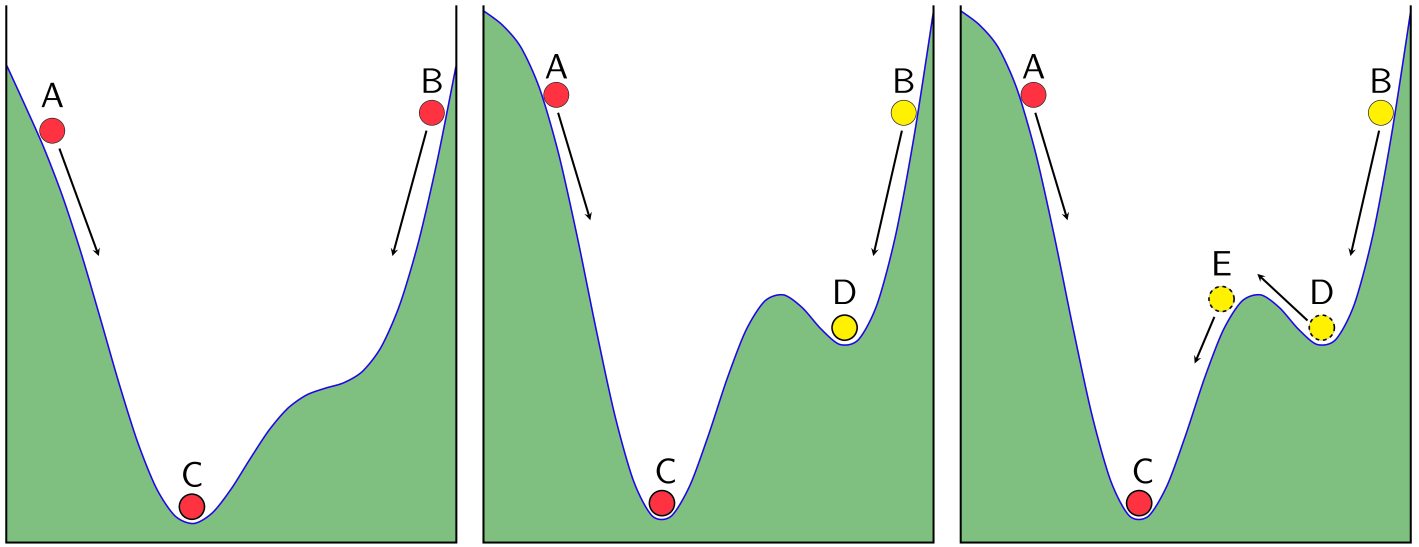
https://www.analyticsvidhya.com/blog/2022/07/gradient-descent-and-its-types/

Issues with Gradient Descent Algorithm



Momentum

- Momentum is an extension to the gradient descent optimization algorithm that allows the search to build inertia in a direction in the search space and overcome the oscillations of noisy gradients and coast across flat spots of the search space.



https://machinelearningcoban.com/2017/01/16/gradientdescent2/

Momentum

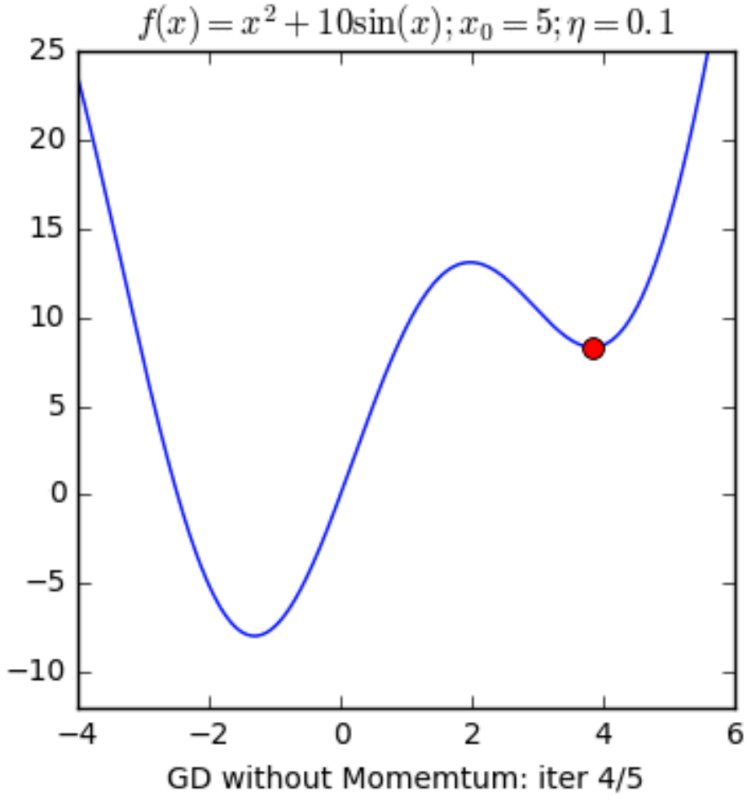
$$\theta_{t+1} = \theta_t - v_t.$$

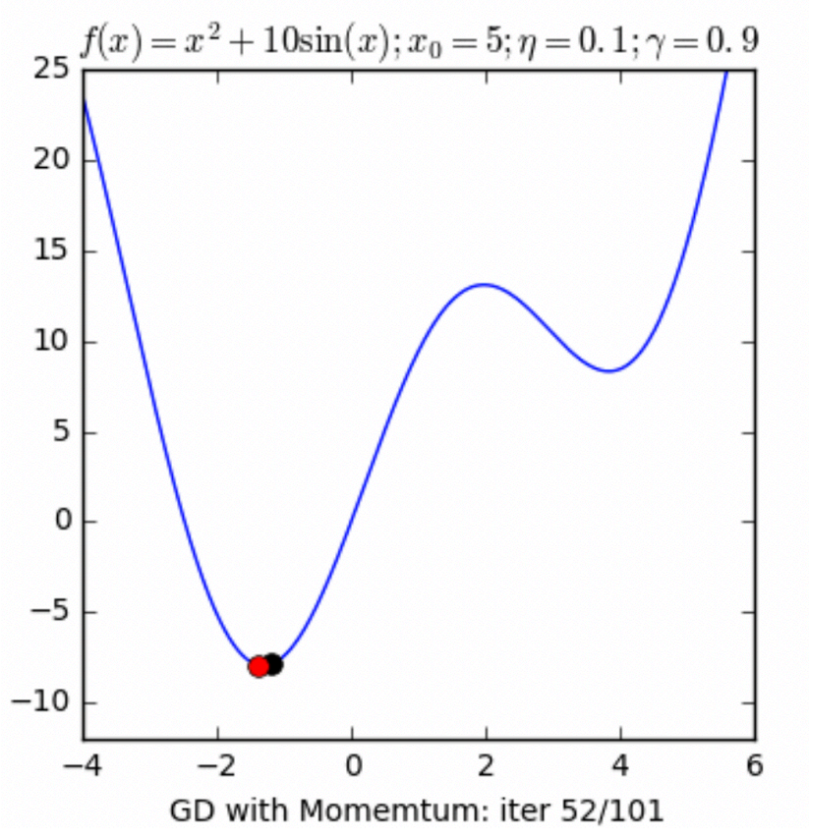
- Our job now is to calculate the quantity v_t so that it both carries information about the slope (ie the derivative) and also carries information about the momentum, which is the previous velocity v_(t-1)

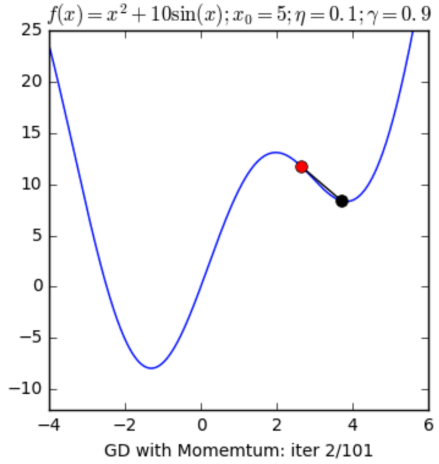
$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$

Trong GD, chúng ta cần tính lượng thay đổi ở thời điểm t để cập nhật vị trí mới cho nghiệm (tức *hòn bi*). Nếu chúng ta coi đại lượng này như vận tốc v_t trong vật lý, vị trí mới của *hòn bi* sẽ là $\theta_{t+1} = \theta_t - v_t$. Dấu trừ thể hiện việc phải di chuyển ngược với đạo hàm. Công việc của chúng ta bây giờ là tính đại lượng v_t sao cho nó vừa mang thông tin của *độ dốc* (tức đạo hàm), vừa mang thông tin của *đà*, tức vận tốc trước đó v_{t-1} (chúng ta coi như vận tốc ban đầu $v_0 = 0$). Một cách đơn giản nhất, ta có thể cộng (có trọng số) hai đại lượng này lại:

Momentum







https://machinelearningcoban.com/2017/01/16/gradientdescent2/

Adagrad (Adaptive Gradients)

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot \frac{\partial J}{\partial \theta_{t-1}}$$

$$G_t = \sum_{i=1}^{t} g_i^2$$

$$g_i = \frac{\partial J}{\partial \theta_i}$$

Trong đó:

θ: Tập các tham số.

η: Learning rate được khởi tạo ban đầu.

G_t: Biểu thị tốc độ học khác nhau cho mỗi tập tham số ở mỗi lần lặp.

ε: Hệ số tránh lỗi(mẫu bằng 0).

RMSProp (root mean square propagation)

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E_t + \epsilon}} \cdot g_t$$

$$E_t = \gamma \cdot E_{t-1} + (1 - \gamma) \cdot g_t^2$$

$$g_t = \frac{\partial J}{\partial \theta_t}$$

Trong đó:

θ: Tập các tham số.

η: Learning rate.

g_t: Gradient tại thời điểm t.

ε: Hệ số tránh lỗi(mẫu bằng 0).

Adam

- Adam is one of the top optimization techniques used today. This method combines the advantages of RMSprop and momentum.

$$\theta_{t+1} = \theta_t + \Delta\theta_t$$

$$\Delta\theta_t = -\frac{\sqrt{E[\Delta\theta^2]_{t-1} + \epsilon}}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t$$

$$E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1-\gamma)\Delta\theta_t^2$$

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)g_t^2$$

$$g_t = \frac{\partial J}{\partial \theta_t}$$

Trong đó:

θ: Tập các tham số.

g_t: Gradient tại thời điểm t.

ε: Hệ số tránh lỗi(mẫu bằng 0).

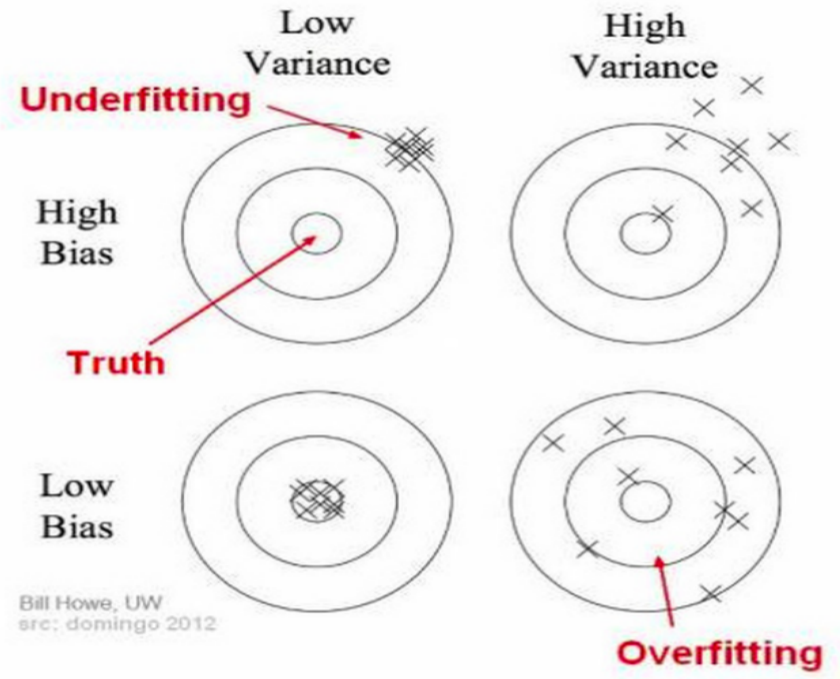
γ: Tốc độ phân rã. Thường được đặt là 0.9

Overfitting

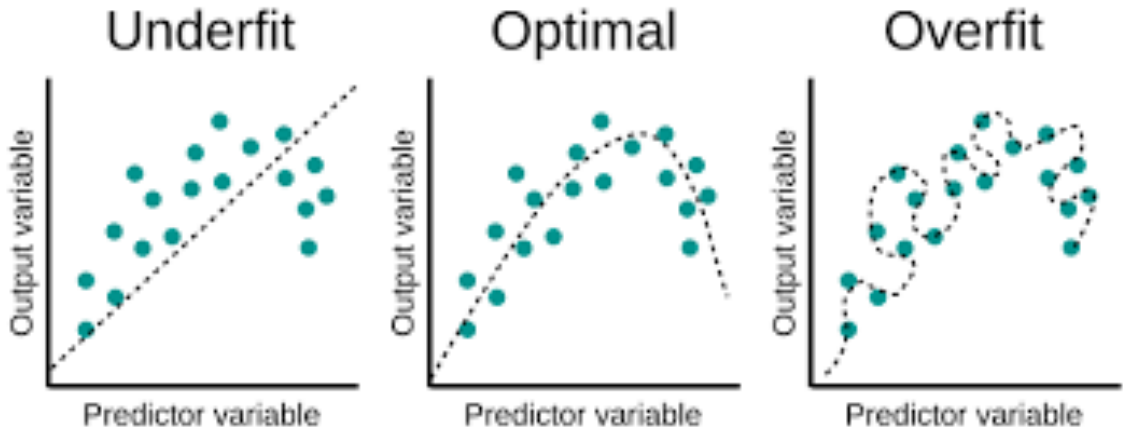
Overfitting: Bias and Variance

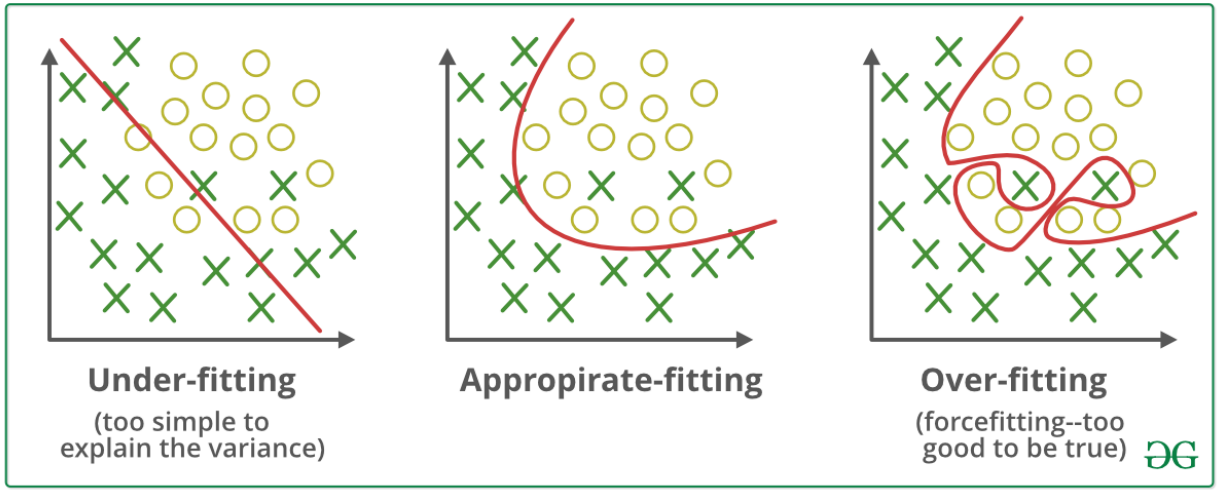
- Bias is the deviation between the value predicted by the model and the actual value (ground truth).

- Variance represents the dispersion of the predicted value. High variance shows high dispersion, the model focuses mainly on training data, the value has large fluctuations without generalizability on new data sets.



Overfitting

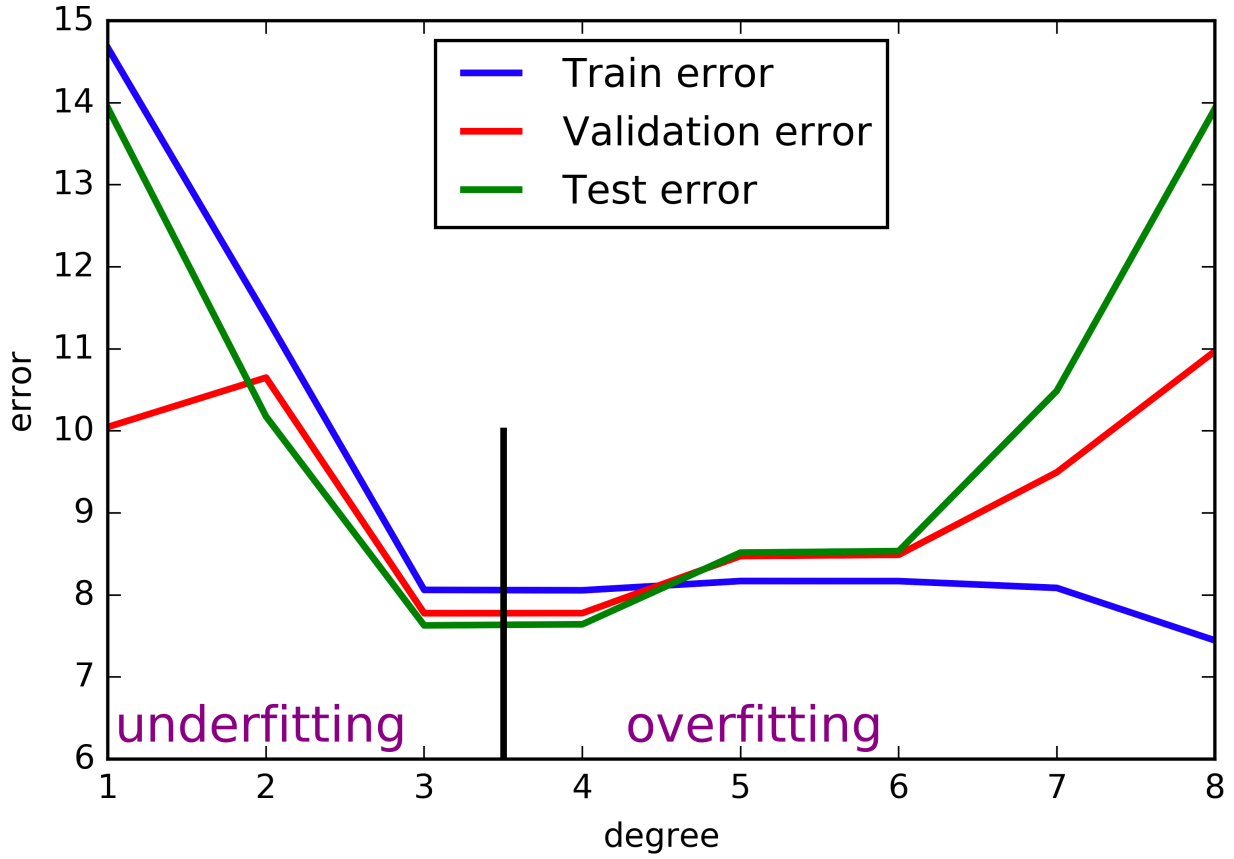




Solutions for preventing overfitting

- Data augmentation
- Feature selection
- Early stopping
- Regularization
- …

Early stopping



Regularization

Kỹ thuật này sử dụng một hàm mất mát mới hay **Regularized Loss Function** được tạo ra bằng cách thêm vào hàm Loss ban đầu một số hạng dùng để đánh giá độ phức tạp của mô hình.

$$Loss_{regularized}(\theta) = Loss(\theta) + \lambda R(\theta)$$

Trong đó:

- Loss(θ): Là hàm Loss ban đầu.

- λ: là hằng số điều chỉnh, được dùng để kiểm soát mức độ của số hạng Regularization.

- R(θ): Là đại lượng điều chỉnh được thêm vào.

a) L2 Regularization (Ridge Regression):

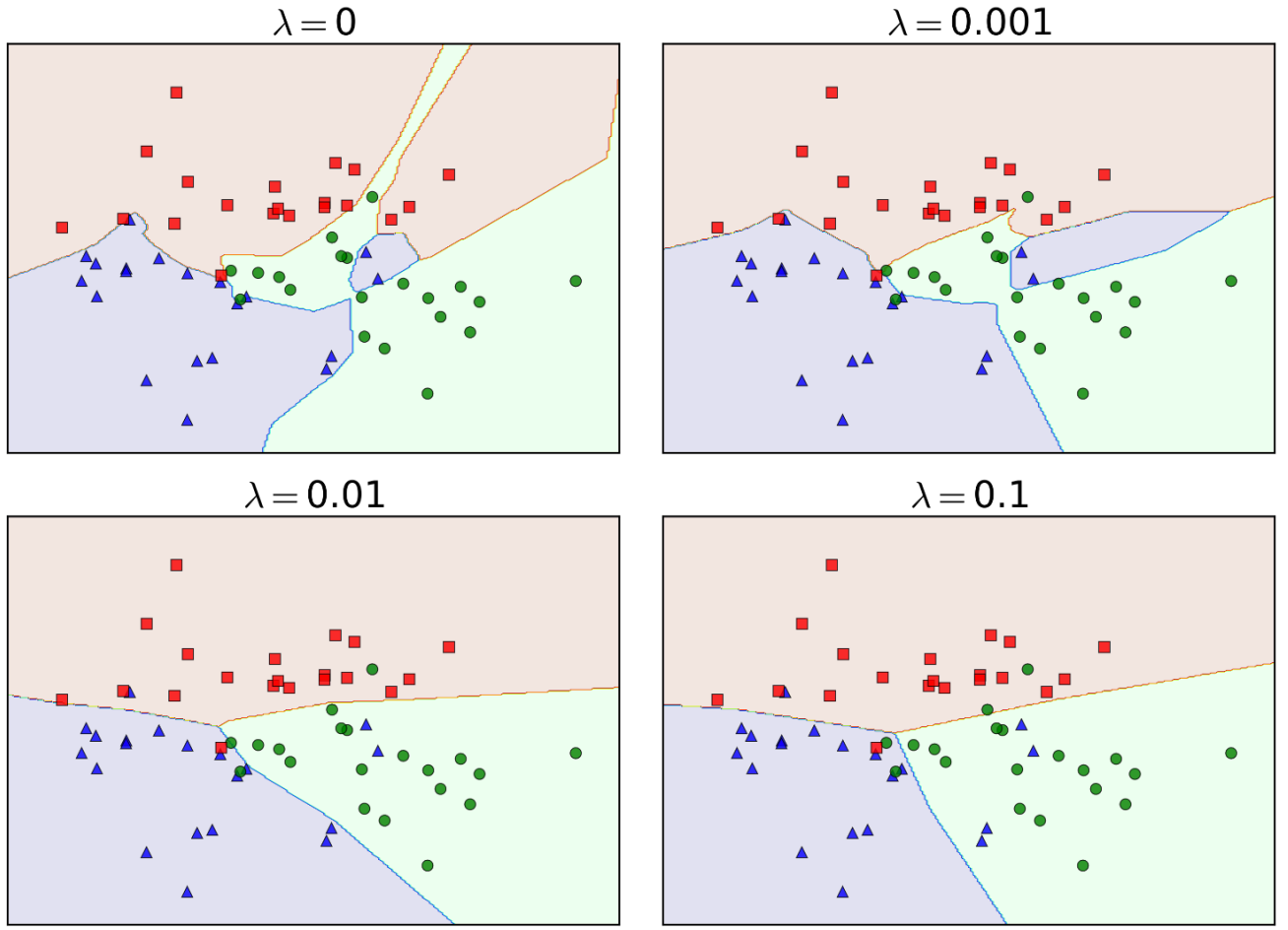
$$R(w) = ||w||_2^2 = \sum_{j=1}^{n} |w_j|^2$$

b) L1 Regularization (Lasso Regression):

$$R(w) = ||w||_1 = \sum_{j=1}^{n} |w_j|$$

Regularization

- Mục tiêu của Regularization là kiểm soát độ phức tạp của mô hình bằng cách giảm trọng số của các thông số không quan trọng hoặc tạo ra sự thưa thớt trong mô hình. Việc này giúp tránh overfitting và tăng khả năng tổng quát hóa của mô hình trên dữ liệu mới.
- Vì vậy trong Neural Network L2 Regression còn được gọi là Weight Decay.



Multi-layer Perceptron với Weight Decay (Source code).

Practice