

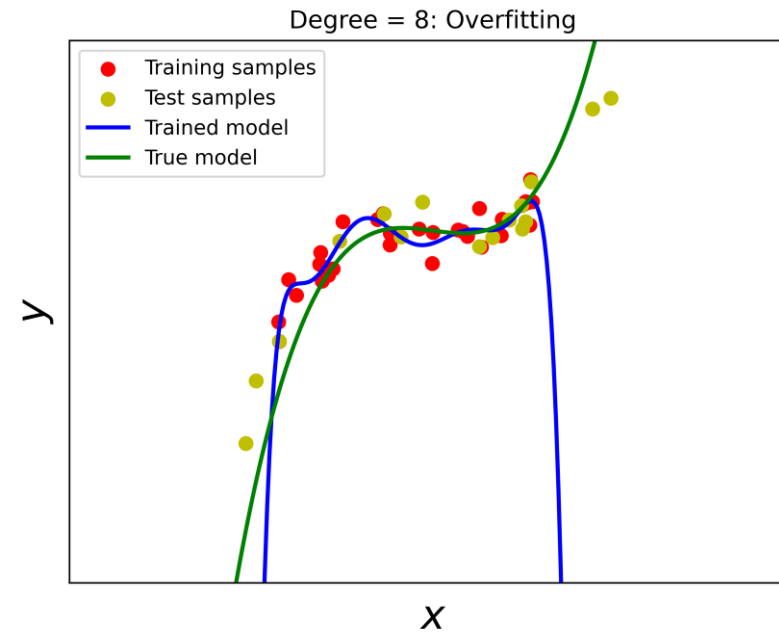
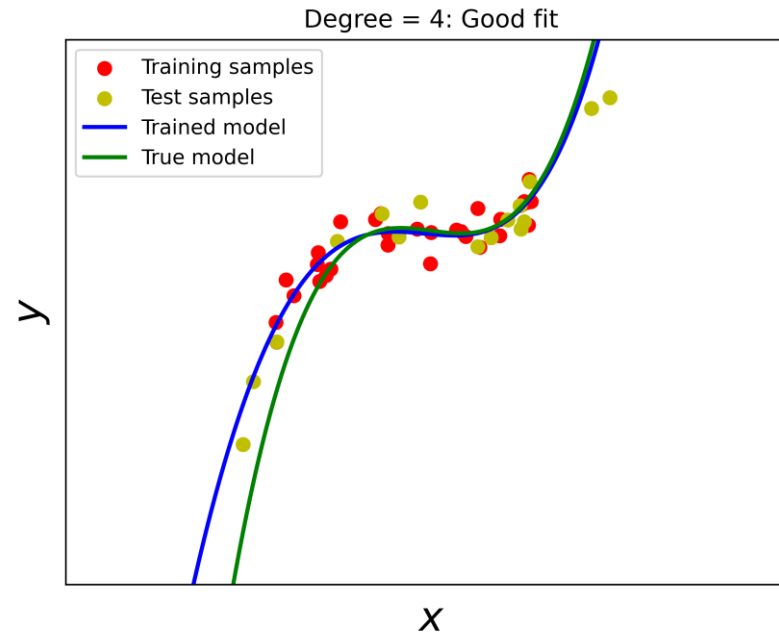
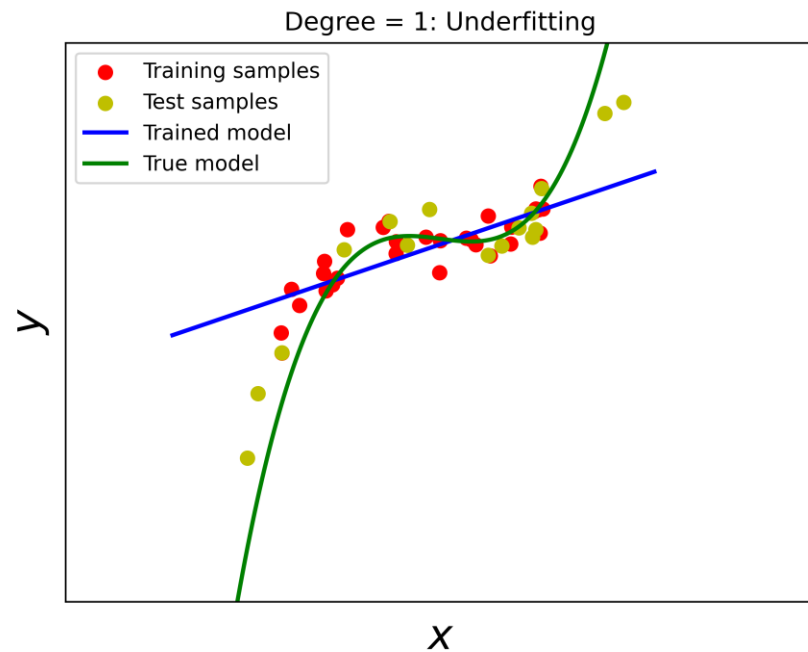
# Overfitting in Machine Learning

LÊ ANH CƯỜNG

# Problem

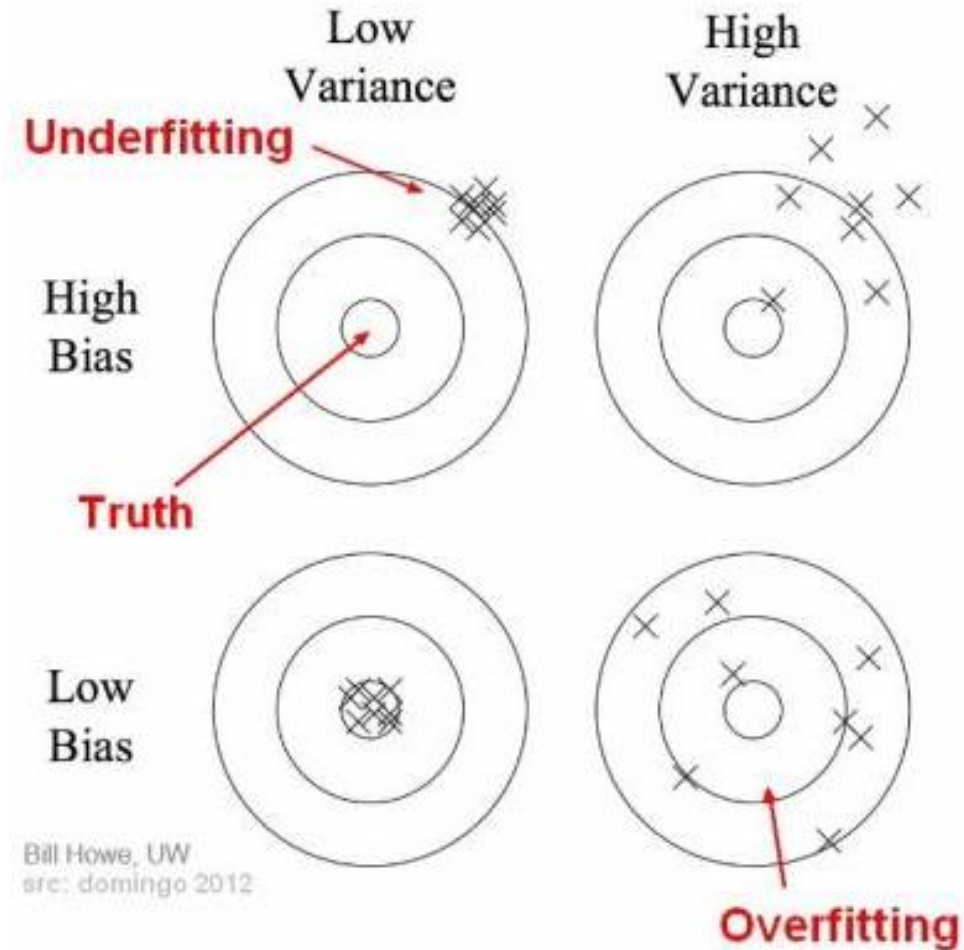
- Công thức nội suy Lagrange:

$$P(x) = \sum_{i=1}^{n+1} y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$$

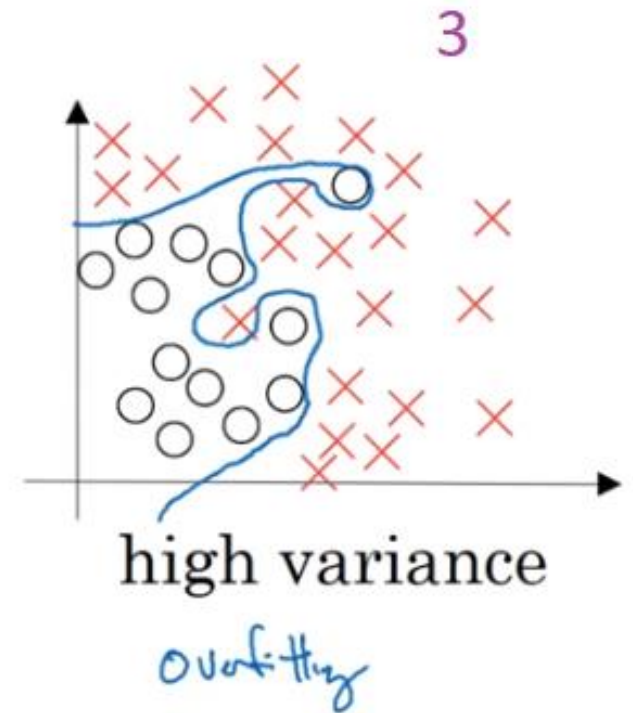
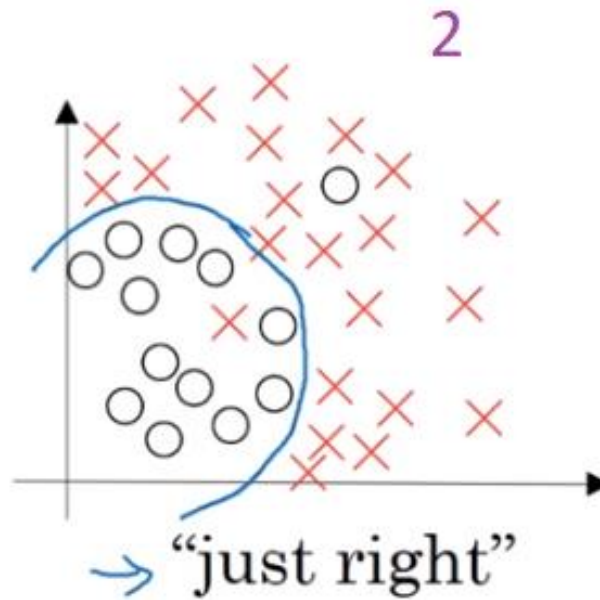
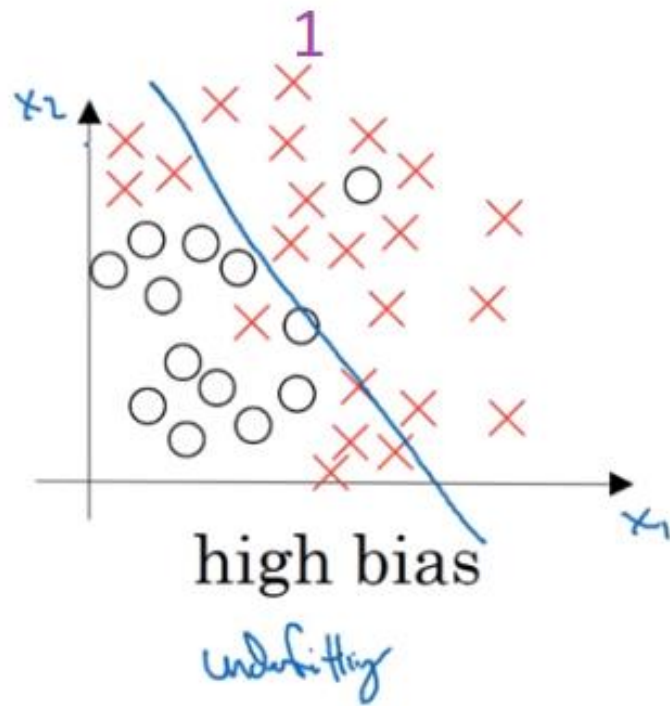


# Bias and Variance

- Bias: nghĩa là độ lệch, biểu thị sự chênh lệch giữa giá trị trung bình mà mô hình dự đoán và giá trị thực tế của dữ liệu.
- Variance: nghĩa là phương sai, biểu thị độ phân tán của các giá trị mà mô hình dự đoán so với giá trị thực tế.
- Ta mong muốn low bias và low variance.



# Bias and Variance



# Solutions

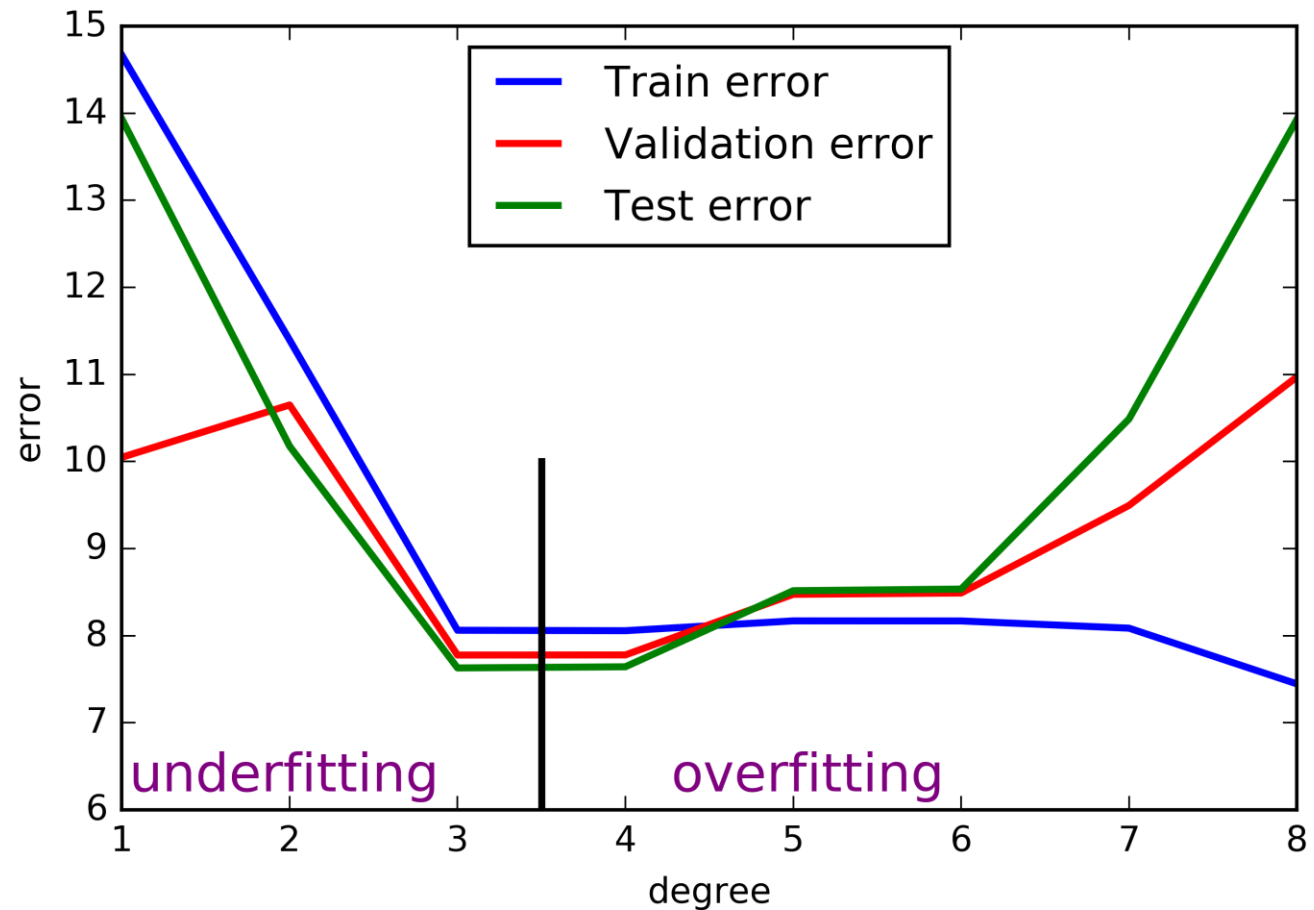
1. Using validation data set
2. Using cross-validation test
3. Regularization
  1. Early stopping
  2. regularized loss function
4. Dropout in Neural Networks

# Data sets

- Training data
- Test data

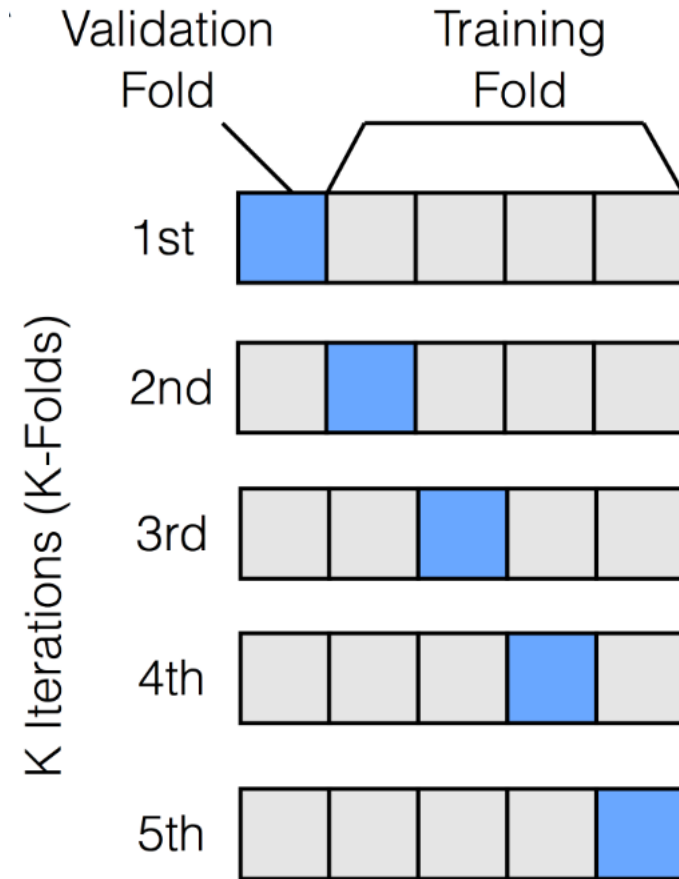
# 1. Validation

- Training dataset  
-> Validation dataset
- Test dataset



## 2. Cross-Validation

- In many cases, we have limited data to build models. If we take too much data from the training set for validation, the remaining part of the training set is not enough to build the model.
- Use k-fold cross validation.

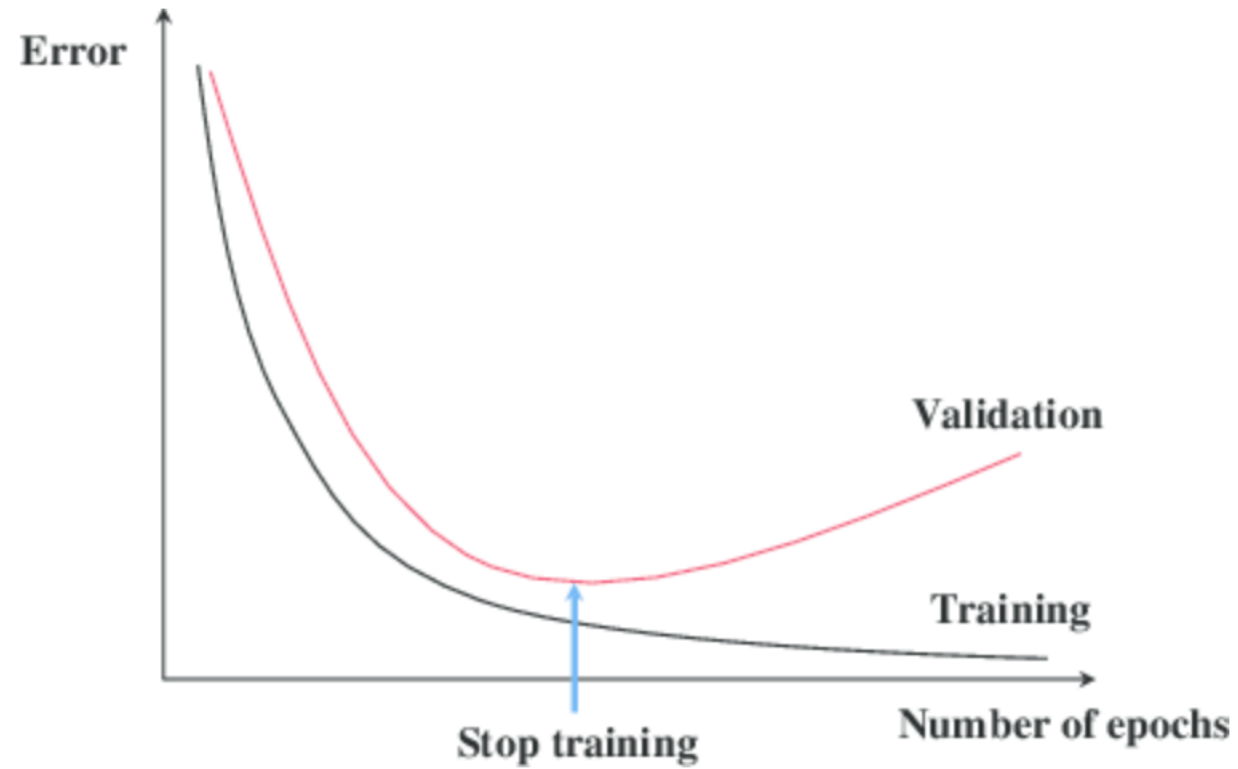


# 3. Regularization

*Regularization*, một cách cơ bản, là thay đổi mô hình một chút để tránh overfitting trong khi vẫn giữ được tính tổng quát của nó (tính tổng quát là tính mô tả được nhiều dữ liệu, trong cả tập training và test). Một cách cụ thể hơn, ta sẽ tìm cách *di chuyển* nghiệm của bài toán tối ưu hàm mất mát tới một điểm gần nó. Hướng di chuyển sẽ là hướng làm cho mô hình *ít phức tạp hơn* mặc dù giá trị của hàm mất mát có tăng lên một chút.

<https://machinelearningcoban.com/2017/03/04/overfitting/>

## 3.1. Early Stopping



## 3.2 Regularized Loss Function

Kỹ thuật regularization phổ biến nhất là thêm vào hàm mất mát một số hạng nữa. Số hạng này thường dùng để đánh giá độ phức tạp của mô hình. Số hạng này càng lớn, thì mô hình càng phức tạp. *Hàm mất mát mới* này thường được gọi là **regularized loss function**, thường được định nghĩa như sau:

$$J_{\text{reg}}(\theta) = J(\theta) + \lambda R(\theta)$$

$$l_2 \text{ regularization} \quad R(\mathbf{w}) = \|\mathbf{w}\|_2^2$$

# $l_2$ regularization

Hàm số này có một vài đặc điểm đang lưu ý:

- Thứ nhất,  $\|\mathbf{w}\|_2^2$  là một hàm số *rất mượt*, tức có đạo hàm tại mọi điểm, đạo hàm của nó đơn giản là  $\mathbf{w}$ , vì vậy đạo hàm của *regularized loss function* cũng rất dễ tính, chúng ta có thể hoàn toàn dùng các phương pháp dựa trên gradient để cập nhật nghiệm. Cụ thể:

$$\frac{\partial J_{\text{reg}}}{\partial \mathbf{w}} = \frac{\partial J}{\partial \mathbf{w}} + \lambda \mathbf{w}$$

- Thứ hai, việc tối thiểu  $\|\mathbf{w}\|_2^2$  đồng nghĩa với việc khiến cho các giá trị của hệ số  $\mathbf{w}$  trở nên nhỏ gần với 0. Với Polynomial Regression, việc các hệ số này nhỏ có thể giúp các hệ số ứng với các số hạng bậc cao là nhỏ, giúp tránh overfitting. Với Multi-layer Perceptron, việc các hệ số này nhỏ giúp cho nhiều hệ số trong các ma trận trọng số là nhỏ. Điều này tương ứng với việc số lượng các hidden units *hoạt động* (khác không) là nhỏ, cũng giúp cho MLP tránh được hiện tượng overfitting.

$l_2$  regularization là kỹ thuật được sử dụng nhiều nhất để giúp Neural Networks tránh được overfitting. Nó còn có tên gọi khác là **weight decay**. *Decay* có nghĩa là *tiêu biến*.

# $l_2$ regularization

$l_2$  regularization là kỹ thuật được sử dụng nhiều nhất để giúp Neural Networks tránh được overfitting. Nó còn có tên gọi khác là **weight decay**. *Decay* có nghĩa là *tiêu biến*.

Trong Xác suất thống kê, Linear Regression với  $l_2$  regularization được gọi là **Ridge Regression**. Hàm mất mát của *Ridge Regression* có dạng:

$$J(\mathbf{w}) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$

trong đó, số hạng đầu tiên ở vế phải chính là hàm mất mát của Linear Regression. Số hạng thứ hai chính là phần regularization.

# $l_1$ regularization and $l_2$ regularization

L1 Regularization

$$\text{Cost} = \sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2 + \lambda \sum_{j=0}^M |W_j|$$

L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2}_{\text{Loss function}} + \lambda \underbrace{\sum_{j=0}^M W_j^2}_{\text{Regularization Term}}$$

# $l_1$ regularization

<https://machinelearningcoban.com/2017/03/04/overfitting/>

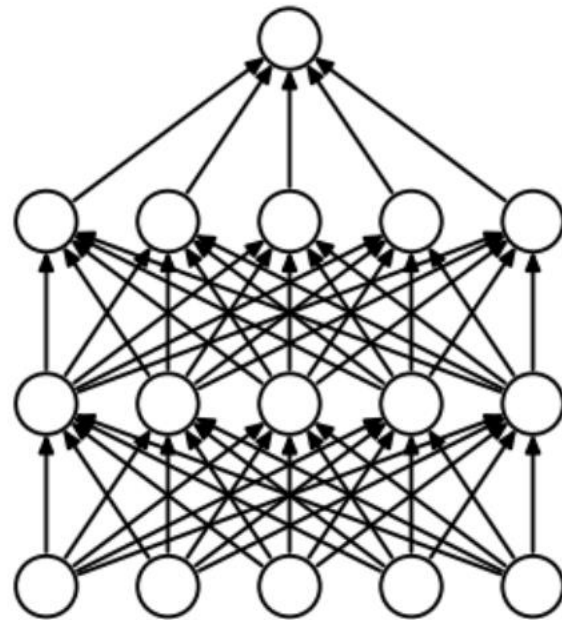
Trong Thống Kê, việc sử dụng  $l_1$  regularization còn được gọi là **LASSO** (Least Absolute Shrinkage and Selection Operator).

Khi cả  $l_2$  và  $l_1$  regularization được sử dụng, ta có mô hình gọi là **Elastic Net Regression**.

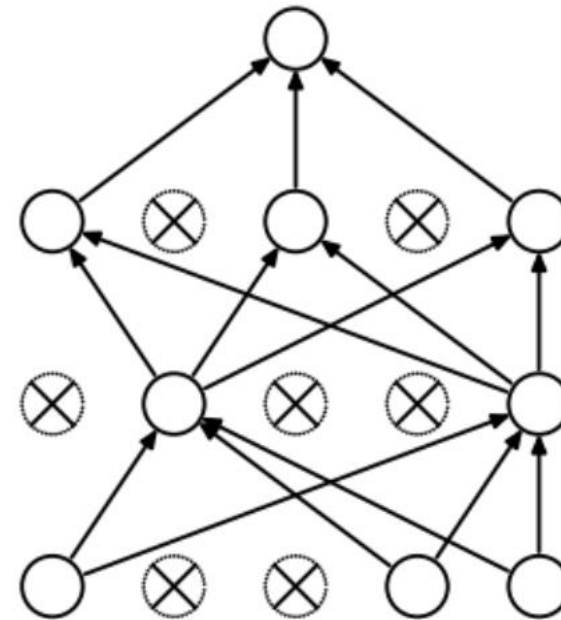
## 3.6. Regularization trong sklearn

Trong **sklearn**, ví dụ **Logistic Regression**, bạn cũng có thể sử dụng các  $l_1$  và  $l_2$  regularizations bằng cách khai báo biến **penalty='l1'** hoặc **penalty = 'l2'** và biến **C**, trong đó **C** là *nghịch đảo* của  $\lambda$ . Trong các bài trước khi chưa nói về Overfitting và Regularization, tôi có sử dụng **C = 1e5** để chỉ ra rằng  $\lambda$  là một số rất nhỏ.

# Dropout in Neural Networks



(a) Standard Neural Net



(b) After applying dropout.

