Summary

Data

- Scope of research problems
- Data source:
  - random sample
  - experiments
  - big data

Python tools

- pandas
- matplotlib and seaborn
- scikit-learn
- Tensorflow 2.x

Machine learning

Statistical model
Data $\longrightarrow$ optimization $\longrightarrow$ Predictions

目標：訓練出預測最準的模型

Types of machine learning

- Supervised learning
- Unsupervised learning
- Semisupervised learning
- Reinforcement learning
- etc.

How do machines learn?

- Find an appropriate decision function (machine learning model)

  $\hat{y} = h(\mathbf{x})$: feature space → decision space

  to predict future outcomes

- Hope that $h(\mathbf{x})$ can make **accurate** predictions

Loss function

A loss function $L\left(y, \hat{y}\right)$ **measures the accuracy** of $\hat{y} = h(\mathbf{x})$, in terms of the cost we will pay for a bad decision:

Regression:

- squared–error loss
- huber loss

Classification:

- cross–entropy loss
- categorical cross–entropy loss

Risk function

After the machine learning algorithm is deployed, how much loss we will pay for a future event?

random (unknown)

- Estimate it by the **expected loss (risk function)**

$$
\begin{aligned}
R(h) = E\left[L\left(Y, h(\mathbf{X})\right)\right] &= \int L\left(y, h(\mathbf{x})\right) f\left(\mathbf{x}, y\right) d\mathbf{x}dy \\
&= \int L\left(y, h(\mathbf{x})\right) dF\left(\mathbf{x}, y\right)
\end{aligned}
\qquad (1)
$$

Lebesgue integral

where $F\left(\mathbf{x}, y\right)$ is the joint c.d.f. of $\mathbf{X}$ and Y.

Optimal decision

- Find the best decision rule that minimizes the risk function, i.e.

$$h^*(\mathbf{x}) = \arg\min_{h \in \mathscr{H}} R\left(h\right) \tag{2}$$

- Machine learning = learn optimal decisions

Empirical risk

- When the data $\left\{\mathbf{x}_i, y_i\right\}_{i=1}^{n}$ are drawn independently from $F\left(\mathbf{x}, y\right)$, equation (2) might be approximated by its empirical version

$$\hat{h} = \arg\min_{h \in \mathscr{H}} \boxed{\frac{1}{n} \sum_{i=1}^{n} \left[L\left(y_i, h(\mathbf{x}_i)\right) \right]} \tag{3}$$

empirical risk

approximate $\boldsymbol{R}$ by LLN (conditions?)

Machine learning

Specify $L\left(y, \hat{y}\right)$
and $\mathscr{H}$
Data $\left\{\mathbf{x}_i, y_i\right\}_{i=1}^{n}$

$\longrightarrow$

Solve
equation (3)

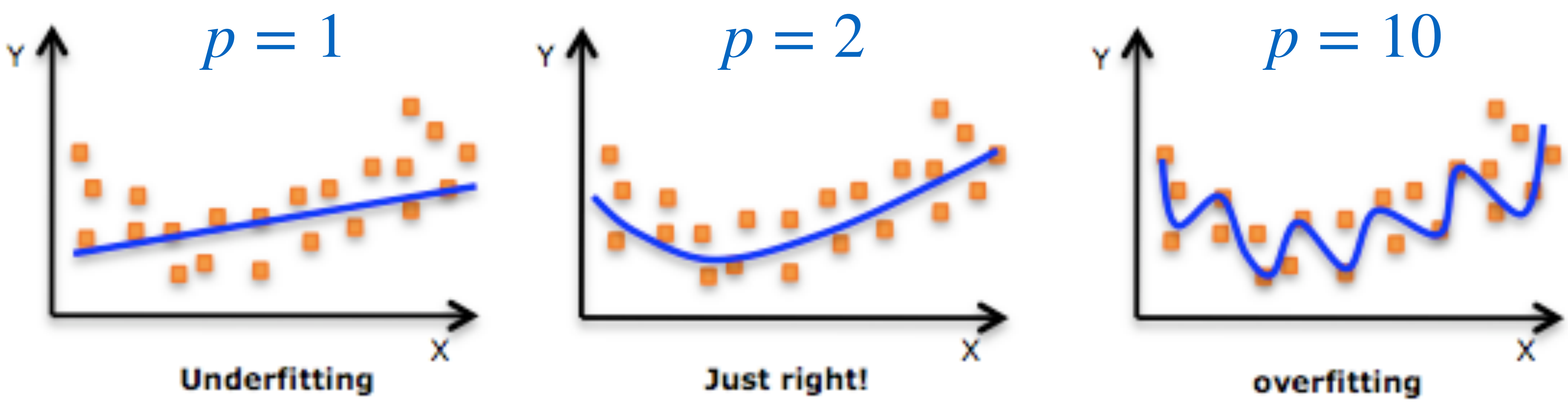
$\longrightarrow$

$\hat{h}(\mathbf{x})$ for
predictions

Decision functions

- Linear function
- **Artificial neural networks**

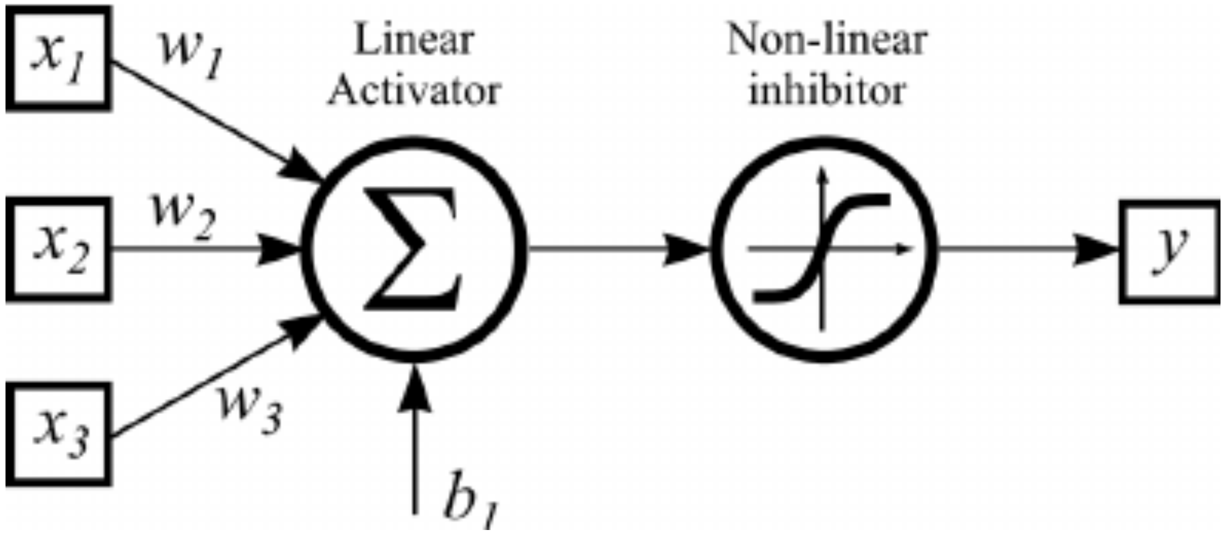
Bias–variance tradeoff

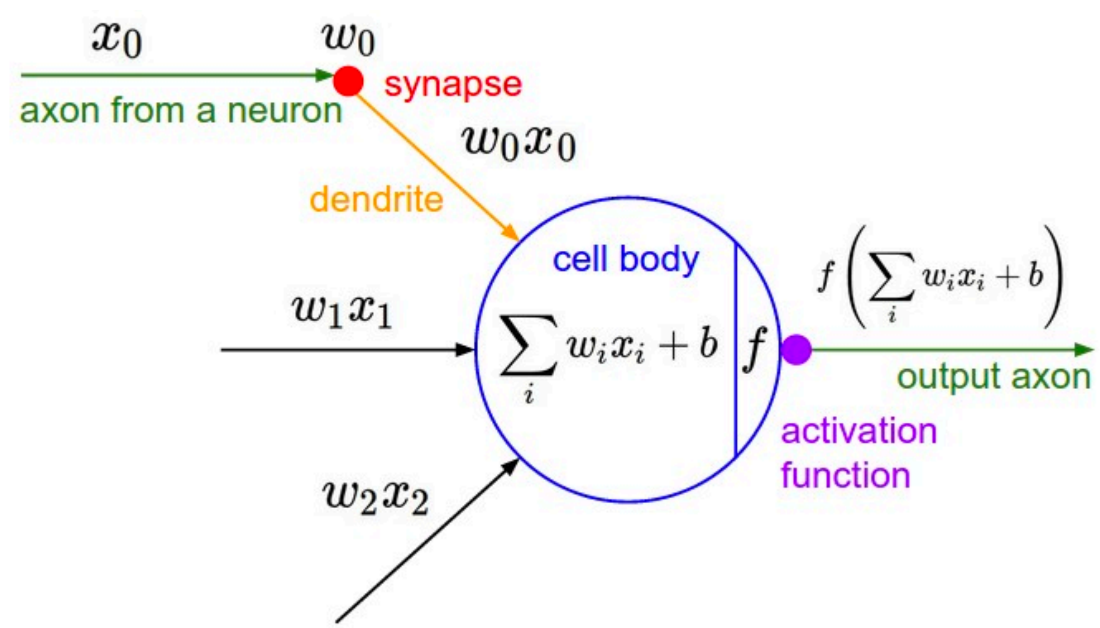


large bias

large variance

Artificial Neurons



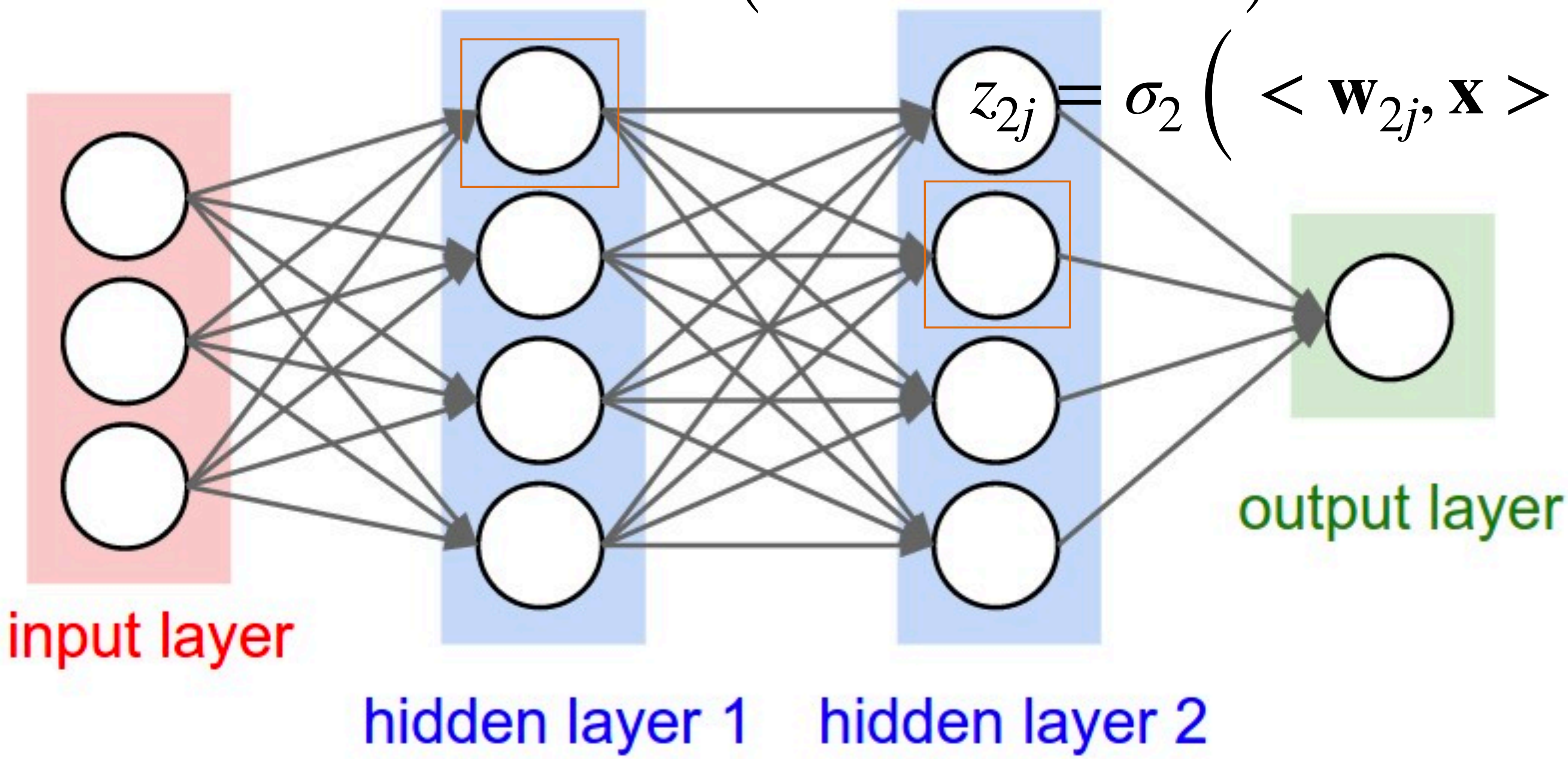


- Computational Methods and Optimization
- https://cs231n.github.io/convolutional-networks/

Artificial neural networks

$$z_{1j} = \sigma_1\left(< \mathbf{w}_{1j}, \mathbf{x} > + b_{1j}\right)$$

$$z_{2j} = \sigma_2\left(< \mathbf{w}_{2j}, \mathbf{x} > + b_{2j}\right)$$



$$
\begin{aligned}
\mathbf{z}_1 &= \sigma_1\left(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1\right)\\
\mathbf{z}_2 &= \sigma_2\left(\mathbf{W}_2\mathbf{z}_1 + \mathbf{b}_2\right)\\
&\vdots\\
\mathbf{z}_\ell &= \sigma_\ell\left(\mathbf{W}_\ell\mathbf{z}_{\ell-1} + \mathbf{b}_\ell\right)\\
f(\mathbf{x}) &= \sigma_f\Big(\mathbf{W}_f\mathbf{z}_\ell + \mathbf{b}_f\Big)
\end{aligned}
$$

The empirical risk of a feed-forward networks becomes

$$
R = \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, \hat{y}_i\right)
$$

where $\hat{y} = h\left(f(\mathbf{x})\right)$.

Predictions are made by the outputs of an artificial neural network:

- regression: $h(t) = t$
- classification: $h(t) =$ sigmoid or softmax functions

Automatic differentiation by <u>backpropagation</u>

Obtain ∇R automatically by chain rules:

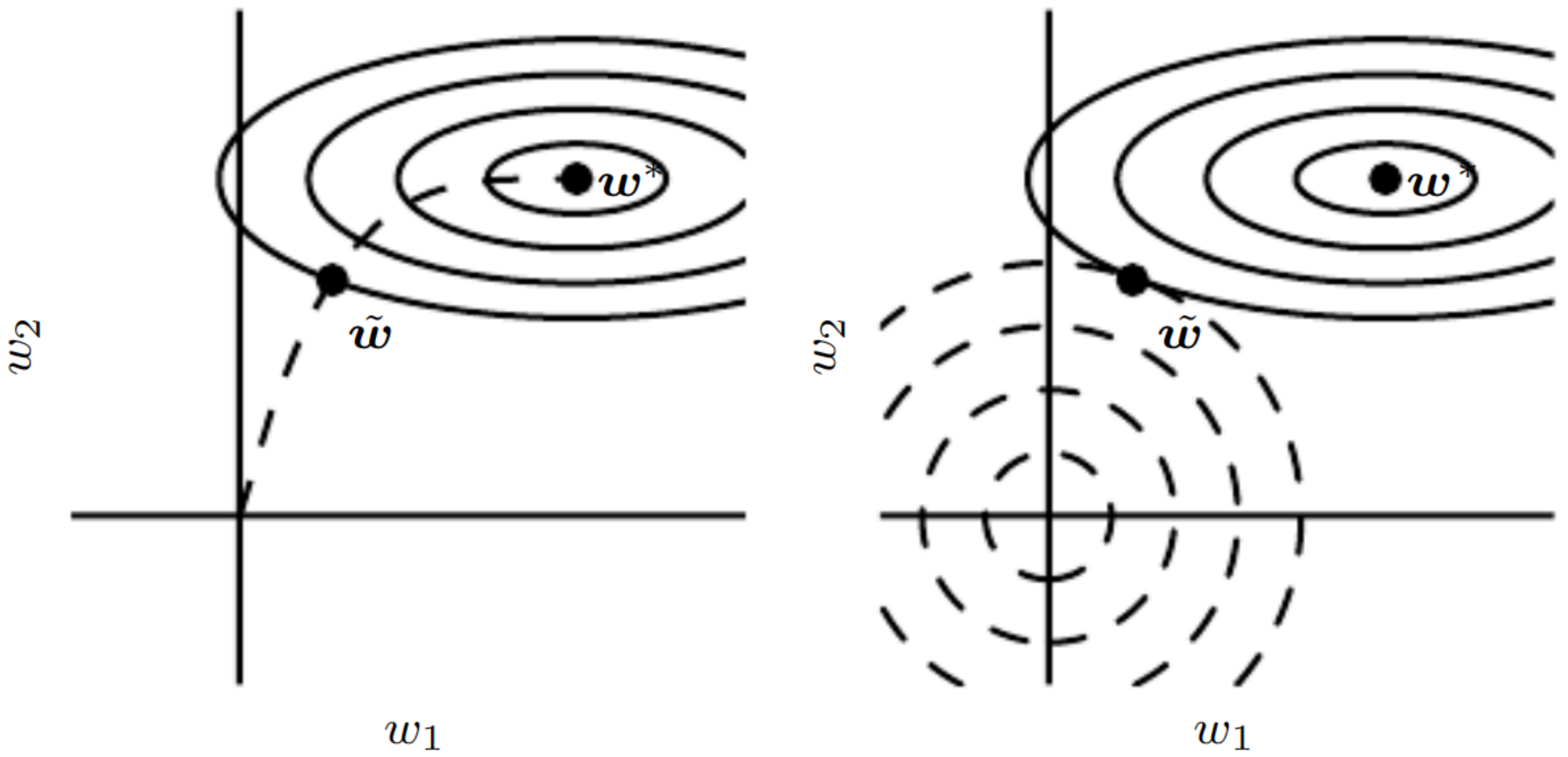
$$\frac{\partial R}{\partial \mathbf{W}_{\ell+1}} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial L\left(y_i, f(\mathbf{x}_i)\right)}{\partial f}\frac{\partial f}{\partial \mathbf{W}_{\ell+1}},$$

$$\frac{\partial R}{\partial \mathbf{W}_{\ell}} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial L\left(y_i, f(\mathbf{x}_i)\right)}{\partial f}\frac{\partial f}{\partial \mathbf{z}_{\ell}}\frac{\partial \mathbf{z}_{\ell}}{\mathbf{W}_{\ell}},$$

$$\frac{\partial R}{\partial \mathbf{W}_{\ell-1}} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial L\left(y_i, f(\mathbf{x}_i)\right)}{\partial f}\frac{\partial f}{\partial \mathbf{z}_{l}}\frac{\partial \mathbf{z}_{\ell}}{\partial \mathbf{z}_{\ell-1}}\frac{\partial \mathbf{z}_{\ell-1}}{\partial \mathbf{W}_{\ell-1}},$$

$$\vdots$$

Early stopping



Early stopping restricts the gradient descent algorithm to a relatively small volume of parameter space in the neighborhood of the initial parameter $\boldsymbol{\theta}_0$

Why does deep learning so successful?

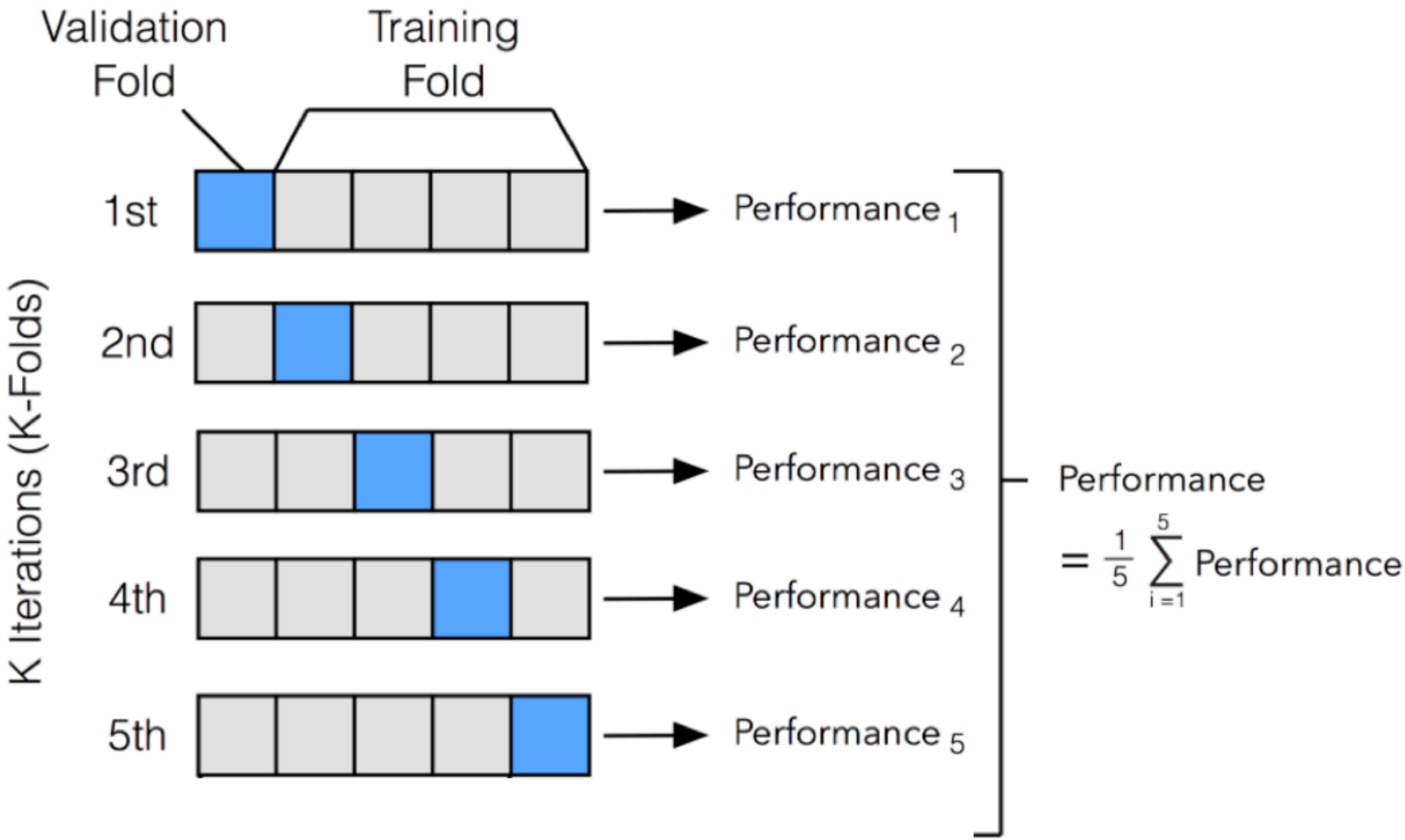
- Universal approximation
- ReLU networks are universal approximations via piecewise linear or constant functions
- Overparameterization in deep learning does not lead to overfitting
- Gradient descent finds global minima of deep neural networks

Cross validation

- **Estimate the prediction error** by monte carlo simulations

- Repeat the random split for multiple times and estimate the future prediction error by the mean cross-validated prediction error

- Reference



http://ethen8181.github.io/machine-learning/model_selection/model_selection.html

Metrics for evaluating prediction errors

Regression:

- mean squared error (MSE)
- mean absolute error (MAE)
- R^2

Classification:

- confusion matrix
- accuracy
- precision
- recall
- AUC
- micro/macro averages

期末考

- 證明題：40分
- 程式題：100分 (regression、classification各50分)

集滿100分即可