Bias and Variance, Under-Fitting and Over-Fitting

Shan He

School for Computational Science
University of Birmingham

Module 06-27818 and 27819:
Introduction to Neural Computation (Level 4/M)
Neural Computation (Level 3)

Outline of Topics

Computational/Learning Power of MLPs

Universal Approximation Theorem: a MLP, i.e., a feed-forward network with a single hidden layer containing a finite number of neurons, can approximate continuous functions on compact subsets of $\mathbb{R}^n$, under mild assumptions on the activation function.

Computational/Learning Power of MLPs

- **Universal Approximation Theorem**: Let φ be a non-constant, bounded, and monotone-increasing continuous function. Then for **any continuous function** $f(\mathbf{x})$ with $\mathbf{x} = \{x_i \in [0,1]\}$, where $i = 1, \cdots m$ and $\xi > 0$, there exists an integer M and real constants $\{a_j, b_j, w_{ij}\}$, where $j = 1, \cdots, M$ and $k = 1, \cdots m$ such that

$$F(x_1, \cdots, x_m) = \sum_{j=1}^{M} a_j \varphi\left(\sum_{k=1}^{m} w_{jk} x_k - b_j\right) = \sum_{j=1}^{M} a_j \varphi\left(\sum_{k=0}^{m} w_{jk} x_k\right)$$

is an approximate realisation of $f(\cdot)$, that is

$$|F(x_1, \cdots, x_m) - f(x_1, \cdots, x_m)| < \xi$$

for all $\mathbf{x}$ that lie in the input space.

Computational/Learning Power of MLPs: implications

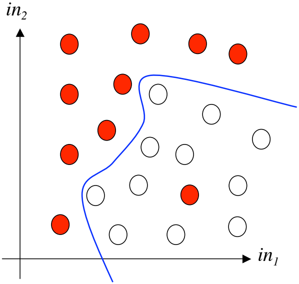
- $F(x_1, \cdots, x_m)$: a 3-layer MLP with M hidden neurons:

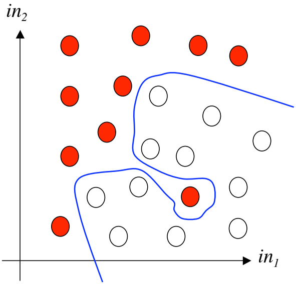
$$F(x_1, \cdots, x_m) = \sum_{j=1}^{M} a_j \varphi \left(\sum_{k=0}^{m} w_{jk} x_k \right)$$

  - φ: hidden layer activation function
  - w_{ij}: hidden layer weights
  - a_j: output layer weights
- Implication 1: given enough hidden units, a two layer MLP can approximate any continuous function
- Implication 2: ξ can be as small as possible.
- **Question**: do we need the minimum value of ξ? Or in other word, to obtain a perfect decision boundary to classify training data with the minimum error?

Learning and Generalization Revisited

- **Aim of learning algorithms**: to generalize to classify/regress new inputs appropriately
- **Reality**: data is known to contain noise
- **Implication**: we dont necessarily want the training data to be classified totally accurately, because that is likely to reduce the generalization ability





Let's play with the toy problem using our MLP

- The problem: a binary non-linearly separable classification problem
- Using make_moon function to generate 2d binary classification problems where data points are two interleaving half circles
- We generate 20 training samples and 20 testing samples with some noise.
- Let's try a few MLP parameters:
  - Number of hidden neurons: 2, 200
  - Number of maximum iterations: 100, 1000

Generalization in General

- Empirically determined data points will usually contain a certain level of **noise**, e.g. incorrect class labels or measured values that are inaccurate.
- In most cases, the underlying "correct decision boundary or function will be **smoother** than that indicated by a given set of noisy training data.
- If we had an infinite number of data points, the errors and inaccuracies would be easier to spot and tend to cancel out of averages.
- Different sets of training data, i.e. different sub-sets of the infinite set of all possible training data, will lead to different network weights and outputs.

Generalization in General

Key question: how to recover the best smooth underlying function or decision boundary from a given set of noisy training data?

A Statistical View of the Training Data

A rigorous **statistical** approach to understand generalization from a theoretical point of view:

- Analogy: frequentist approach to probability, i.e., takes a finite set of measurements from an infinite set of possible measurements to estimate a probability,
- We take a finite set of data points to train our neural networks.
- Suppose we have a training data set D for our neural network $D = \{\mathbf{x}^p, t^p\}$, $p = 1, \cdots, P$, $\mathbf{x}^p = \{x_1^p, x_2^p, \cdots, x_M^p\}^{\mathrm{T}}$
  - Note: We assume that we only have one output unit the extension to many outputs is obvious.
- **Aim**: to find a way to understand statistically the generalization obtained using such a data set, and then optimise the generalization process

A Regressive Model of the Data

- We assume: the training data is generated by some **actual function** $f(x)$ plus random noise ϵ.
- We assume: noise ϵ is normally distributed with a mean of zero, i.e., $\epsilon \in \mathcal{N}(0, \sigma_\epsilon)$.
- Therefore, the real values t in the training set is:

$$t = f(x) + \epsilon$$

- We want to find a **model** $\hat{f}(x)$ to approximate the **actual function** $f(x)$ as well as possible
- We define the expected squared prediction error at a point x:

$$\mathrm{Err}(x) = \mathrm{E}\left[(t - \hat{f}(x))^2\right]$$

where E is the statistical expectation operator that averages over all possible training patterns

Just give me the equation

- The error is $\mathrm{Err}(x)$ is

$$\mathrm{Err}(x) = \mathrm{E}\left[\left(t - \hat{f}(x)\right)^2\right] = \mathrm{Bias}\left[\hat{f}(x)\right]^2 + \mathrm{Var}\left[\hat{f}(x)\right] + \sigma^2$$

Where:

$$\mathrm{Bias}\left[\hat{f}(x)\right] = \mathrm{E}\left[\hat{f}(x) - f(x)\right]$$

and

$$\mathrm{Var}\left[\hat{f}(x)\right] = \mathrm{E}[\hat{f}(x)^2] - \mathrm{E}[\hat{f}(x)]^2$$

is the variance and $\sigma = \mathrm{Var}(\epsilon)$ is the irreducible error

Explanation: Error due to bias

- Error due to bias:

$$\mathrm{Bias}\big[\hat{f}(x)\big] = \mathrm{E}\big[\hat{f}(x) - f(x)\big]$$

  means the difference between the expected (or average) prediction of our model and the correct value which we are trying to predict.
- **Intuition**: the error caused by the simplifying assumptions built into the method. E.g., when approximating a non-linear function $f(x)$ using a learning method for linear models, there will be error in the estimates $\hat{f}(x)$ due to this assumption.

Explanation: Error due to variation

- Error due to variation:

$$\mathrm{Var}\left[\hat{f}(x)\right] = \mathrm{E}[\hat{f}(x)^2] - \mathrm{E}[\hat{f}(x)]^2$$

  means the variability of a model prediction for a given data point.
- **Intuition**: represents how much the learning method $\hat{f}(x)$ will move around its mean

Detailed derivation

- Let's abbreviate $f = f(x)$, $\hat{f} = \hat{f}(x)$
- Note: for any random variable X, we have

$$\mathrm{Var}[X] = \mathrm{E}[X^2] - \mathrm{E}[X]^2$$

which can be rearranged as

$$\mathrm{E}[X^2] = \mathrm{Var}[X] + \mathrm{E}[X]^2$$

Detailed derivation

- For any random variable X its variance is:

$$\begin{aligned}\mathrm{Var}(X) &= \mathrm{E}\left[X^2\right] - (\mathrm{E}[X])^2 \\ &= \mathrm{E}\left[(X - \mathrm{E}[X])^2\right]\end{aligned}$$

- Proof:

$$\begin{aligned}\mathrm{Var}(X) &= \mathrm{E}\left[(X - \mathrm{E}[X])^2\right] \\ &= \mathrm{E}\left[X^2 - 2X\,\mathsf{E}[X] + (\mathrm{E}[X])^2\right] \\ &= \mathrm{E}\left[X^2\right] - 2\mathrm{E}[X]\mathrm{E}[X] + (\mathrm{E}[X])^2 \\ &= \mathrm{E}\left[X^2\right] - (\mathrm{E}[X])^2\end{aligned}$$

Detailed derivation

- Since f is deterministic

$$\mathrm{E}[f] = f,$$

so given $t = f + \epsilon$ and $\mathrm{E}[\epsilon] = 0$, implies $\mathrm{E}[t] = \mathrm{E}[f + \epsilon] = f$

- Since $\mathrm{Var}[\epsilon] = \sigma^2$

$$\begin{aligned}\mathrm{Var}[t] = \mathrm{E}[(t - \mathrm{E}[t])^2] = \mathrm{E}[(t - f)^2] = \mathrm{E}[(f + \epsilon - f)^2] = \mathrm{E}[\epsilon^2] \\ = \mathrm{Var}[\epsilon] + \mathrm{E}[\epsilon]^2 = \sigma^2 + 0 = \sigma^2\end{aligned}$$

Detailed derivation

$$\mathrm{E}\big[(t-\hat{f})^2\big] = \mathrm{E}[t^2 - 2t\hat{f} + \hat{f}^2] = \mathrm{E}[t^2] + \mathrm{E}[\hat{f}^2] - \mathrm{E}[2t\hat{f}]$$

Since $\mathrm{E}[X^2] = \mathrm{Var}[X] + \mathrm{E}[X]^2$

$$\mathrm{E}\big[(t-\hat{f})^2\big] = \mathrm{Var}[t] + \mathrm{E}[t]^2 + \mathrm{Var}[\hat{f}] + \mathrm{E}[\hat{f}]^2 - \mathrm{E}[2t\hat{f}]$$

Since $\mathrm{E}[t] = \mathrm{E}[f+\epsilon] = f$ and $\mathrm{E}[2t\hat{f}] = 2\mathrm{E}[t]\mathrm{E}[\hat{f}] = 2f\mathrm{E}[\hat{f}]$

$$\begin{aligned}\mathrm{E}\big[(t-\hat{f})^2\big] &= \mathrm{Var}[t] + \mathrm{Var}[\hat{f}] + f^2 - 2f\mathrm{E}[\hat{f}] + \mathrm{E}[\hat{f}]^2 \\ &= \mathrm{Var}[t] + \mathrm{Var}[\hat{f}] + (f - \mathrm{E}[\hat{f}])^2\end{aligned}$$

Since $\mathrm{E}[f] = f$ and $\mathrm{Var}[t] = \sigma^2$

$$\begin{aligned}\mathrm{E}\big[(t-\hat{f})^2\big] &= \mathrm{Var}[t] + \mathrm{Var}[\hat{f}] + (\mathrm{E}(f) - \mathrm{E}[\hat{f}])^2 \\ &= \mathrm{Var}[t] + \mathrm{Var}[\hat{f}] + \mathrm{E}[f - \hat{f}]^2 \\ &= \sigma^2 + \mathrm{Var}[\hat{f}] + \mathrm{Bias}[\hat{f}]^2\end{aligned}$$

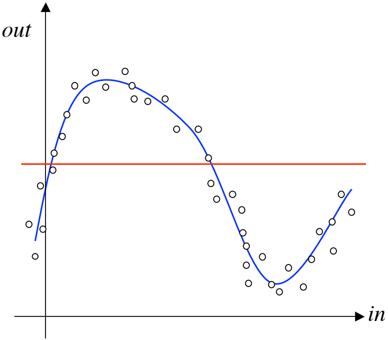
Extreme Case of Bias and Variance: Under-fitting

- Suppose the neural network is lazy and just produces the same **constant output** whatever training data we give it, i.e. $\hat{f}(x, \mathbf{W}, D) = c$, where x is an unseen sample, $\mathbf{W} = w_{ij}^{(n)}$ is a set of network weights, trained based on the data D, that best approximates $f(x)$
- Therefore:
  - $\mathrm{Bias}\left[\hat{f}\right]^2 = \mathrm{E}\left[\hat{f} - f\right]^2 = \mathrm{E}\left[c - f\right]^2$
  - $\mathrm{Var}\left[\hat{f}\right] = \mathrm{Var}\left[c\right] = 0$
- Implication: variance term will be zero, but the bias will be large, because the network has made no attempt to fit the data

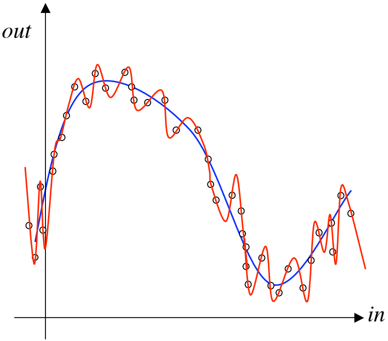
Extreme Case of Bias and Variance: Over-fitting

- Suppose the neural network is very hard working and makes sure that it exactly fits every data point, which means $\hat{f} = t$ and $\mathrm{E}\left[\hat{f}\right] = \mathrm{E}(t) = \mathrm{E}(f + \epsilon) = f$
- Therefore:
  - $\mathrm{Bias}\left[\hat{f}\right] = \mathrm{E}\left[\hat{f} - f\right] = 0$
  - $\mathrm{Var}\left[\hat{f}\right] = \mathrm{Var}\left[t\right] = \sigma^2$
- Implication: the bias is zero, but the variance is the square of the noise on the data, which could be substantial.

Extreme Case of Bias and Variance: Under vs Over-fitting



Ignore the data ⇒
Big approximation error (high bias)
No variation between data sets (no variance)



Fit every data point ⇒
No approximation error (zero bias)
Variation between data sets (high variance)

Overview and Reading

- We began by looking at the computational power of MLPs.
- Then we saw why the generalization is often better if we dont train the network all the way to the minimum of its error function.
- A statistical treatment of learning showed that there was a trade-off between bias and variance.
- Both under-fitting (resulting in high bias) and over-fitting (resulting in high variance) will result in poor generalization.
- There are many ways we can try to improve generalization.
- Further reading:
  - Bishop: Sections 6.1, 9.1
  - Haykin-2009: Sections 2.7, 4.11, 4.12