Generalisation

Tirtharaj Dash

Dept. of CS & IS and APPCAIR
BITS Pilani, Goa Campus

September 9, 2021

Where we are:

Summary of the previous lecture:

- Backpropagation with tensor-level computation

I told you earlier that we build models for unseen data (generalisation).
Today, we intend to know (albeit in brief):

- Model capacity and Model complexity
- Model selection
- Underfitting
- Overfitting

Generalisation

- The ability of a model to perform well on previously unobserved inputs is called *generalisation*.
- It is only possible if our model could discover *general patterns* that captures regularities in the underlying population from which our training set was drawn.
- Example: A model for predicting patients' dementia status from their genetic markers.
  - It is possible for a model to just *memorise* the entire genetic markers dataset.
  - But, this kind of model will not able to reliably predict the dementia status for an unseen patient.

Generalisation Error I

- We only have access to a *training set* to build a model from.
- We can compute some error measure on this training set.
- An optimisation procedure helps us in iteratively reducing the training error and obtain a (best) set of parameters:
  1. Given a training set, $\mathcal{D}_{tr}$
  2. Randomly initialise model parameters, $\mathbf{w}^{(0)}$
  3. While error is not *minimum*:
     a. Update model parameters: $\mathbf{w}^{(new)} \leftarrow \mathbf{w}^{(old)} \pm \Delta$
  4. Return training error, model with $\mathbf{w}^*$

These details are not relevant at this point:

1. We are assuming here a fixed structure π for our model.
2. Δ is the amount of update to be done to $\mathbf{w}^{(old)}$.

Generalisation Error II

- The procedure in the previous slide is just 'optimisation'.
- Question: Is ML more than just optimisation?
  - Yes. In addition to the error on the training set, we also want to reduce the error on a (unseen) test set (also called, generalisation error).
- Generalisation error is defined as the expected value of the error on a new input.
  - The expectation is taken across different possible inputs, drawn from the distribution of inputs we expect the system to encounter in practice.

Estimating Generalisation Error I

- We typically estimate it by measuring the model's performance on a test set of examples that were collected separately from the training set.
- Denoting the test set as $\mathcal{D}_{te} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n^{(test)}}$.
- Generalisation error is then:

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

where L denotes some loss measure.

Estimating Generalisation Error II

Issue:

- If $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$ were collected arbitrarily; we cannot estimate the generalisation error.

Estimating Generalisation Error III

Solution: Assume that we are allowed to make some assumptions.

- $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$ are generated by a probability distribution over datasets called the **data-generating process**.
- We make certain assumptions, known as **i.i.d.** assumptions:
  - The examples in each dataset are **independent** from each other.
  - $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$ are **identically distributed**, drawn from same probability distribution as each other.

Note: I am not going to tell you how each y_i is obtained. It is actually sampled from a distribution. Please refer to the extra material that is available in the course page (in the context of linear regression; but, the principle is the same.)

Estimating Generalisation Error IV

- These assumptions allow us to describe the data-generating process with a probability distribution over a single example.
- The same distribution is then used to generate every train example and every test example. We can call this shared distribution as p_{data}.
- This probabilistic framework and the i.i.d assumptions enable us to study the relationship betwee training error and generalisation error.

Estimating Generalisation Error V

Connection between training error and generalisation error:

- Suppose we have a distribution $p(\mathbf{x}, y)$.
- $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$ are obtained by repeated sampling from this $p(\mathbf{x}, y)$.
- For a randomly selected model with some fixed $\mathbf{w}$, the expected training error and test error are exactly the same.
- The only difference is based on the name of the dataset: Train-set or Test-set.

There is no learning involved here. The model is fixed beforehand, unlike in machine learning, where a model is built from the train-set. See next:

Estimating Generalisation Error VI

w is not known beforehand:

- We sample $\mathcal{D}_{tr}$ and use it to choose **w** that reduces the error on this sample.
- Then, we sample $\mathcal{D}_{te}$.
- Under this process, the expected generalisation error is greater than or equal to the expected value of the training error.

Estimating Generalisation Error VII

We determine how well our model will perform will be based on:

1. How well it performs on the training set? (We want it to be small)
2. How well it closes the gap between training and test set? (We want this gap to be small)

The above two factors correspond to: **underfitting** and **overfitting**.

Note: The usage of the word 'small' is relative.

Underfitting and Overfitting I

- *Underfitting*: the model is not able to obtain a sufficiently low error value on the training set.
- *Overfitting*: the gap between the training error and test error is too large.
- Underfitting and overfitting are related to **model capacity**.

Underfitting and Overfitting II

- *Capacity*: Informally, a model's capacity is its ability to fit a wide variety of functions.
  - Low capacity models may not fit the training set.
  - High capacity models can overfit by memorizing some properties of the training set that do not serve them well on the test set.
- *Model complexity*: Model capacity is an informal term used for model complexity, albeit not synonymous.

Underfitting and Overfitting III

- Model capacity is controlled by appropriately *choosing* a **hypothesis space**.
- *Hypothesis space*: Space of all functions (model structures) that the learning algorithm is allowed to select as being the solution.
- Examples:
  - A model with no hidden layer is limited to represent only simple function of the form:

$$f(\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{x}\mathbf{w})$$

  - A model with ℓ layers can represent more complex functions:

$$f(\mathbf{x}; \mathbf{w}) = \sigma(\sigma(\ldots\sigma(\mathbf{x}; \mathbf{w}^{(1)}); \mathbf{w}^{(2)}); \mathbf{w}^{(\ell)})$$

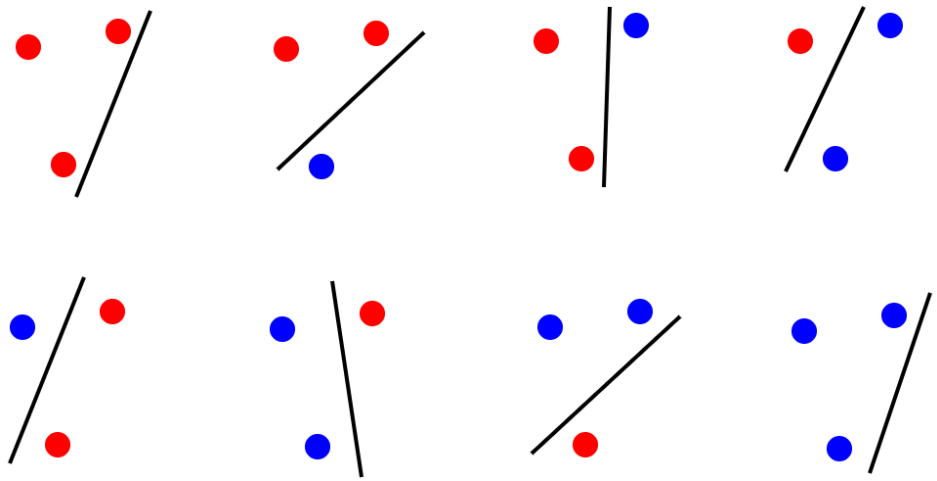
(This is serious abuse of notations!)

Underfitting and Overfitting IV

- *Occam's razor* (c. 1287–1347): Among competing hypotheses that explain known observations equally well, we should choose the "simplest" one.
- *VC dimension*:
  - Vapnik-Chervonenkis dimension, named after Vladimir Vapnik and Alexey Chervonenkis.
  - It measures the capacity of a binary classifier.
  - Defined as the cardinality of the largest set of examples (denoted as n) that the binary classification algorithm can *shatter*, meaning, the algorithm can always learn a perfect classifier for any labelling of n data points.

Underfitting and Overfitting V

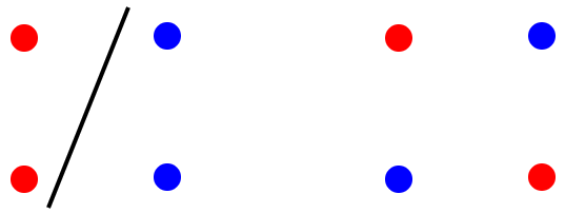
- Example 1: For $n = 3$, there are $2^3 = 8$ possible labelling of these 3 points, such as $\{0, 0, 0\}$ through to $\{1, 1, 1\}$ (as shown below: circle: 1, circle: 0)



  - If the hypothesis space is a set of linear classifier in two-dimension.
  - The diagram above shows: There exist sets of 3 (non colinear) data points that can be shattered by the model, irrespective of the labelling used.

Underfitting and Overfitting VI

- Example 2: For $n = 4$, there are $2^4 = 16$ possible labellings.



  - Notice that although you might get a linear classifier that can *classify* the points for some labelling of these 4 points.
  - The linear classifier cannot shatter these 4 points, that is, not all possible labellings can be shattered by a linear model.

Underfitting and Overfitting VII

- *Ideal Model*:
  - It is an oracle that simply knows the true distribution $p(\mathbf{x}, y)$.
  - Even this model will still incur some error on many problems, because of noise in the distribution.
  - Also, the mapping from $\mathbf{x}$ to y maybe inherently stochastic, or maybe a deterministic function that involves other variables than those in $\mathbf{x}$.
  - The error incurred by an oracle making predictions from $p(\mathbf{x}, y)$ is called the **Bayes error**.

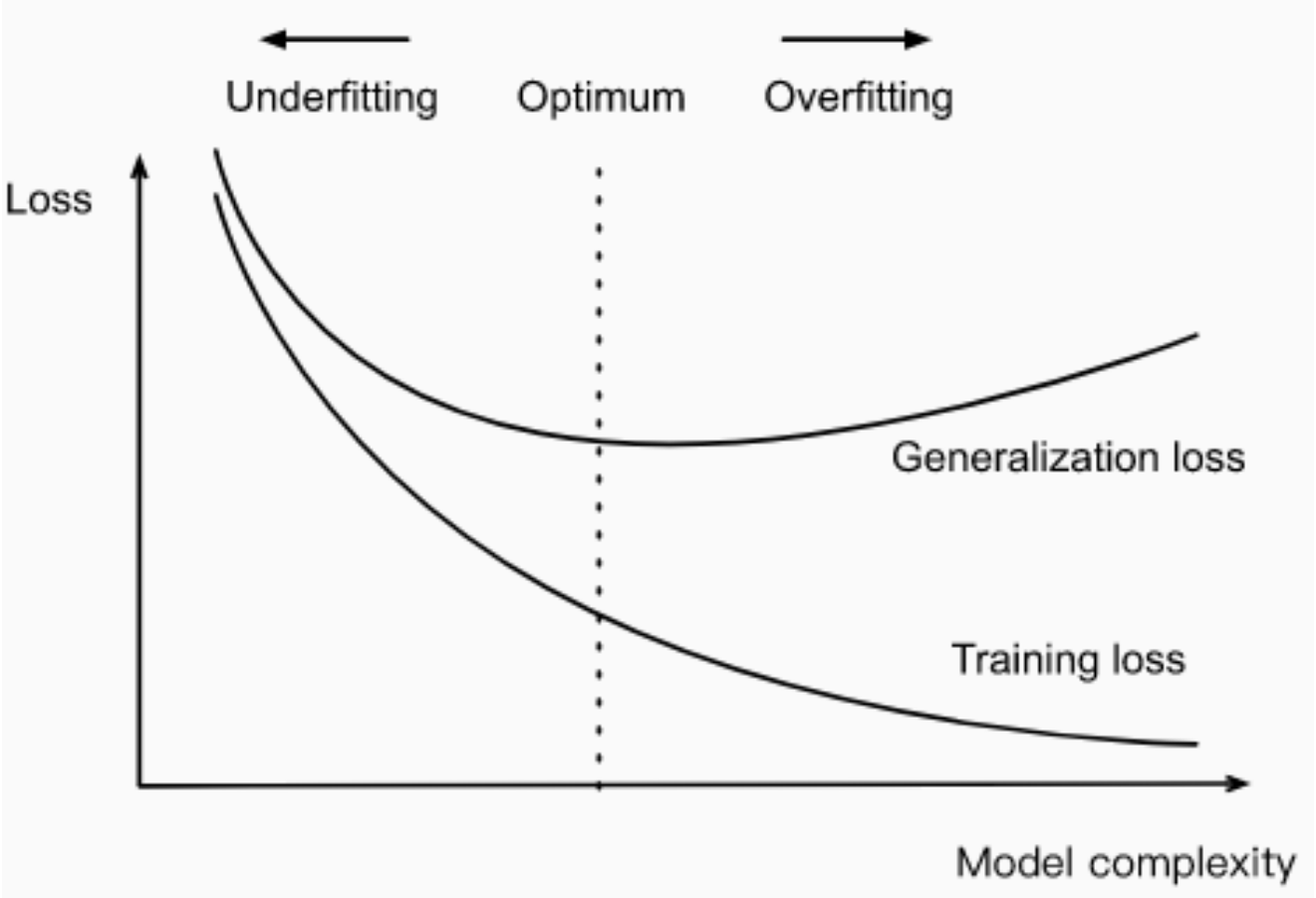
Note: Computational Learning Theory (COLT) or Statistical Learning Theory presents significant theoretical aspect of these concepts. And, for this course, it is sufficient to know this much.

Model Complexity I

- Determining what factors precisely consitute model complexity is a complex matter.
- But, at this point, we will adhere to the following:
  - A model with more parameters might be considered more complex.
  - A model whose parameters can take a wider range of values might be more complex.
  - A model that takes more training iterations is more complex, and one subject to *early stopping* (fewer training iterations) as less complex.
- Further, it is difficult to compare complexities across model classes (for example, decision trees vs. MLPs).

Model Complexity II

Influence of model complexity on overfitting and underfitting:



Model Complexity III

Factors affecting generalisability of a model class:

The number of tunable parameters When the number of tunable parameters, sometimes called the *degrees of freedom*, is large, models tend to be more susceptible to overfitting.

The values taken by the parameters When weights can take a wider range of values, models can be more susceptible to overfitting.

The number of training examples It is trivially easy to overfit a dataset containing only one or two examples even if your model is simple. But overfitting a dataset with millions of examples requires an extremely flexible model.

Model Selection

- Selecting a final model after evaluating several candidate models.
- Here we are concerned with selcting models from the same class such as MLPs.
- In MLPs, this means:
  - Comparing members that have been trained with different hyperparameter settings.
  - Example: compare models with across (a) numbers of hidden layers, (b) different number of hidden units, and (c) various choices of hte activation functions.
- Choice of a (best) final model is based on the model's performance on a *validation dataset*.