DL Components and a Bayesian Perspective

Tirtharaj Dash

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

August 31, 2021

Deep Learning: Components I

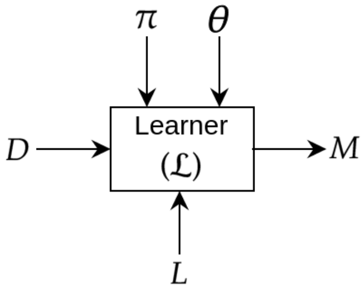
D: Dataset

M: Model

π: Structure of the model

$\boldsymbol{\theta}$: Parameters of the model

L: A loss function



(T.Dash et al.: https://arxiv.org/abs/2107.10295)

Deep Learning: Components II

Dataset, D A dataset consists of “labelled” examples. Each element in the dataset is a pair $(\mathbf{x}, y)$, where $\mathbf{x}$ is a data-point and y is its class-label (for classification) or a real-value (for regression).

Model, M Model refers to the mathematical machinery for ingesting data ($\mathbf{x}$s) and computing a set of prediction(s). We are concerned with complex problems which cannot be solved by simple statistical models. Therefore, our interest is in models that consist of sucessive transformations of the data that are chained together top-to-bottom (“deep models”). These models are known as “deep neural networks” or “deep nets”.

Deep Learning: Components III

Structure, π Each deep network has a structure (synonym: architecture). It mainly refers to: number of layers, number of neurons in each layer, and interconnection between the neurons in any two layers.

Parameters, $\boldsymbol{\theta}$ A parameter in a deep network refers to a connection strength or synaptic weight. Each connection is associated a (real-valued) weight.

Deep Learning: Components IV

Loss function, L We need to have formal measures of how good (or bad) our model is. We will call these loss functions. The term "loss" roughly translates to "error". While constructing a deep model, our intention will always be to minimise the error.

Deep Learning: Components V

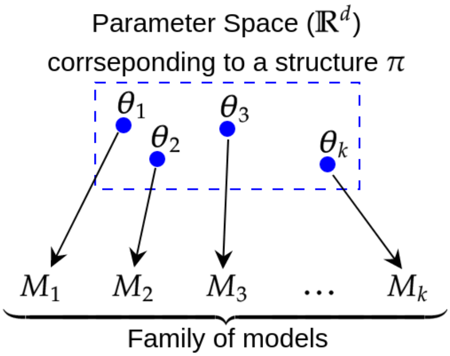
Learner, $\mathcal{L}$ This refers to the learning algorithm that uses the dataset D to choose the best set of possible parameters $\boldsymbol{\theta}$ for the model M with structure π by minimizing the loss function L. It has access to a learning algorithm, for example, mini-batch gradient descent.

Deep Learning: Components VI

Notes:

- The set of parameter $\boldsymbol{\theta}$ of a deep model is directly related to its structure π.
- Manipulating $\boldsymbol{\theta}$ results in a set of instances of deep model, each with same structure *pi*.
- The set of all distinct model instances is called a **family of models**.
- $\mathcal{L}$ is a meta-program that uses D to choose a best set of parameters $\boldsymbol{\theta}^*$.

Deep Learning: Components VII



- "Learning" refers to a **search** over these models.
- As you can see there can be infinitely many model instances in a family of models.

Deep Learning: Components VIII

- It is impossible to enumerate all the models, let alone the search over these.
- An optimisation procedure (such as Gradient Descent) provides a set of steps to make this search feasible.
- But, what does this “search” really mean in Bayesian formalism? – Let’s see this next.

Deep Learning from a Bayesian Perspective I

- A model is defined by its structure (π) and its parameters ($\mathbf{w}$). (Just following the convention of writing $\boldsymbol{\theta}$ as a set of weights $\mathbf{w}$)
- A search over the model instances given some model structure π would refer to a search over these parameter space.
- We represent our prior beliefs on what the model parameters are. This results in a prior distribution over model's parameters: $p(\mathbf{w})$.
- As we collect more data $\mathbf{D} = (\mathbf{X}, \mathbf{Y})$, we upate the prior distribution and turn it into a posterior distribution:

$$p(\mathbf{w}|\mathbf{D}) = \frac{p(\mathbf{D}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{D})}$$

Deep Learning from a Bayesian Perspective II

That is:

$$p(\mathbf{w}|\mathbf{X},\mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{X},\mathbf{w})p(\mathbf{X})p(\mathbf{w})}{p(\mathbf{Y}|\mathbf{X})p(\mathbf{X})}$$

or,

$$p(\mathbf{w}|\mathbf{X},\mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{X},\mathbf{w})p(\mathbf{w})}{p(\mathbf{Y}|\mathbf{X})}$$

The first term in the numerator is called the **likelihood** and it represents how likely the data is, given the model's parameters **w**.

Deep Learning from a Bayesian Perspective III

- A neural net's goal is to estimate the likelihood $p(\mathbf{Y}|\mathbf{X}, \mathbf{w})$. (Recall that: you were maximising the likelihood by minimizing the MSE for linear models)
- To find the best model weights are the Maximum Likelihood Estimates (MLE) of the weights:

$$\begin{aligned}\mathbf{w}^{MLE} &= \underset{\mathbf{w}}{\operatorname{argmax}} \; \log p(\mathbf{D}|\mathbf{w}) \\ &= \underset{\mathbf{w}}{\operatorname{argmax}} \; \sum_i \log p(y_i|\mathbf{x}_i, \mathbf{w})\end{aligned}$$

Deep Learning from a Bayesian Perspective IV

- Alternatively, we can use our prior knowledge, represented as prior distribution over the weight parameters and maximise the *posterior* distribution. This is called the Maximum Aposteriori Estimates (MAP) of **w**:

$$\begin{aligned}
\mathbf{w}^{MAP} &= \underset{\mathbf{w}}{\operatorname{argmax}} \ \log p(\mathbf{w}|\mathbf{D}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \ \log p(\mathbf{D}|\mathbf{w}) + \log p(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \ \sum_i \log p(y_i|\mathbf{x}_i, \mathbf{w}) + \log p(\mathbf{w})
\end{aligned}$$

Deep Learning from a Bayesian Perspective V

- The term $\log p(\mathbf{w})$ acts as a regularization term.
- Choosing a Gaussian distribution with mean 0 as the prior, we get the mathematical equivalence of L_2 regularisation.

Deep Learning from a Bayesian Perspective VI

Summary:

- We intend to find a posterior distribution over weights $\mathbf{w}$

$$p(\mathbf{w}|\mathbf{D}) = \frac{p(\mathbf{D}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{D})} = \frac{p(\mathbf{D}|\mathbf{w})p(\mathbf{w})}{\int p(\mathbf{D}|\mathbf{w})p(\mathbf{w})d\mathbf{w}}$$

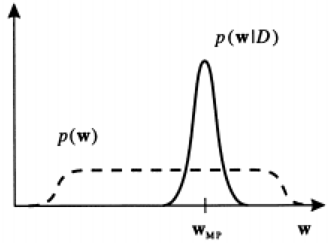
  The denomiator turns out to be a normalisation constant, and therefore, we can simply write:

$$p(\mathbf{w}|\mathbf{D}) \propto p(\mathbf{D}|\mathbf{w})p(\mathbf{w})$$

- In simple words, the search for models will not be affected if we just use the numerator as a metric.

Deep Learning from a Bayesian Perspective VII

- Learning the weights means changing our belief about the weights from prior $p(\mathbf{w})$ to posterior $p(\mathbf{w}|\mathbf{D})$ as a consequence of seeing data $\mathbf{D}$:



(Actually, we should be using the notation f instead of p as these are density functions. Conventionally, p is used to a denote probability mass function.)

A re-look at the likelihood I

- The principles of MLE and MAP provide a guide for how to design a good cost function for nearly any kind of outputs.
- Let's re-write the MAP:

$$\mathbf{w}^{MAP} = \underset{\mathbf{w}}{\operatorname{argmax}} \sum_i \log p(y_i|\mathbf{x}_i, \mathbf{w}) + \log p(\mathbf{w})$$

A re-look at the likelihood II

- One of the goals is to maximise the likelihood term $p(\mathbf{y}|\mathbf{x}, \mathbf{w})$. Instead of maximising the likelihood.
  - equivalently, maximise the log of likelihood.
  - equivalently, minimise the negative of the log of likelihood.
- That is: if we define a conditional distribution $p(\mathbf{y}|\mathbf{x}, \mathbf{w})$, MLE or MAP suggest that we use $-\log p(\mathbf{y}|\mathbf{x}, \mathbf{w})$ as the **cost function**.
  - Recall that this is what we exactly did for logistic regression and perceptron (c.f. tutorial).

A re-look at the likelihood III

- In general, we can think of the neural network as representing a function $f(\mathbf{x}, \mathbf{w})$.
- The output of $f(\mathbf{x}, \mathbf{w})$ are not direct predictions of the value y.
- $f(\mathbf{x}, \mathbf{w}) = \boldsymbol{\omega}$ provides the parameters for a distribution over y.
- Our loss function can then be interpreted as $-\log p(\mathbf{y}|\boldsymbol{\omega}(\mathbf{x}))$.