Deep Learning with Python Ch. 5 Fundamentals

The primary goal of machine learning

- **Generalization**
  - We want to learn a set of weights that will correctly solve problems that haven't been seen before by the system.
  - In early epochs the system will be **underfitting**, then it reaches a point of robust fit (the best fit that it will get with the current settings), then is often moves into **overfitting**.

What can cause overfitting?

- Noisy data (look at MNIST examples)
- Mislabeled data (inevitable in large datasets)
- Ambiguous features
- Rare features and spurious connections (you can see the effect of spurious connections if you add extra white noise data)

The nature of generalization

- A remarkable fact about deep learning systems is that they can be trained to fit anything as long as they have enough representational power.
- See the example that randomly shuffles MNIST labels. There is no relationship between the inputs and the labels, but the training loss goes down just fine. Of course, the validation loss does not improve at all and there is no chance of generalization.
- In fact, you don't even need MNIST examples. You could generate white noise inputs and random labels and train just fine.

The true nature of generalization

- It turns out the nature of generalization in deep learning has little to do with deep learning methods and much to do with the structure of information in the real world.
- **The Manifold Hypothesis**
- The MNIST data is a 28x28 array of integers between 0 and 255. The total number of possible configurations is thus 256 raised to the power 784, much greater than the number of atoms in the universe. But very few of these inputs would look anything like a valid digit.

Generalization continued

- The subspace of valid handwritten digits is continuous. You can make small changes to a digit and still have a digit. All the samples are connected by smooth paths that run through the subspace.
- In technical terms, you would say that the handwritten digits form a **manifold** within the space of possible 28x28 arrays.
- A manifold is a lower-dimensional subspace of some parent space that is locally similar to a linear (Euclidian) space.

The Manifold Hypothesis

- More generally, the **Manifold Hypothesis** posits that all natural data lies on a low-dimensional manifold within a high-dimensional space where it is encoded. Note that this is a pretty strong statement about the structure of information in the universe.
- It appears to be accurate and it is the reason deep learning works. It is true for MNIST data, but it is also true for human faces, tree mophology, the sounds of the human voice, and even natural language.

more Manifold Hypothesis

- The Manifold Hypothesis implies:
  - Machine learning models only have to fit relatively simple low-dimensional, highly structured subspaces within their potential input space.
  - Within one of these manifolds, it is always possible to interpolate between two inputs, that is, to morph one into another via a continuous path along which all points fall on the manifold.

Interpolation as a source of generalization

- If you work with data points that can be interpolated, you can start to make sense out of points you have never seen before by relating them to other points that are close on the manifold.
- You can make sense of the totality of the space using only a sample by using interpolation to fill in the blanks.
- Note that this interpretation is happening on the manifold, not linear interpolation in the parent space.
- Interpolation can help you to make sense of things that are close to to what you have seen before. This is a form of local generalization.

Humans can do extreme generalization

- Humans can do more than just local generalization.
- Humans can do **extreme generalization**, enabled by cognitive mechanisms other than interpolation, such as abstraction, symbolic models of the world, reasoning, logic, common sense, and innate priors about the world. Together we call these things reason, as opposed to just intuition and pattern recognition. But all of this is part of intelligence.

Training data is paramount

- While deep learning is well suited to manifold learning, the power to generalize is more a consequence of the natural structure of your data than a consequence of any particular property of your model. These means your data needs to be informative and not noisy. So data curation and feature engineering are essential.
- Further, since deep learning is curve fitting, for a model to perform well it needs to be trained on a dense sampling of the input space.
- A "dense sampling" means that the data should densely cover the entirety of the input data manifold. This is especially true near decision boundaries.

Dataset curation

- Spending more effort and money on data collection almost always yields a better return than spending the money on developing a better model.
- Make sure you have enough data. More data will yield a better model.
- Minimize labeling errors. Proofread your labels.
- Clean your data and deal with missing values (more later).
- If you have many features and aren't sure which are most important, do feature selection.

Feature engineering

- See example pg. 143 of book.
- The idea is to apply hardcoded (non-learned) transformations to the data before it goes into the model. The intent is to make learning easier.
- Make the latent manifold smoother, simpler, better organized. But doing this requires understanding the problem in depth.
- This was even more important before deep learning, because the simpler models were incapable of getting past these issues.
- Modern deep learning can overcome many of these issues, but it is still important because good features let you solve the problem more elegantly and with fewer resources, including needing less data.

Early stopping

- One of the things we have already seen is that it is important to find the exact point during training when you have reached the most generalizable fit. We have been regularly overtraining, then going back and rerunning the system when we find the sweet spot. There are tools in Keras to do this automatically and it can be important if running the tests is particularly expensive and time consuming.

Regularlizing the model

- **Regularization techniques** are a set of best practices that actively impede the model's ability to perfectly fit the training data, with the goal of making the model perform better during validation.
- It tends to make the model simpler, with a smoother curve, and more generic.
- But measurement is important as you are doing this.

Regularization techniques

- Reducing the network's size: This will limit the resources the network has to memorize the training set. We are looking for a compromise between not enough capacity and too much capacity. Generally this requires trial and error. You will get experience with this later.
- Adding weight regularization: The idea here is to limit the range of values that the weights can take on. This puts constraints on the complexity of the model and makes the distribution of the weights more regular. You do this by adding a cost for high weights to the loss function of the model.
- Adding dropout: This is due to Geoff Hinton's group at Toronto. The idea is to randomly drop output features of a layer to 0 randomly during training. Typically 20 to 50 percent of the outputs are dropped to 0. At testing time, the dropout doesn't happen but the weights are scaled down to keep the same level of activity.

Summary

- The most common ways to maximize generalization and prevent overfitting:
  - Get more training data and/or better training data.
  - Develop better features.
  - Reduce the capacity of the model.
  - Add weight regularization (for smaller models).
  - Add dropout.