Neural Networks for Pattern Recognition: Session 01

HYPOTHETICAL PREDICTION & COST FUNCTION

In supervised Machine Learning, we have Input $\mathbf{X}$ and actual Target Output $\mathbf{Y}$ and try to create and train a model to learn to produce Hypothetical Prediction $\mathbf{H} = h(\mathbf{X}, \mathbf{W})$ that is close/similar to $\mathbf{Y}$.

We evaluate how good the model is by a (usually non-negative) real-valued scalar **cost function** $c(\mathbf{H}, \mathbf{Y})$. Below are the most common cost functions ($i = 1 \ldots m$ data cases, $j = 1 \ldots n$ predictions per case):

- For real-valued $\mathbf{H}$: $c_{ij} = \frac{1}{2}(h_{ij} - y_{ij})^2$
- For binary $\mathbf{H}$ in binary classification problems: $c_{ij} = -y_{ij}\ln h_{ij} - (1 - y_{ij})\ln(1 - h_{ij})$
- For binary $\mathbf{H}$ in multi-class classification problems: $c_{ij} = -y_{ij}\ln h_{ij}$

We usually measure costs on an average per-case basis, i.e. $c = \frac{1}{m}\sum\limits_{i=1}^{m} c_i$.

Cost functions tend to zero when $\mathbf{H}$ is very close to $\mathbf{Y}$, and tend to $+\infty$ when $\mathbf{H}$ goes far from $\mathbf{Y}$.

CREATION, TRAINING & PREDICTION

There are 3 fundamental things we do with a supervised Machine Learning model:

1. **CREATION**: We fix the structure of the prediction function h, including how many weights there are in $\mathbf{W}$; usually $\mathbf{W}$ is initialized at some random values around zero, so that the model does not store any knowledge at first (*note: it is important NOT to initialized* $\mathbf{W}$ *at zeros*)
2. **TRAINING**: We help the model with the above structure **learn** through looking at a **Training data set** one or multiple times and adjust weights $\mathbf{W}$ to **make the average Training-set cost decrease** to a low level
3. **PREDICTION**: We **evaluate** the learned model by making predictions on a **Test data set** that the model has **never seen before** and compute the **average Test-set hypothesis-vs.-target cost**; the Test performance is the final result we care about

MODEL CAPACITY: UNDERFITTING & OVERFITTING

Classic **theory-driven linear/logistic regression models** are **small-capacity** Machine Learning models: their structures are simple, and their underlying theories limit the number of input data features they admit. The resulting predictive functions are consequently relatively unsophisticated. If such unsophisticated predictive functions do not do well on Test data sets, then we have an **under-fitting** problem.

By contrast, with Big Data, complex model structures (e.g. Neural Networks) and a relative disregard of theory, we can build very **large-capacity** models that perform sophisticated behaviors that can mimick very intricate relationships. However, when we train a large-capacity model on Training data, its internal flexibility will make it learn not only the overall generalizable rules but also **specific, non-generalizable idiosyncracies** in the Training data. The latter makes the model **over-fit** the Training data and underperform on Test data.

REGULARIZATION: KEEPING MODEL SIMPLE BY LIMITING WEIGHT MAGNITUDES

Given Big Data and a lack of theory, it is inevitable to create large-capacity Machine Learning models. One way to prevent/mitigate the inherent risk of over-fitting is to **limit the magnitudes of the weights**. (**small weights make simpler models** - this is a great insight that may take some time to sink in)

There a various ways to limit weight magnitudes, and they can be used independently or as a combo:

- **Penalize weight magnitudes**: just for cost function optimization on the Training set, augment the cost function with an extra term related to the magnitudes of the weights, so that large weights impose additional cost
- **Use a Validation data set & return to the point after which Validation performance starts deteriorating**: use gradual optimization procedures to slowly adjust $\mathbf{W}$ through multiple repeated looks at Training data, and track a **pair of Learning Curves** that measure average costs (*EXCLUDING any weight penalty*) on the Training and Validation data sets. Pick the instance of $\mathbf{W}$ that corresponds to the best Validation performance