Regularization*

Khanh Nguyen

June 14, 2015

1 Overview

Overfitting is a persistent problem in machine learning. Although, in most cases, training machine learning models requires maximizing the likelihood of training data, the ultimate goal is about making the model be able to generalize its predictive capability to unseen data. Therefore, if the training objective function consists of only the likelihood of the training dataset, the model will perform poorly on an unseen dataset if its distribution is different from the training set. This phenomenon is known as *overfitting*. From a different perspective, overfitting is the case when the model's variance is high, i.e. it alters dramatically on different datasets.

To avoid overfitting, a model should be given information beyond its training dataset. Hence, the general purpose of regularization methods is to inject *inductive bias* into the model. For example,

- Prevent the training algorithm from picking undesirable parameters (e.g. all-zero or overflowed parameters).
- Direct the training algorithm to picking desirable parameters (e.g. sparse parameters).
- Represent the prior(s) of the data generating process. This prior acts as a list of preference scores over all parameter configurations. This is the generalization of the two points above.

Regularization can be achieved by several ways. One of the most popular forms of regularization is called *regularization penalty*. In this case, an extra term that is dependent on the model parameters is added to the training objective. This term is then trained jointly with the log likelihood of the training data. There is also a regularization strength constant to control the degree of regularization. Regularization penalty is successfully used in many classic models such as logistic regression or conditional random fields. However, there are two major drawbacks with this approach: (a) non-smooth regularization terms might hinder optimization; (b) tuning the regularization strength requires training the model many times. Hence, it is not very suitable for non-convex and complex models such as large-scaled neural networks. Modern approach on regularization for these types of models involves techniques that do not modify the training objective. Two most common ones are *early stopping* and *dropout*. At first, they are not intuitively recognized as regularization techniques. However, their effects on the controlling model parameters has been

*This material is for self-study only.

shown. For example, one can show that early stopping in linear regression is equivalent to L2-regularization. Moreover, they are easy to implement but still efficient in practice.

In this note, I will discuss all of those regularization methods mentioned so far. This material is largely based on the book *Deep Learning* by Yoshua Bengio, Ian J. Goodfellow and Aaron Courville, and the book *Machine Learning: A Probabilistic Perspective* by Kevin Murphy.

2 L2-regularization

The L2-regularizer, also known as weight decay, penalizes the training objective by a norm-2 of the parameter vector (or the weight vector) w:

$$J(w) = NLL(w) + \frac{1}{2}\alpha \left\|w\right\|_2$$

where NLL(w) is the negative log-likelihood of the training data and $\|w\|_2 = \sum_i w_i^2$, and α is the regularization strength [1].

The gradient of the object with respect to w is:

$$\nabla J(w) = \nabla_w NLL(w) + \alpha w \tag{1}$$

Consider a quadratic approximation of the negative log-likelihood near the neighborhood of the optimal value parameters w^*:

$$\hat{NLL}(w) = NLL(w^*) + \frac{1}{2}(w - w^*)^T H (w - w^*)$$

where H is the Hessian matrix of $NLL(w)$ with respect to w [2].

Then,

$$\nabla \hat{NLL}_w(w) = H(w - w^*) \tag{2}$$

Replace $\nabla_w NLL(w)$ in (1) by the approximation in (2) and set the gradient to be zero, we have:

$$H(w - w^*) + \alpha w = 0$$

or

$$w = (H + \alpha I)^{-1} H w^*$$

We observe the followings from the above equation:

- As λ goes to, w approaches its optimal value w^* as when training with the log-likelihood only.
- Since H is symmetric, $H = Q\Lambda Q^T$, for some orthonormal matrix Q and some diagonal matrix Λ. Then:

$$w = Q(\Lambda + \alpha I)^{-1} \Lambda Q^T w^*$$

[1] Notice that to penalize the objective, we have to "add" the regularizer since the goal is to minimize the objective.

[2] The gradient term does not appear since w^* is an optimum.

We interpret the left hand side as rotating $w*$ to the space spanned by the eigenvectors of H, scaling each component i by a factor of $\frac{\lambda_i}{\lambda_i+\alpha}$, then rotating it back to the original space.

This process explains the name *weight decay*. Each component of the parameters is suppressed. The strength of suppression varies depending on the magnitude of each component. Particularly, when λ_i is negligible, the α term in the denominator will dominate and scale the corresponding component to almost zero. Hence, the model will be more likely to go with a set of small-valued parameters. As a result, the variance of the parameters is limited to a smaller range so overfitting will be mitigated.