Chapter 7. Regularization for Deep Learning

A central problem in machine learning is how to make an algorithm that will perform well not just on the training data, but also on new inputs.

Regularization

Any modification we make to a learning algorithm that is intended to reduce its regularization error but not its training error.

Two ways of imposing regularization

- Encode specific kinds of **prior knowledge**
- Express a **generic preference for a simpler model class** in order to promote generalization

Trading increased bias for reduce variance

The Importance of regularization

Controlling the complexity of the model is not intended to find the model of the right size with the right number of parameters. Instead, we might find and indeed in practical deep learning scenarios, we almost always do find that the best fitting model is a large model that has been regularized properly

7.1 parameter Norm Penalties

Regularization by adding a parameter norm penalty $\Omega(\theta)$ to the objective function $j(\theta)$

$$\tilde{J}(\theta; X, y) = J(\theta; X, y) + \alpha\Omega(\theta)$$

- For neural network, the parameter norm penalty **only penalize the weights of the affine transformation but not the bias**
    - Bias do not impose too much variance (why ???)
    - Regularize bias will lead to underfitting

7.1.1 L^2 Parameter Regularization

In L^2 norm penalty, the regularization term is

$$\Omega(\theta) = \frac{1}{2}||w||_2^2$$

L^2 regularization is also called **ridge regression** or **Tikhonov regularization**

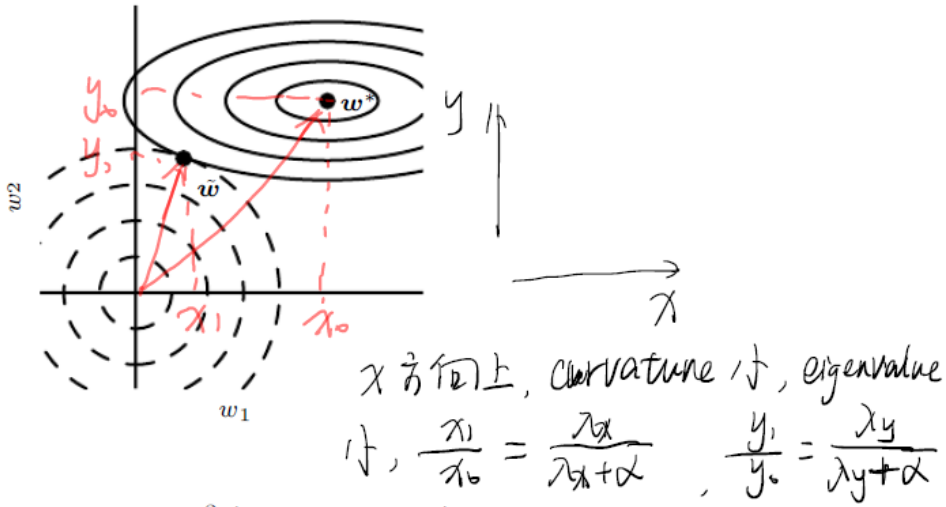

Figure 7.1: An illustration of the effect of L^2 (or weight decay) regularization on the value of the optimal $\boldsymbol{w}$. The solid ellipses represent contours of equal value of the unregularized objective. The dotted circles represent contours of equal value of the L^2 regularizer. At the point $\tilde{\boldsymbol{w}}$, these competing objectives reach an equilibrium. In the first dimension, the eigenvalue of the Hessian of J is small. The objective function does not increase much when moving horizontally away from $\boldsymbol{w}^*$. Because the objective function does not express a strong preference along this direction, the regularizer has a strong effect on this axis. The regularizer pulls w_1 close to zero. In the second dimension, the objective function is very sensitive to movements away from $\boldsymbol{w}^*$. The corresponding eigenvalue is large, indicating high curvature. As a result, weight decay affects the position of w_2 relatively little.

对 Cost function 影响大的 value penlty 小.
对 cost function 影响小的 penlty 大

The effects of L^2 regularization

- Rescale w along the axes define by the eigenvectors of H, where H is the hessian matrix of $J(\theta)$

7.1.2 L^1 Regularization

In L^1 regularization, the penalty term is

$$\Omega(\theta) = \alpha \sum_i |w_i|$$

Effects of L^1 regularization on Quadratic Cost function

- Add L^1 regularization is equal to shift w by $\frac{\alpha}{H_{ii}}$

Properties

- L^1 result in a solution that is more sparse

7.2 Norm Penalties as Constrained Optimization

Instead of using Norm penalties as an additional term to the cost function

$$\tilde{J}(\theta; X, y) = J(\theta; X, y) + \alpha\Omega(\theta)$$

We can add a inequality constrain to the optimization process. And we could construct a generalized Lagrange function

$$\mathcal{L}(\theta, \alpha; X, y) = J(\theta; X, y) + \alpha(\Omega(\theta) - k)$$

The solution to the constrained problem is given by

$$\theta^\star = \arg\min_\theta \max_{\alpha, \alpha \geq 0} \mathcal{L}(\theta, \alpha)$$

Reason to use constraints rather than norm penalty

- Penalties can cause non-convex optimization procedures to get stuck in local minima corresponding to small θ

7.3 Regularization and Under-Constrained Problems

> 这里没看懂是什么意思， 没学过优化很吃亏啊

7.4 Dataset Augmentation

> Overfitting 最根本的原因其实就是训练数据太少了，如果可以增加训练数据，那么就可以解决overfitting的问题， 也就等价于加了regularization

Main Idea

- Add training dataset by creating fake examples
- Useful for classification tasks

Used in image classification

- Adding additional training examples by rotating or rescaling origin images

Used in speech recognition

- Regularize neural network by adding noise to input
- By adding noise to hidden units

7.5 Noise Robustness

Key points

- For some models, adding noise with infinitesimal variance at the input of the model is equivalent to imposing a norm penalty on the weights
- Noise injection to hidden units is powerful regularization tool
- Noise injection to weights is used primarily in the context of recurrent neural networks

7.5.1 Injecting Noise at the Output Targets

Key points

- Most datasets have training example with mislabeled y, which is harmful to the training algorithm
- **Label smoothing** can regularize the model in this case

7.6 Semi-Supervised Learning

Key points

- Adding unsupervised learning model help to reduce the overfitting problem and thus conduct regularization

Semi-supervised Learning

- Bothe $P(x)$ and $P(x, y)$ are used to estimate $P(y|x)$

Separate unsupervised and supervised learning

1. Use unsupervised learning to learn a new **representation** $h = f(x)$
2. Use the transformed representation h as the input to supervised learning model

Example: Use PCA first, and then use the transformed features at the input to the supervised learning model

Share parameter between unsupervised and supervised learning

> 这里的叙述没看懂

7.7 Multi-Task learning

Key points

- A model shared by multi-task tends to be more general, which avoid overfitting and regularize

Disadvantage

- Only useful when there exists several tasks and have some shared factors of variations

7.8 Early Stopping

Key points

- We tend to use large model with sufficient representational capacity to overfit the task
- The relationship between train error and test error are U-shape as following image
- We can stop early to reduce the generalization error
- Need a validation set to conduct early stopping

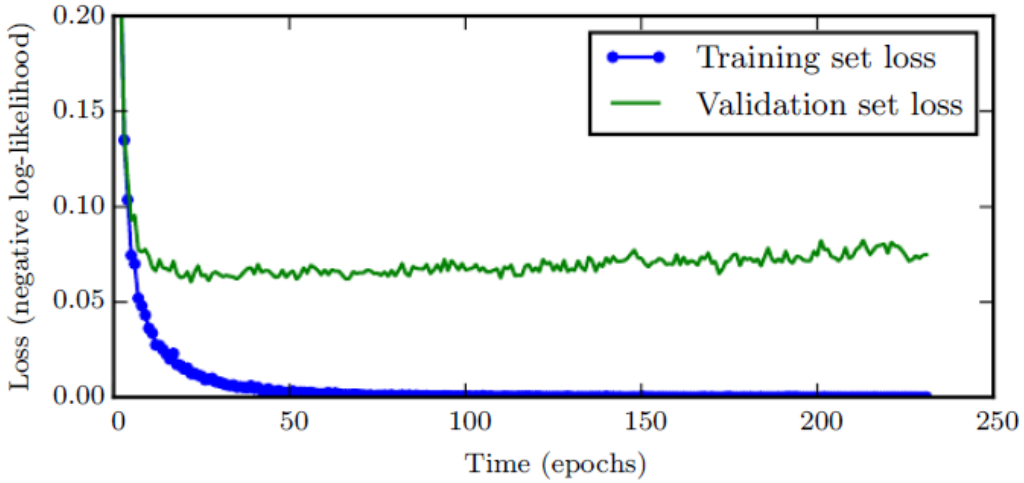

Figure 7.3: Learning curves showing how the negative log-likelihood loss changes over time (indicated as number of training iterations over the dataset, or **epochs**). In this example, we train a maxout network on MNIST. Observe that the training objective decreases consistently over time, but the validation set average loss eventually begins to increase again, forming an asymmetric U-shaped curve.

Why overfitting can regularize

- Early stopping can be regarded as **controlling the effective capacity** of the model by determining how many steps it can take to fit the training set
- The number of training step can be regarded as a hyperparameter. And we use a validate set to choose a proper number of training step

Advantage of early stopping

- Early stopping requires no change in the underlying training procedure, the objective function and the allowable parameter values

Disadvantage of early stopping

- Require a validate set, thus reducing the number of training examples

Solution to the disadvantage

- Initialize the model again and retrain the model on all of the data with the same number of training step as the early stopping procedure determined in the first pass
    - Question about training the same number of epochs or the same number of steps
- Continue the training process with the all the data.
    - How to determine when to stop in this training process

How early stopping acts a a regularizer

- In the case of a simpler linear model with a quadratic cost function and a simple gradient descent, early stopping is equivalent to L^2 regularization

> 这里的推导就省略了，其实也挺好奇对于其他的cost function和task这个结论是不是也成立

7.9 Parameter Tying and Parameter Sharing

> 之前的Penalty-based 或者 constraint-based Regularization都是引入了一些prior knowledge about the suitable values of the model parameters。 比如L^2 regularization penalizes model parameters fro deviating from the fixed value of zero.
>
> 另外一个思路是Add prior knowledge about the relationship between the model parameters

Key point

- CNN 里面其实广泛应用了Parameter sharing. 比如同一层里面，一个filter 其实被整个input features共用的

7.10 Sparse Representations

Sparse parameterization

- L^1 regularization will make the model parameter sparse

Sparse representation

- Apply similar norm penalties to representation

$$\tilde{J}(\theta; X, y) = J(\theta; X, y) + \alpha\Omega(h)$$

7.11 bagging and Other Ensemble Methods

Key points

- **Bagging(short for bootstrap aggregating)** is a technique for reducing generalization error by combining several methods
- Bagging is an example of more general strategy in machine learning called **model averaging**.
- Techniques employing this strategy are know as **ensemble methods**

Performance of ensemble method

- The ensemble method will perform at least as well as any of its members

> 这一点有疑问， 从书中的内容看， 应该说是跟所有members的平均水平一样好吧 。

- If the members make independent errors, the ensemble will perform significantly better

How to use ensemble method

- The member can be different model or the same model with different initial hyperparameters
- The training set of each member is of the same size as the origin training dataset and is generated by sampling the origin training dataset with replacement

> 这里为什么要sample with replacement也有问题。 直接用原来的origin training dataset去训练不同的模型不是能得到更好的member model 从而进一步减少ensemble model的误差么。 不太懂为什么要通过采样得到

7.12 Dropout

How to interpret Dropout

- Add multiplicative noise to the hidden units
- Trains a bagged ensemble of models that share hidden units

Bagging

- Dropout provide an inexpensive way to train and evaluate a bagged ensemble of exponential many neural networks
- Obtain new model by randomly remove hidden unit

Difference with traditional bagging in previous section

- Each model is train by only one step with SGD from mini-batch
- The parameter are shared within all the potential models

How to get the output

> 对于前面提到的bagged， 一般模型的数量是有限的，最后的结果是取所有模型结果的平均值，但是对dropout来说，模型的数量太大了，可以说是每一个training step都用了不一样的模型，最后取全部模型的平均是不可能的，所以就有几种方法
>
> 1. 采样足够多的模型，在跟traditional bagging一样取平均， 需要做多次feedforward
> 2. Geometric mean + one feedforward propagation
>
> 为什么这里要用Geometric mean ？？ 不理解

Weight scaling inference rule

We can approximate $P_{ensemble}$ by evaluating $P(x|y)$ in one model: the model with all units, but the weights going out of unit i multiplied by the probability of including unit i

E.g. If the dropout probability of a layer is 0.5 , then the weights of this layer will be divide by 2 after training.

- Weight scaling inference rule is just an approximation of the ensemble output
- Have not theoretically characterized but works well practically

Advantages of Dropout

- Computation inexpensive
- Adaptive to many type of models
    - Works well with nearly all model that uses a distributed representation and can be trained with SGD

Disadvantages

- Not efficient for when the training dataset is small

Extension to dropout

- Fast dropout
    - Reduce the stochasticity
    - Faster convergence time
    - Indicate stochasticity is not necessary for generalization
- DropConnect
    - Dropout is drop hidden unit
    - DropConnect is drop the connection between each unit (More like sparse representation)
- Use $\mu \sim \mathcal{N}(1, I)$ to replace $\mu \in \{0, 1\}$. Use a real number mask instead of binary mask

Why Dropout can regularize the neural network

- Dropout boosting revel that the stochasticity is not sufficient
- A large potion of the power of dropout arises from the fast that the masking noise is applied to the hidden units
    - Random drop force the hidden unit to be replaceable and learn more distinct feature from the training examples

Multiplicative and additive noise

- If only additive noise is added, the hidden state tends to be very large to make the noise insignificant
- Multiplicative noise(Dropout) do not have this problem
- Batch normalization has both multiplicative and additive noise

7.13 Adversarial Training

Adversarial Example

- We can intentionally construct a input x' near a data point x that the model output is very different at x'

Adversarial Training

- We can use adversarial examples to train the model to regularize

Reason of Adversarial Examples

- Excessive linearity
  - Neural network contains many linear part
  - High dimension data
  - Small change will be accumulated when use linear transform with high dimension data

Virtual adversarial examples

> 这里没看懂

7.14 Tangent Distance, Tangent Prop, and Manifold Tangent Classifier

Assumption

Many machine learning algorithm aim to overcome the curse of dimensionality by assuming that the data lies near a low-dimensional manifold. (Refer to section 5.11.3 Manifold learning)

Tangent Distance Algorithm

- Based on Manifold assumption
- Non-parametric
- Calculated the tangent distance to classify

Tangent prop algorithm

- Add extra penalty by the invariance on the manifold locally
- Closely related to dataset augmentation
- Related to Double Back prop

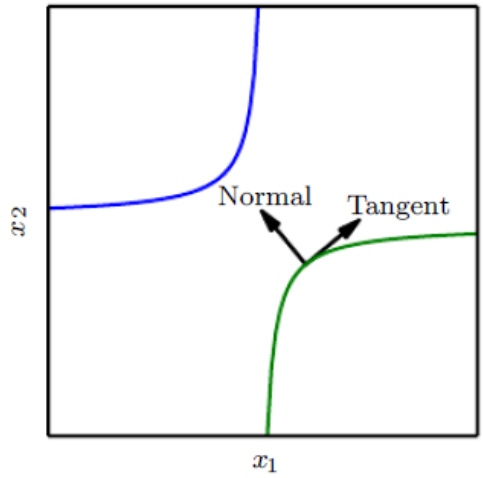

Figure 7.9: Illustration of the main idea of the tangent prop algorithm (Simard *et al.*, 1992) and manifold tangent classifier (Rifai *et al.*, 2011c), which both regularize the classifier output function $f(\boldsymbol{x})$. Each curve represents the manifold for a different class, illustrated here as a one-dimensional manifold embedded in a two-dimensional space. On one curve, we have chosen a single point and drawn a vector that is tangent to the class manifold (parallel to and touching the manifold) and a vector that is normal to the class manifold (orthogonal to the manifold). In multiple dimensions there may be many tangent directions and many normal directions. We expect the classification function to change rapidly as it moves in the direction normal to the manifold, and not to change as it moves along the class manifold. Both tangent propagation and the manifold tangent classifier regularize $f(\boldsymbol{x})$ to not change very much as $\boldsymbol{x}$ moves along the manifold. Tangent propagation requires the user to manually specify functions that compute the tangent directions (such as specifying that small translations of images remain in the same class manifold) while the manifold tangent classifier estimates the manifold tangent directions by training an autoencoder to fit the training data. The use of autoencoders to estimate manifolds will be described in chapter 14.