Regularization techniques for Deep Learning

Vladimir Feinberg

July 17, 2017

1 Regularization

Regularization shrinks the capacity of an ANN. In other words, it restricts the richness of the family of functions which the ANN can take the value of. As Vincent Vanhouke puts it in his Udacity course, ANN regularization can be likened to fitting into tight jeans, where extremely expressive model classes are tightened by generic regularization techniques, which seem to find a "right fit" in many problems, empirically speaking.

At a high level, regularization changes the objective from usual loss $J(\boldsymbol{\theta})$ to loss plus weight penalty $J(\boldsymbol{\theta}) + \Omega(\boldsymbol{\theta})$ or loss plus constraints. In both supervised and unsupervised models this has justification in moving overfit models of high capacity to those of appropriate capacity for the problem: lower variance for a bit higher bias, where the bias and variance are with respect to the loss J on the training set as an estimator for the loss on the entire data distribution.

In the sections that follow, we recount general regularization techniques for ANNs, most of which were collected from Dr. Goodfellow's Deep Learning Book.

2 Weight Decay

For all affine transforms let the weight matrix terms (not bias), vectorized, be $\mathbf{w}$. Add $\Omega(\boldsymbol{\theta}) = \frac{\alpha}{2}\left\|\mathbf{w}\right\|_2^2$. Intuitively, bias is unrestricted because it only interacts with one variable directly (the output feature map), so it isn't prone to overfitting. For a quadratic approximation of the loss at $J(\boldsymbol{\theta})$, with hessian $H = Q\Lambda Q^\top$, L^2 regularization would result in a new optimum $Q(\Lambda + \alpha I)^{-1}\Lambda Q^\top \mathbf{w}_*$ where $\mathbf{w}_*$ is the unconstrained optimum. If $\mathbf{v}_i$ is the i-th eigenvector of H and λ_i its eigenvalue, then the regularization adjusts the solution by rescaling the $\mathbf{v}_i$-th component of $\mathbf{w}_*$ by a factor of $\frac{\lambda_i}{\lambda_i+\alpha}$. Regularization can also be viewed as constraints by identifying that α is a KKT multiplier: a fixed α corresponds to optimizing $\min J(\boldsymbol{\theta})$ such that $\|\mathbf{w}\| \le k$ for some k, which is in practice difficult to find given α (Fig. 1). L^2 regularization is equivalent to Bayesian MAP if weights have $N(0, \alpha^{-1})$ prior.

Proximal Regularization. Similar to weight decay, but $\Omega(\boldsymbol{\theta}) = \alpha\left\|\mathbf{w}\right\|_1$. Assuming a (convex) diagonal Hessian approximation to $J(\boldsymbol{\theta})$ (more strict than above), the i-th component solution is pulled down to

$$w_i = \begin{cases} \left(|w_i| - \frac{\alpha}{H_{i,i}}\right) \operatorname{sgn} w_i & H_{i,i} > 0, |w_i| > \frac{\alpha}{H_{i,i}} \\ 0 & \text{otherwise} \end{cases}$$

L^1 regularization is equivalent to Bayesian MAP if weights have $\text{Laplace}(0, \alpha^{-1})$ prior.

Early Stopping. Stop training if validation increases; even if training is unconverged. If validation set is large, you miss out on a lot of data. More complicated algorithms are necessary to find an applicable early stopping time when re-training to include validation set (see Dr. Goodfellow's DL book, Algorithm 7.3). Early stopping is similar to L^2 regularization, and for quadratic loss it is just that (Fig. 2).

Early stopping can thus be viewed as an efficient way of regularizing weights just the right amount for the problem at hand, whereas we might have to cross-validate multiple settings of α to find an appropriate value for the α hyperparameter. Nonetheless, early stopping still suffers from generalization issues of overfitting to the validation set, see discussions in SjÃűberg and Ljung 1995 and Cataltepe et al 1999.

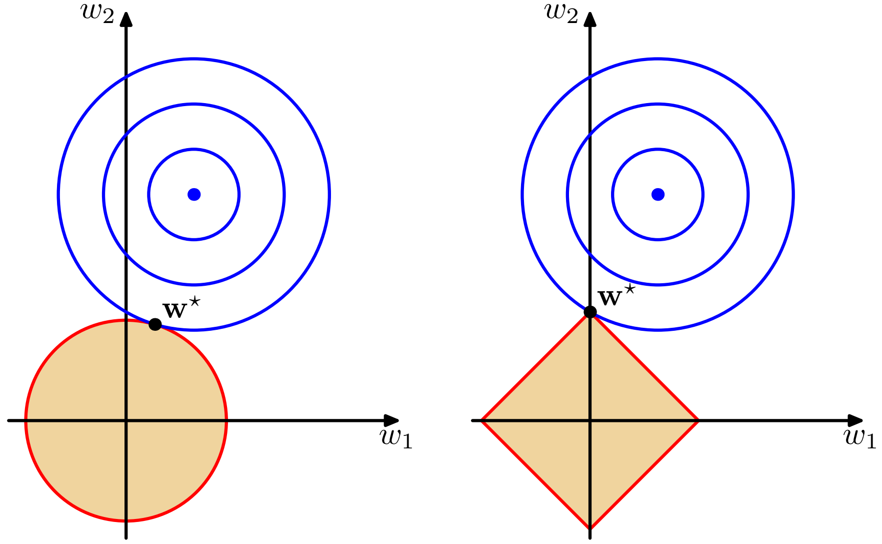
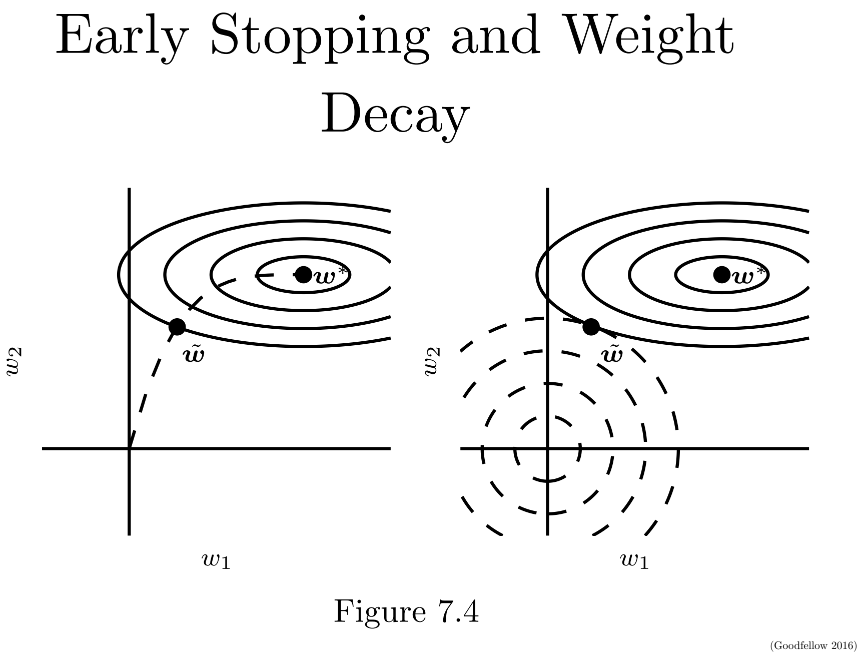


Figure 1: Regularization as constrained optimization: the above shows the contour plots of a quadratic objective, and L^2, L^1 regularizations, respectively, as constraints constructed by weighing the objective against the shape of the regularizer's ball. Image from Bishop's Pattern Recognition and Machine Learning, Chapter 3, page 146.



Early Stopping and Weight Decay

Figure 7.4

(Goodfellow 2016)

Figure 2: From Dr. Goodfellow's DL book lecture, Chapter 7.

3 Dataset Augmentation and Manifold Learning

Modify supervised pairs $(\mathbf{x}, y)$ to include $(T(\mathbf{x}), y)$ where T is a transformation we expect the classifier to be invariant to. Noise injection into inputs, too can be a form of dataset augmentation. An automatic technique for doing this is tangent propagation (Rifai et al 2011), which adds a penalty if the output is sensitive to changes along learned manifold vectors $\mathbf{v}_i$: $\Omega(f) = \sum_i \left(\mathbf{v}_i^\top \nabla f)^2\right)$.

4 Noise Robustness

(Graves 2011). Inject noise into weights during training, forcing learned weights to be robust to small variations.

5 Parameter Sharing

Reuse part of a network for another task, encoding prior knowledge that the feature maps must be the same between two tasks. CNNs, discussed in a separate document (root), are a form of parameter sharing, since the linear matrix W is restricted to a convolution transform.

6 Ensemble Methods

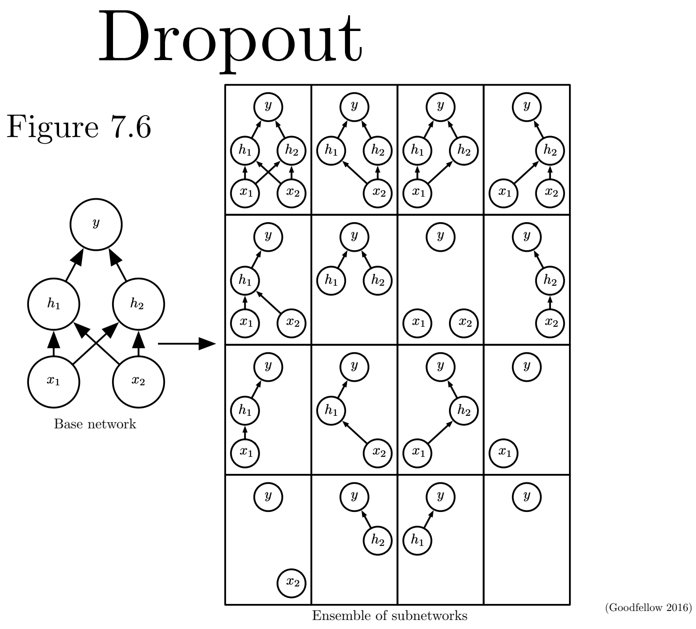
Bagging uses votes of several models: k uncorrelated models will tend to have $1/k$-th generalization error when bagged. Training data can be resampled to create more uncorrelated models.

Dropout (Hinton et al 2012, Wang et al 2013, Srivastava 2014, Gal et al 2015). Dropout is a cheap, practical approximation to averaging exponentially many models (exponential in the number of dropout layers). Dropout at a certain hidden or input layer wp p drops an activation by setting it to zero during training and multiplying all activations (or the weights) by p with no drops during testing (the weight scaling inference rule); theoretically this does not guarantee a proper output activation averaging, only individual hidden unit output expectation is preserved, but this works well in practice (Fig. 3). Weight scaling actually corresponds to taking a geometric means of the votes, and has been showed to work better than Monte Carlo sub-network averaging (which is intractable). First, this prevents feature co-adaptation: since a given activation might vanish at any time during training, the net is forced to learn those activations at each layer which independently contribute to the loss minimization task. Note that this conflicts with locality-focused tasks which rely on specific co-adaptations, like convolutional layers, but might be used at higher levels in CNNs to force the networks to identify more than one underlying feature for their classification (though lower levels can be locally co-adapted). Naive Bayes is an extreme form of dropout which demonstrates the co-adaptation avoidance.

Dropconnect ()Li et al 2013). Generalizes dropout by applying it to edges rather than hidden units. It's not as commonly used as dropout, but it has a more mathematically sound explanation of how its inference procedure mimics ensembles.

7 Adversarial Training

(Goodfellow 2014). Tiny changes in input can result in large differences in output, showing that the learned manifold for a classification category is too sensitive to perturbations in directions that it shouldn't be. Adversarial training forces nets to overcome this.



Dropout

Figure 7.6

(Goodfellow 2016)

Figure 3: From Dr. Goodfellow's DL book lecture, Chapter 7.