Chapter 7: Regularization (Part 1)

Qinyuan Ye
`qinyuany@usc.edu`

Introduction

We want to make an algorithm **perform well** on the training data, and also **on new inputs**. The collection of strategies addressing this issue is regularization – reduce the test error, possibly at the expense of increased training error.

Some common strategies include:

- Add restrictions on parameter values (hard constraint)
- Add extra terms in objective function (soft constraint)
  - Encode specific patterns
  - Express preference for a simpler model
  - Make under-determined problem determined
- Ensemble methods

Controlling the complexity of a model is not by controlling its size or number of parameters. Rather, empirically we found large models that have been regularized properly are the best-fitting models.

1 Parameter Norm Penalties

Limit the capacity of models by adding a parameter norm penalty $\Omega(\boldsymbol{\theta})$. The regularized objective function becomes:

$$\tilde{J}(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) = J(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) + \alpha\Omega(\boldsymbol{\theta}) \tag{1}$$

where $\tilde{J}(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})$ is regularized objective, and $J(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})$ is original objective. Additionally, $\hat{J}(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})$ means a quadratic approximation of $J(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})$ in the following parts. These notations are used throughout this handout.

- Choice of Ω indicates preference of solution;
- $\alpha \in [0, \infty)$ determines the contribution of penalty term; Sometimes desirable to use different α for different layers, but this increase hyper-parameter search space, so usually we use the same α throughout the model;
- In affine transformations, we usually only regularize weight parameters ($\boldsymbol{w}$, how two variables interact), but leave bias terms ($\boldsymbol{b}$) un-regularized (require less data than weights to fit accurately); in practice, regularizing $\boldsymbol{b}$ may lead to under-fitting.

1.1 L^2 Parameter Regularization

Method. Add a regularization term $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{w}||_2$ to the objective function, so that

$$\tilde{J}(\boldsymbol{w}, \mathbf{X}, \mathbf{y}) = \frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w} + J(\boldsymbol{w}, \mathbf{X}, \mathbf{y}) \tag{2}$$

Alternative Names. Weight decay, ridge regression, and Tikhonov regularization.

Studying the gradient. In one single step (**micro perspective**), if we take the gradient of Eq. (2),

$$\nabla_{\boldsymbol{w}} \tilde{J}(\boldsymbol{w}, \mathbf{X}, \mathbf{y}) = \alpha \boldsymbol{w} + \nabla_{\boldsymbol{w}} J(\boldsymbol{w}, \mathbf{X}, \mathbf{y}) \tag{3}$$

Denoting ϵ as learning rate in stochastic gradient descent (SGD), the update to weight $\boldsymbol{w}$ is

$$\boldsymbol{w} \leftarrow (1 - \epsilon\alpha)\boldsymbol{w} - \epsilon \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) \tag{4}$$

Therefore, in one single step, the weight vector shrinks by a constant factor, compared to standard SGD. What happens if we look from a **macro perspective** (i.e. how L^2 regularization takes effect in the whole training process)? Denote $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} J(\boldsymbol{w})$ and $\tilde{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \tilde{J}(\boldsymbol{w})$. We have $\hat{J}(\boldsymbol{w})$, the quadratic approximation of $J(\boldsymbol{w})$ with Taylor series:

$$\hat{J}(\boldsymbol{w}) \approx J(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^T \boldsymbol{H} (\boldsymbol{w} - \boldsymbol{w}^*) \tag{5}$$

where $\boldsymbol{H}$ is the Hessian matrix of J with respect to $\boldsymbol{w}$ evaluated at $\boldsymbol{w}^*$. Because $\boldsymbol{w}^*$ is the optimum for J, there is no first-order term in this Taylor series; also, $\boldsymbol{H}$ is positive semi-definite for this reason, and can be decomposed to $\boldsymbol{H} = \boldsymbol{Q}\Lambda\boldsymbol{Q}^T$.

To solve $\tilde{\boldsymbol{w}}$, the solution for regularized objective $\tilde{J}$, $\nabla_{\boldsymbol{w}} \tilde{J}$ must be zero,

$$\nabla_{\boldsymbol{w}} \tilde{J} = \nabla_{\boldsymbol{w}} \hat{J} + \nabla_{\boldsymbol{w}} (\frac{\alpha}{2} \boldsymbol{w}^T \boldsymbol{w}) \tag{6}$$

$$\boldsymbol{H}(\tilde{\boldsymbol{w}} - \boldsymbol{w}^*) + \alpha \tilde{\boldsymbol{w}} = 0 \tag{7}$$

$$\tilde{\boldsymbol{w}} = (\boldsymbol{H} + \alpha \boldsymbol{I})^{-1} \boldsymbol{H} \boldsymbol{w}^* = \boldsymbol{Q}(\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1} \boldsymbol{\Lambda} \boldsymbol{Q}^T \boldsymbol{w}^* \tag{8}$$

Therefore, from the macro perspective, L^2 regularization is re-scaling $\boldsymbol{w}^*$ along the axes defined by the eigen-vectors of $\boldsymbol{H}$. Specifically, the component of $\boldsymbol{w}^*$ that is aligned with the i-th eigen-vector of $\boldsymbol{H}$ is re-scaled by a factor of $\frac{\lambda_i}{\lambda_i + \alpha}$. When $\lambda_i \gg \alpha$, effect of regularization is small. When $\lambda_i \ll \alpha$, component will shrink to nearly zero.

Example of Linear Regression. Conclusions of this example may be used in later parts.

	w/o regularization	w/ regularization
objective function	$(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$	$(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) + \frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w}$
solution	$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$	$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X} + \alpha\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$

The covariance matrix of $\boldsymbol{X}$ is $\frac{1}{m}\boldsymbol{X}^T\boldsymbol{X}$. L^2 regularization replaces $\boldsymbol{X}^T\boldsymbol{X}$ with $\boldsymbol{X}^T\boldsymbol{X} + \alpha\boldsymbol{I}$ in the solution, which can be interpreted as the models "perceive" the input $\boldsymbol{X}$ as having higher variance.

1.2 L^1 Parameter Regularization

Method. Add a regularization term $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1 = \sum_i |w_i|$ to the objective function.

Studying the gradient. Corresponding gradient,

$$\nabla_{\boldsymbol{w}} \tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \alpha \text{sign}(\boldsymbol{w}) + \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) \tag{9}$$

Suppose (1) we're fitting a linear regression model, with quadratic loss, (2) the Hessian matrix of objective function is diagonal, and (3) the original optimal without regularization is $\boldsymbol{w}^*$. In this case, the analytical solution for w_i becomes

$$w_i = \text{sign}(w_i^*) \max \left\{ |w_i^*| - \frac{\alpha}{H_{i,i}} \right\} \tag{10}$$

That is, where $w_i^* > 0$, if it is small ($< \frac{\alpha}{H_{i,i}}$), it will be regularized to zero; otherwise w_i is moved closer to zero by $\frac{\alpha}{H_{i,i}}$.

Compare with L^2 regularization. L^1 regularization results in a more **sparse** solution. This property enables L^1 regularization to be used for feature selection (e.g. LASSO).

2 Norm Penalties as Constrained Optimization

Apart from adding penalty term $\Omega(\boldsymbol{\theta})$ to objective function J and try to minimize their sum $\tilde{J}$, we can also make sure it is small by optimizing J with a constraint $\Omega(\boldsymbol{\theta}) < k$. This can be done by constructing a generalized Lagrange function, consisting of original objective function plus a set of penalties:

$$\mathcal{L}(\boldsymbol{\theta}, \alpha; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha(\Omega(\boldsymbol{\theta}) - k) \tag{11}$$

The solution will be:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \max_{\alpha, \alpha \geq 0} \mathcal{L}(\boldsymbol{\theta}, \alpha) \tag{12}$$

Note that both $\boldsymbol{\theta}$ and α are variables in this objective function. When α^* is fixed (say, we already know the best α), the optimization problem becomes

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \mathcal{L}(\boldsymbol{\theta}, \alpha^*) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha^* \Omega(\boldsymbol{\theta}) \tag{13}$$

which is the same as regularization with parameter norm penalty. For example, if Ω is L^2 norm, we can think of it as limiting weights to be in a L^2 ball.

Benefits of regularization as constraints:

- We can specify a concrete constraint region, while the effect of adjusting α for $\Omega(\boldsymbol{\theta})$ is vague. We can take a step with stochastic gradient descent, and re-project $\boldsymbol{\theta}$ back to the feasible region $\Omega(\boldsymbol{\theta}) < k$.
- We can avoid getting stuck in local minima, which is common with regularization term in objective function. These constraints only take effect when the weights attempt to leave the constraint region.
- More stable optimization procedure. A large learning rate may result in positive feedback loop. Explicit constraints with re-projection prevent this.

3 Regularization and Under-Constrained Problems

In a linear regression problem, when the number of instances is smaller than the number of variables, the problem is under-constrained, and its closed form solution $\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ can not be calculated. In a logistic regression problem, when the two classes are linearly separable with vector $\boldsymbol{w}$, $2\boldsymbol{w}$ will also be a feasible solution. An iterative optimization algorithm may keep increasing the magnitude of $\boldsymbol{w}$ and never stops.

However, when we add a regularization term to loss function, convergence is guaranteed. For example, $\boldsymbol{w}$ will not be updated to $2\boldsymbol{w}$ because the likelihood loss is not decreased, while the regularization term is huge.

The idea of using regularization to solve under-determined problems extends beyond machine learning. In linear algebra, we have Moore-Penrose pseudo-inverse of matrix $\boldsymbol{X}$, which is $\boldsymbol{X}^+ = \lim_{\alpha \to 0}(\boldsymbol{X}^T\boldsymbol{X} + \alpha\boldsymbol{I})^{-1}\boldsymbol{X}^T$. This is exactly the same as performing linear regression with weight decay. We can interpret the pesudo-inverse as stabilizing under-determined problems using regularization.

4 Dataset Augmentation

Create fake data and add it to the training set. Easy for classification task, which takes in a high-dimensional input $\boldsymbol{x}$ and summarize it with a single category identity y. We can generate new $(\boldsymbol{x}, y)$ pairs by transforming the $\boldsymbol{x}$ inputs in the training set.

Difficult for other tasks such as density estimation, because to generate new fake data we have to first know the density.

Very effective for a specific classification problem: object recognition. Moving a few pixels in each direction, rotating and scaling are effective operations. Still, we need to make sure we don't do rotation or mirroring for classifying 'b' and 'd' or '6' and '9'.

Injecting noise can also be considered as a form of augmentation. This can be either done with input, or in hidden layers. This can be applied to supervised learning, as well as unsupervised learning (auto-encoders). Dropout, can also be consider as multiplying by noise.

For fair comparison between models, data augmentation should be taken into consideration. Model A (with augmentation) outperforming Model B (without augmentation) does not necessarily mean Model A is better. However, such judgements are subjective. Usually, operations generally applicable to are considered part of the algorithm (e.g. adding Gaussian noise to input), while operations specific to one domain is considered preprocessing (e.g. cropping an image).

5 Noise Robustness

Adding noise to model weights. usually used in recurrent neural networks. Can be interpreted as a stochastic implementation of Bayesian inference over the weights. The Bayesian treatment of learning would consider model weights to be uncertain. The weights should be represented with a distribution. Adding noise to weights reflects such uncertainty.

Noise applied to weights are also encouraging the stability of the model. Consider a regression problem, the objective function is

$$J = \mathbb{E}_{p(\boldsymbol{x},y)} \left(\hat{y}(\boldsymbol{x}) - y\right)^2$$

We now add a random perturbation $\epsilon_{\boldsymbol{W}} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \eta \boldsymbol{I})$ to network weights. The perturbed model becomes $\hat{y}_{\boldsymbol{\epsilon}_W}(\boldsymbol{x})$. The objective function becomes

$$\tilde{J}_{\boldsymbol{W}} = \mathbb{E}_{p(\boldsymbol{x},y,\boldsymbol{\epsilon}_W)} \left[(\hat{y}_W(\boldsymbol{x}) - y)^2\right] = \mathbb{E}_{p(\boldsymbol{x},y,\boldsymbol{\epsilon}_W)} \left[\hat{y}^2_{\boldsymbol{\epsilon}_W}(\boldsymbol{x}) - 2y\hat{y}_{\boldsymbol{\epsilon}_W}(\boldsymbol{x}) + y^2\right] \tag{15}$$

With a small η, minimizing $\tilde{J}$ is equivalent to minimizing J with additional regularization term $\eta \mathbb{E}_{p(\boldsymbol{x},y)} \left[||\nabla_{\boldsymbol{W}} \hat{y}(\boldsymbol{x})||\right]$. This term pushes the model into regions where model is relatively insensitive to small variation in the weights (minima surrounded by flat region).

Injecting noise at output stage. If the label y in dataset is wrong, maximizing $\log p(y|\boldsymbol{x})$ becomes harmful. To deal with this, we explicitly model the noise on the labels. For constant $\epsilon \ll 1$, label y is correct with $1-\epsilon$ probability. Label smoothing tries to soften the target of $[0, 1, 0, 0]$ to $[\frac{\epsilon}{3}, 1-\epsilon, \frac{\epsilon}{3}, \frac{\epsilon}{3}]$.

The output of softmax could never be 0 or 1 exactly, so model may never 'converge' and the weights are still getting larger. Weight decay may solve this issue, but at the cost of encouraging wrong classification. Label smoothing prevent the pursuit of fitting hard probabilities, while still encourage correct classification.

6 Semi-supervised Learning

In the paradigm of semi-supervised learning, both unlabeled examples from $P(\boldsymbol{x})$ and labeled examples from $P(\boldsymbol{x}, y)$ are used to estimate $P(y|\boldsymbol{x})$ or predict y from $\boldsymbol{x}$.

One way of semi-supervised learning is combining unsupervised and supervised components in one model. Unsupervised learning first learns how to group examples in representation space by learning a representation $\boldsymbol{h} = f(\boldsymbol{x})$. A long-standing approach is to apply principal components analysis for creating representation. Then a classifier in representation space is adopted. Usually a linear classifier achieves better generalization.

Another way of semi-supervised learning is to construct a generative model of either $P(\boldsymbol{x})$ or $P(\boldsymbol{x}, y)$, and a discriminative model of $P(y|\boldsymbol{x})$. The two models share parameters. One can trade-off the supervised criterion ($-\log P(y|\boldsymbol{x})$ with unsupervised (or generative) one ($-\log P(\boldsymbol{x})$ or $-\log P(\boldsymbol{x}, y)$. The generative criterion expresses a particular form of prior belief about the solution to the supervised learning problem.

7 Multitask Learning

When part of a model is shared across tasks, this part is more constrained toward good values, often yielding better generalization. [maybe a figure here.]

Under certain assumptions, the generalization error bound is improved because of shared parameters. The prior belief of multi-task learning is that, among the factors that explain the variations observed in the data associated with the different tasks, some are shared across two or more tasks.

8 Early Stopping

When training models with over-sufficient representational ability, it begins to overfit - training error decrease steadily over time, but validation error begins to rise. A effective and simple way to deal with this is to store the model when validation error improves, and terminate training when validation error is not improved for a pre-defined number of iterations.

One way to think of early stopping is to consider number of training steps as a hyperparameter tuned on validation set. This hyperparameter is special in that we don't have to try out different values by re-training the model. The only cost is running the validation set periodically during training. Another cost is the memory used to save the best set of parameters so far. This is negligible because the parameters can be slowly written into disks.

Early stopping is unobstrusive, as it makes almost no change in training procedure, the objective function, or the feasible region for parameters, and thus will not damage learning dynamics. This is in contrast with weight decay, where a bad local optimum with extremely small weights is chosen if not used carefully.

Early stopping requires a validation set, which means some training data was not included into the training process. To utilize data in validation set, there are two strategies. (1) Re-train the whole model with the same number of steps, or same number of passes on the whole dataset; (2) Keep the parameters in the previous training process, and continue training with the whole dataset. Stop when the loss per instance on validation set falls below that of original training set at the termination of previous training.

How early stopping regularize the model. Given initial parameter value $\boldsymbol{\theta}_0$, we take τ optimization steps with learning rate ϵ, and bounded gradients. The reachable destination $\boldsymbol{\theta}_\tau$ is limited. This is a straight forward way of understanding early stopping as regularization.

Actually, in the case of linear models, early stopping is equivalent to L^2 regularization.

Quadratic approximation of objective function $J(\boldsymbol{\theta})$ near empirically optimial value of $\boldsymbol{w}^*$

$$\hat{J}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^T \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*) \tag{16}$$

where $\boldsymbol{H}$ is the Hessian matrix of J with respect to $\boldsymbol{w}$ evaluated at $\boldsymbol{w}*$. Given $\boldsymbol{w}^*$ is the minimum of $J(\boldsymbol{w})$, $\boldsymbol{H}$ is positive semidefinite and can be decomposed to $\boldsymbol{H} = \boldsymbol{Q}\Lambda Q^T$, where $\boldsymbol{\Lambda}$ is a diagonal matrix and $\boldsymbol{Q}$ is an orthonormal basis of eiginvectors.

In each optimization step, we have

$$\boldsymbol{w}^{(\tau)} = \boldsymbol{w}^{(\tau-1)} - \epsilon \nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}^{(\tau-1)}) = \boldsymbol{w}^{(\tau-1)} - \epsilon \boldsymbol{H}(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^*) \tag{17}$$

$$\boldsymbol{w}^{(\tau)} - \boldsymbol{w}^* = (\boldsymbol{I} - \epsilon \boldsymbol{H})(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^*) \tag{18}$$

Replacing $\boldsymbol{H}$ with $\boldsymbol{Q}\Lambda Q^T$ and changing the representation, results in

$$\boldsymbol{Q}^T(\boldsymbol{H}^{(\tau)} - \boldsymbol{H}^*) = (\boldsymbol{I} - \epsilon\Lambda)\boldsymbol{Q}^T(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^*) \tag{19}$$

Assuming $\boldsymbol{w}^{(0)} = 0$ and ϵ is small enough for $|1 - \epsilon\lambda_i| < 1$, we have

$$\boldsymbol{Q}^T \boldsymbol{w}^{(\tau)} = [\boldsymbol{I} - (\boldsymbol{I} - \epsilon\Lambda)^\tau]\, \boldsymbol{Q}^T \boldsymbol{w}^* \tag{20}$$

Recall that in L^2 regularization

$$\boldsymbol{Q}^T \tilde{\boldsymbol{w}} = \left[\boldsymbol{I} - (\Lambda + \alpha \boldsymbol{I})^{-1} \alpha\right] \boldsymbol{Q}^T \boldsymbol{w}^* \tag{21}$$

As long as $(\boldsymbol{I} - \epsilon \boldsymbol{\Lambda}^\tau) = (\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1} \alpha$, the two regularization methods are equivalent in terms of the resulting minimum. A trajectory of length τ ends at a point that corresponds to a minimum of the L^2-regularized objective. However, early stopping does more than restricting the trajectory length; instead, it monitors validation error in order to stop at the a particularly good point. Early stopping therefore has the advantage over weight decay in that it automatically determines the correct amount of regularization, while with weight decay a hyper-parameter needs to be chosen.