Regularization

• Many strategies used in machine learning are explicitly designed to reduce the test error, possibly at the expense of increased training error. These strategies are known collectively as regularization.

• In section 5.2.2, we defined regularization as “any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.”

• We almost always find that the best fitting model is a large model that has been regularized appropriately.

• Many regularization approaches are based on limiting the capacity of models by adding a parameter norm penalty Ω(θ) to the objective function J.

• We denote the regularized objective function by J~:

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha\Omega(\boldsymbol{\theta}), \qquad (7.1)$$

where α ∈ [0, ∞) is a hyperparameter that weights the relative contribution of the norm penalty term, Ω, relative to the standard objective function J.

• Setting α to 0 results in no regularization. Larger values of α correspond to more regularization.

Parameter norm penalties

• We typically choose to use a parameter norm penalty Ω that penalizes only the weights of the affine transformation at each layer and leaves the biases unregularized.

• Each weight specifies how two variables interact. Fitting the weight well requires observing both variables in a variety of conditions.

• Each bias controls only a single variable. This means that we do not induce too much variance by leaving the biases unregularized. The biases typically require less data than the weights to fit accurately. Also, regularizing the bias parameters can introduce a significant amount of underfitting.

• In the context of neural networks, it is sometimes desirable to use a separate penalty with a different α coefficient for each layer of the network.

• Because it can be expensive to search for the correct value of multiple hyperparameters, it is still reasonable to use the same weight decay at all layers just to reduce the size of search space.

L2 parameter regularization

• The $L2$ parameter norm penalty is commonly known as weight decay. This regularization strategy drives the weights closer to the origin by adding a regularization term Ω(θ) = ½ ||$\boldsymbol{\omega}$|| 22 to the objective function.

• In other academic communities, $L2$ regularization is also known as ridge regression or Tikhonov regularization.

• We can gain some insight into the behavior of weight decay regularization by studying the gradient of the regularized objective function.

• We assume no bias parameter, so θ is just $\boldsymbol{\omega}$. Such a model has the following total objective function:

$$\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \frac{\alpha}{2}\boldsymbol{w}^\top \boldsymbol{w} + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}), \qquad (7.2)$$

with the corresponding parameter gradient

$$\nabla_{\boldsymbol{w}} \tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \alpha \boldsymbol{w} + \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}). \tag{7.3}$$

• To take a single gradient step to update the weights, we perform this update:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \epsilon \left(\alpha \boldsymbol{w} + \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) \right). \tag{7.4}$$

Written another way, the update is

$$\boldsymbol{w} \leftarrow (1 - \epsilon\alpha)\boldsymbol{w} - \epsilon \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}). \tag{7.5}$$

• The addition of the weight decay term has modified the learning rule to multiplicatively shrink the weight vector by a constant factor on each step before gradient update.
• This describes what happens in a single step. But what happens over the entire course of training?
• We further simplify the analysis by making a quadratic approximation to the objective function in the neighborhood of the value of the weights that obtains minimal unregularized training cost, $\boldsymbol{\omega}* = \arg\min_{\boldsymbol{\omega}} J(\boldsymbol{\omega})$.
• If the objective function is truly quadratic, as in the case of fitting a linear regression model with mean squared error, then the approximation is perfect.
• The approximation J^ is given by

$$\hat{J}(\boldsymbol{\theta}) = J(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^\top \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*), \tag{7.6}$$

where $\boldsymbol{H}$ is the Hessian matrix of J with respect to $\boldsymbol{\omega}$ evaluated at $\boldsymbol{\omega}*$.
• There is no first-order term in this quadratic approximation, because $\boldsymbol{\omega}*$ is defined to be a minimum, where the gradient vanishes. Likewise, because $\boldsymbol{\omega}*$ is the location of a minimum of J, we can conclude that $\boldsymbol{H}$ is positive semidefinite.
• The minimum of J^ occurs where its gradient

$$\nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*) \tag{7.7}$$

is equal to 0.
• To study the effect of weight decay, we modify equation 7.7 by adding the weight decay gradient. We can now solve for the minimum of the regularized version of J^. We use the variable $\boldsymbol{\omega}$~ to represent the location of the minimum.

$$\alpha \tilde{\boldsymbol{w}} + \boldsymbol{H}(\tilde{\boldsymbol{w}} - \boldsymbol{w}^*) = 0 \tag{7.8}$$

$$(\boldsymbol{H} + \alpha \boldsymbol{I})\tilde{\boldsymbol{w}} = \boldsymbol{H}\boldsymbol{w}^* \tag{7.9}$$

$$\tilde{\boldsymbol{w}} = (\boldsymbol{H} + \alpha \boldsymbol{I})^{-1} \boldsymbol{H}\boldsymbol{w}^* \tag{7.10}$$

• As α approaches 0, the regularized solution $\boldsymbol{\omega}$~ approaches $\boldsymbol{\omega}*$. But what happens as α grows?
• Because $\boldsymbol{H}$ is real and symmetric, we can decompose it into a diagonal matrix $\boldsymbol{\Lambda}$ and an orthonormal basis of eigenvectors $\boldsymbol{Q}$, such that $\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$.
• Applying the decomposition to equation 7.10, we obtain

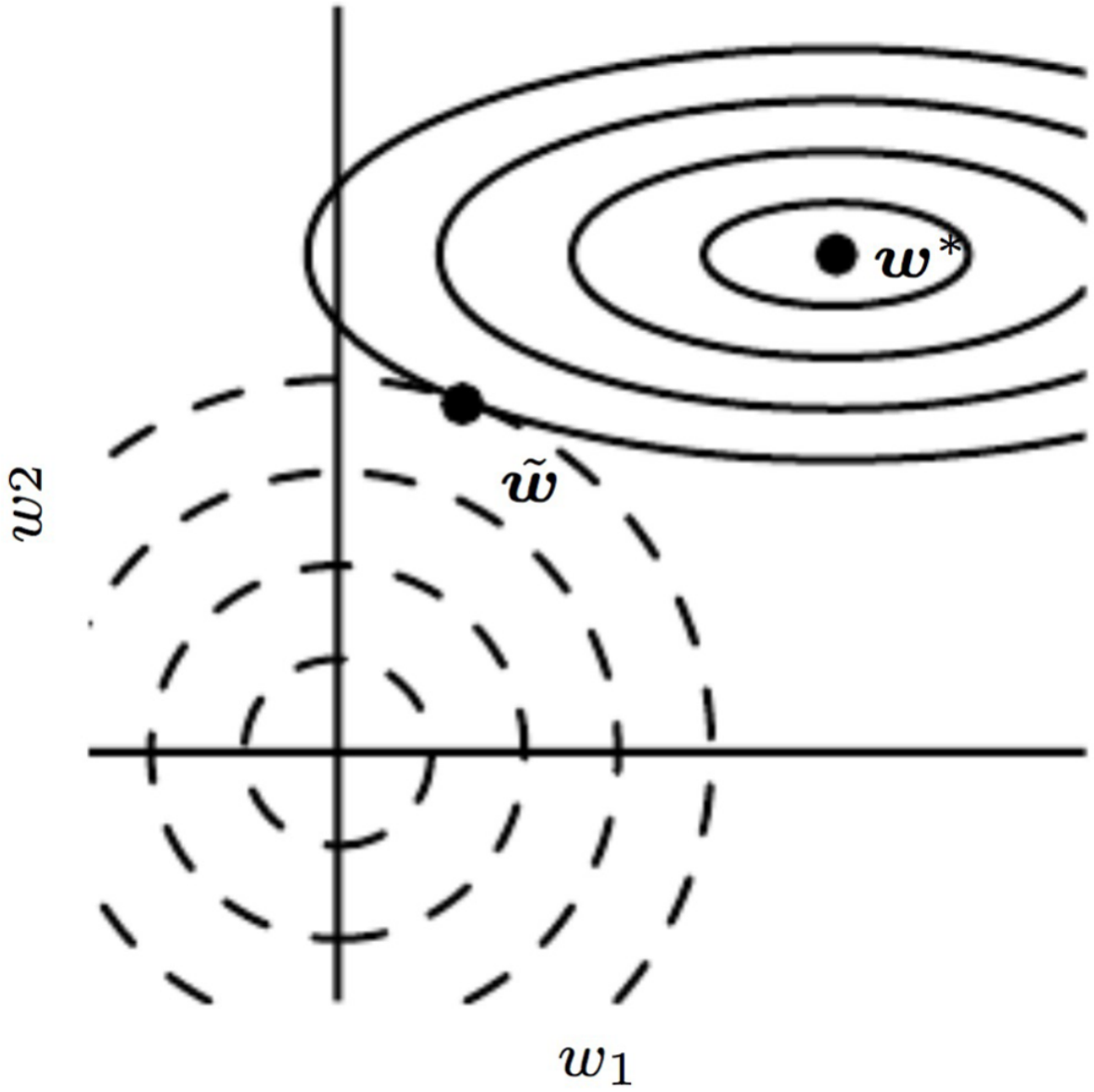
$$\tilde{\boldsymbol{w}} = (\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top + \alpha \boldsymbol{I})^{-1}\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top \boldsymbol{w}^* \tag{7.11}$$

$$= \left[\boldsymbol{Q}(\boldsymbol{\Lambda} + \alpha \boldsymbol{I})\boldsymbol{Q}^\top\right]^{-1}\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top \boldsymbol{w}^* \tag{7.12}$$

$$= \boldsymbol{Q}(\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1}\boldsymbol{\Lambda}\boldsymbol{Q}^\top \boldsymbol{w}^*. \tag{7.13}$$

• We see that the effect of weight decay is to rescale $\boldsymbol{\omega}*$ along the axes defined by the eigenvectors of $\boldsymbol{H}$. Specifically, the component of $\boldsymbol{\omega}*$ that is aligned with the i-th eigenvector of $\boldsymbol{H}$ is rescaled by a factor of λi/(λi+α).
• Along the directions where the eigenvalues of H are relatively large, for example, where λi $\gg \alpha$, the effect of regularization is relatively small. Yet components with λi $\ll \alpha$ will be shrunk to have nearly zero magnitude.
• Only directions along which the parameters contribute significantly to reducing the objective function are preserved relatively intact. In directions that do not contribute to reducing the objective function, a small eigenvalue of the Hessian tells us that movement in this direction will not significantly increase the gradient.
• Components of the weight vector corresponding to such unimportant directions are decayed away through the use of the regularization throughout training.

——



• The solid ellipses represent contours of equal value of the unregularized objective. The dotted circles represent contours of equal value of the $L2$ regularizer.
• At the point ω~, 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 $\omega*$. 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 ω1 close to zero.
• In the second dimension, the objective function is very sensitive to movements away from $\omega*$. The corresponding eigenvalue is large, indicating high curvature. As a result, weight decay affects the position of ω2 relatively little.
• For linear regression, the cost function is the sum of squared errors:

$$(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^\top (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}). \tag{7.14}$$

When we add $L!$ regularization, the objective function changes to

$$(\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y})^{\top}(\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y})+\frac{1}{2}\alpha\boldsymbol{w}^{\top}\boldsymbol{w}. \tag{7.15}$$

• This changes the normal equations for the solution from

$$\boldsymbol{w}=(\boldsymbol{X}^{\top}\boldsymbol{X})^{-1}\boldsymbol{X}^{\top}\boldsymbol{y} \tag{7.16}$$

to

$$\boldsymbol{w}=(\boldsymbol{X}^{\top}\boldsymbol{X}+\alpha\boldsymbol{I})^{-1}\boldsymbol{X}^{\top}\boldsymbol{y}. \tag{7.17}$$

• The matrix $\boldsymbol{X}$T$\boldsymbol{X}$ in equation 7.16 is proportional to the covariance matrix . 1/m $\boldsymbol{X}$T$\boldsymbol{X}$. Using $L!$ regularization
replaces this matrix with ($\boldsymbol{X}$T$\boldsymbol{X}$ + $\alpha\boldsymbol{I}$)^-1 in equation 7.17.
• The new matrix is the same as the original one, but with the addition of α to the diagonal. The diagonal entries of this matrix correspond to the variance of each input feature.
• $L2$ regularization causes the learning algorithm to “perceive” the input $\boldsymbol{X}$ as having higher variance, which makes it shrink the weights on features whose covariance with the output target is low compared to this added variance.

L1 regularization
• While $L2$ weight decay is the most common form of weight decay, another option is to use $L1$ regularization.
• Formally, $L1$ regularization on the model parameter $\boldsymbol{\omega}$ is

$$\Omega(\boldsymbol{\theta})=||\boldsymbol{w}||_1=\sum_i|w_i|, \tag{7.18}$$

that is, as the sum of absolute values of the individual
parameters.
• As with $L2$ weight decay, $L1$ weight decay controls the strength of the regularization by scaling the penalty Ω using a positive hyperparameter α.
• Thus, the regularized objective function $J(\boldsymbol{\omega}; \boldsymbol{X}, \boldsymbol{y})$ is

$$\tilde{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})=\alpha||\boldsymbol{w}||_1+J(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y}), \tag{7.19}$$

with the corresponding gradient (actually, sub gradient)

$$\nabla_{\boldsymbol{w}}\tilde{J}(\boldsymbol{w};\boldsymbol{X},\boldsymbol{y})=\alpha\text{sign}(\boldsymbol{w})+\nabla_{\boldsymbol{w}}J(\boldsymbol{X},\boldsymbol{y};\boldsymbol{w}), \tag{7.20}$$

where sign($\boldsymbol{\omega}$) is the sign of $\boldsymbol{\omega}$ applied element-wise.
• By inspecting equation 7.20, we can see that the regularization contribution to the gradient no longer scales linearly with each ωi; instead it is a constant factor with a sign equal to sign(ωi).

• One consequence of this form of the gradient is that we will not necessarily see clean algebraic solutions to quadratic approximations of $J(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\omega})$ as we did for $L2$ regularization.
• Our simple linear model has a quadratic cost function that we can represent via its Taylor series. Alternately, we could imagine that this is a truncated Taylor series approximating the cost function of a more sophisticated model.
• The gradient in this setting is given by

$$\nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*), \tag{7.21}$$

where, again, $\boldsymbol{H}$ is the Hessian matrix of J with respect to $\boldsymbol{\omega}$ evaluated at $\boldsymbol{\omega}*$.
• Because the $L1$ penalty does not admit clean algebraic expressions in the case of a fully general Hessian, we will further assume that the Hessian is diagonal, $\boldsymbol{H}$ = diag([H1,1, ... , Hn,n]), where each Hi,i > 0.
• This assumption holds if the data for the linear regression problem has been preprocessed to remove all correlation between the input features, which may be accomplished using PCA.
• Our quadratic approximation of the L" regularized objective function decomposes into a sum over the parameters:

$$\hat{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{w}^*; \boldsymbol{X}, \boldsymbol{y}) + \sum_i \left[\frac{1}{2} H_{i,i} (\boldsymbol{w}_i - \boldsymbol{w}_i^*)^2 + \alpha |w_i| \right]. \tag{7.22}$$

• The problem of minimizing this approximate cost function has an analytical solution (for each dimension i), with the following form:

$$w_i = \text{sign}(w_i^*) \max \left\{ |w_i^*| - \frac{\alpha}{H_{i,i}}, 0 \right\}. \tag{7.23}$$

• Consider the situation where ωi∗ > 0 for all i. There are two possible outcomes:
    1. The case where ωi∗ ≤ α/Hi,i . Here the optimal value of ωi under the regularized objective is simply ωi = 0. This occurs because the contribution of $J(\boldsymbol{\omega}; \boldsymbol{X}, \boldsymbol{y})$ to the regularized objective J~($\boldsymbol{\omega}$; X, y) is overwhelmed—in direction i—by the $L1$ regularization, which pushes the value of ωi to zero.
    2. The case where ωi∗ > α/Hi,i. In this case, the regularization does not move the optimal value of ωi to zero but instead just shifts it in that direction by a distance equal to α/Hi,i .
• A similar process happens when ωi < 0, but with the L1 penalty making ωi less negative by α/Hi,i , or 0.
• In comparison to $L2$ regularization, $L1$ regularization results in a solution that is more sparse. Sparsity in this context refers to the fact that some parameters have an optimal value of zero.
• The sparsity property induced by $L1$ regularization has been used extensively as a feature selection mechanism (choosing which subset of the available features should be used).
• The well known LASSO (least absolute shrinkage and selection operator) model integrates an $L1$ penalty with a linear model and a least-squares cost function. The $L1$ penalty causes a subset of the weights to become zero, suggesting that the corresponding features may safely be discarded.

Dataset augmentation

• The best way to make a machine learning model generalize better is to train it on more data. In practice, the amount of data we have is limited. One way to solve this problem is to create fake data and add it to the training set.

• This approach is easiest for classification. The main task facing a classifier is to be invariant to a wide variety of transformations. We can generate new ($\boldsymbol{x}$, y) pairs easily just by transforming the $\boldsymbol{x}$ inputs in our training set.

• This approach is not as readily applicable to many other tasks. For example, it is difficult to generate new fake data for a density estimation task unless we have already solved the density estimation problem.

• Dataset augmentation is particularly effective for a specific classification problem: object recognition.

• Operations like translating the training images a few pixels in each direction can often greatly improve generalization, even if the model has already been designed to be partially translation invariant. Many other operations, such as rotating the image or scaling the image, have also proved quite effective.

• One must be careful not to apply transformations that would change the correct class. For example, optical character recognition tasks require recognizing the difference between “b” and “d” and the difference between “6” and “9,” so horizontal flips and 180◦ rotations are not appropriate.

• Injecting noise in the input to a neural network can also be seen as a form of data augmentation.

• For many classification and even some regression tasks, the task should still be possible to solve even if small random noise is added to the input. One way to improve the robustness of neural networks is simply to train them with random noise applied to their inputs.

• Noise injection also works when the noise is applied to the hidden units, which can be seen as doing dataset augmentation at multiple levels of abstraction.

• Dropout can be seen as a process of constructing new inputs by multiplying by noise.

Noise robustness

• For some models, the addition of noise with infinitesimal variance at the input of the model is equivalent to imposing a penalty on the norm of the weights. In the general case, noise injection can be much more powerful than simply shrinking the parameters, especially when the noise is added to the hidden units.

• Another way that noise has been used is by adding it to the weights. This technique has been used primarily in the context of RNN.

• Assume that with each input presentation we include a random perturbation $\epsilon \boldsymbol{W} \sim N(\boldsymbol{\epsilon}; \mathbf{0}, \eta \boldsymbol{I})$ of the network weights. We use a standard l-layer MLP, and denote the perturbed model as yY ($\boldsymbol{x}$). The objective function is 4$\boldsymbol{W}$

• For small η, the minimization of J with added weight noise (with covariance $\eta \boldsymbol{I}$) is equivalent to minimization of J with an additional regularization term:
$\eta \mathbb{E}5(\boldsymbol{x},8)\ \nabla \boldsymbol{W} y\mathrm{Y}(\boldsymbol{x})!.$