Regularization Review

Akhil Vasvani

April 2019

1 Questions

Exercise 1. What is regularization?

Proof. Regularization is a technique to discourage the complexity of the model. It does this by penalizing the loss function, which helps solve the overfitting problem.

Example. Let's look at a simple Linear Regression to illustrate the use of regularization. Assuming no bias parameter, say $\boldsymbol{Y}$ represents the approximate values from the input $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and their associated weights $\boldsymbol{w} = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n\}$– for n features– such that

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{w} = \sum_{i=1}^{n} \boldsymbol{w}_i \boldsymbol{x}_i.$$

Now, if we compare these approximate values to our actual values we can see how close (or far) we are between the values– this our loss function. In this specific case, we not only take the difference between the two values, but square the difference as well to avoid negative differences and sum all of them– this is called *residual sum of differences* or *RSS*.

$$\begin{aligned}\text{RSS} = (y - \boldsymbol{Y})^2 = \sum_{i=1}^{n} (y_i - \boldsymbol{Y}_i)^2 = \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} \boldsymbol{w}_j \boldsymbol{x}_{ij})^2 \\ = (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^\top (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}).\end{aligned}$$

Initially, the weights are chosen to minimize this loss function but will adjust based on your training data. If there is noise in the training data, then the estimated weights will not generalize well to the future data. See Figure 1. This is where regularization comes in and shrinks or regularizes these learned weights towards zero.

There are three types of regularization used in Machine Learning: L^1 **regularization**, L^2 **regularization**, and a hybrid of L^1 **and** L^2 **regularization**. □

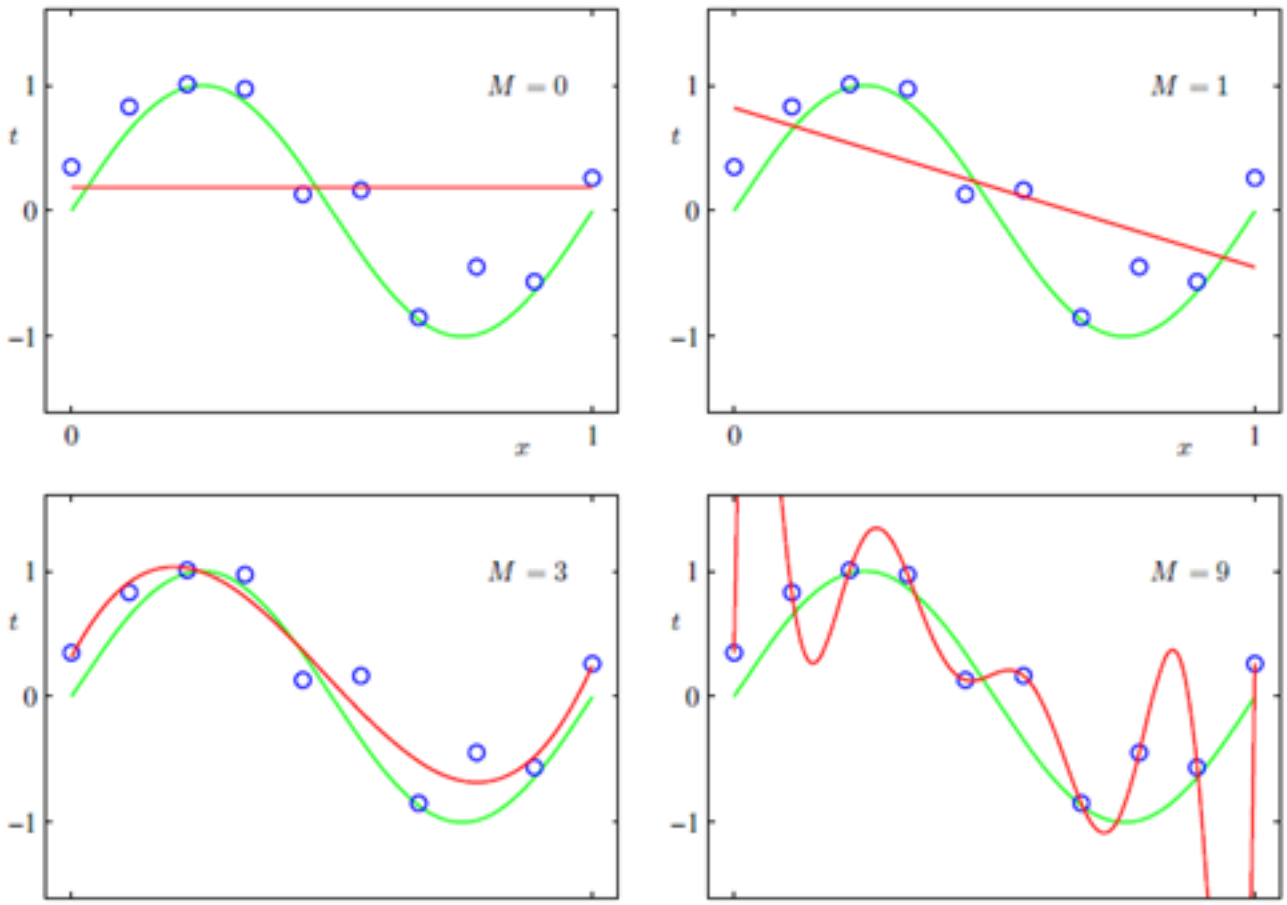

Figure 1: This figure shows the fitting of polynomials of various order on the data points. The function with degree 3 is able to fit our training data to a great extent. So the polynomial of order 9 should also be able to fit the underlying pattern even better? We see that polynomial of degree 9 is able to make accurate predictions for all the datapoints in the training set. But wait, the function obtained for the polynomial of degree 9 looks nothing like our chosen function $sin(2\pi x)$. So what has happened here? This is called overfitting. The polynomial of degree 9 has trained itself to get the correct target values for all the noise induced data points and thus has failed to predict the correct pattern. This function may give zero error for training set but will give huge errors in predicting the correct target values for test dataset. Regularization helps to avoid this problem.

Exercise 2. What is L^2 regularization?

Proof. This regularization strategy drives the weights closer to the origin (or any other specified parameter value) by adding a regularization term $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{w}||_2^2$ to the objective or loss function. L^2 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. To simplify the presentation, we assume no bias parameter. Such a model has the following total objective function:

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

with the corresponding parameter gradient

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

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

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \epsilon \left(\alpha \boldsymbol{w} + \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})\right) = (1 - \epsilon\alpha)\boldsymbol{w} - \epsilon \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}).$$

We can see that 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, just before performing the usual gradient update. This describes what happens in a single step. But what happens over the entire course of training?

We can 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, $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} J(\boldsymbol{w})$. 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 $\hat{J}$ is given by

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

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

The minimum of $\hat{J}$ occurs where its gradient

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

is equal to $\boldsymbol{0}$.

To study the effect of weight decay, we modify the above equation by adding the weight decay gradient. We can now solve for the minimum of the regularized $\hat{J}$. We use the variable $\tilde{\boldsymbol{w}}$ to represent the location of the minimum.

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

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

As α approaches 0, the regularized solution $\tilde{\boldsymbol{w}}$ approaches $\boldsymbol{w}^*$. 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 $\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top$. Applying the decomposition to the above, we obtain:

$$\begin{aligned}\tilde{\boldsymbol{w}} &= \left(\boldsymbol{Q}\Lambda\boldsymbol{Q}^\top + \alpha \boldsymbol{I}\right)^{-1}\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top\boldsymbol{w}^* \\ &= \left[\boldsymbol{Q}(\boldsymbol{\Lambda} + \alpha \boldsymbol{I})\boldsymbol{Q}^\top\right]^{-1}\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top\boldsymbol{w}^* \\ &= \boldsymbol{Q}(\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1}\boldsymbol{\Lambda}\boldsymbol{Q}^\top\boldsymbol{w}^*.\end{aligned}$$

We see that the effect of weight decay is to rescale $\boldsymbol{w}^*$ along the axes defined by the eigenvectors of $\boldsymbol{H}$. Specifically, the component of $\boldsymbol{w}^*$ that is aligned with the i-th eigenvector of $\boldsymbol{H}$ is rescaled by a factor of $\frac{\lambda_i}{\lambda_i + \alpha}$.

Along the directions where the eigenvalues of $\boldsymbol{H}$ are relatively large, for example, where $\lambda_i \gg \alpha$ the effect of regularization is relatively small. However, components with $\lambda_i \ll \alpha$ will be shrunk to have nearly zero magnitude. This effect is illustrated in the Figure 2.

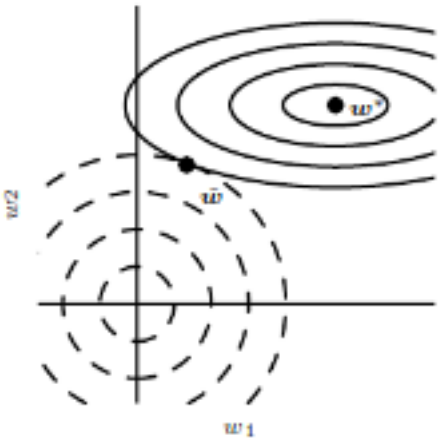

Figure 2: An illustration of the effect of L^2 (or weight decay) regularization on the value of the optimal $\boldsymbol{w}$. The solid ellipses represent contours of equal value of the unregularized objective. The dotted circles represent contours of equal value of the L^2 regularizer. At the point $\tilde{\boldsymbol{w}}$, 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.

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.

Example. For linear regression, the cost function is the sum of squared errors:

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

When we add L^2 regularization, the objective function changes to

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

This changes the normal equations for the solution from

$$\boldsymbol{w} = \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

to

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

The matrix $\boldsymbol{X}^\top \boldsymbol{X}$ in the original normal equation is proportional to the covariance matrix $\frac{1}{m}\boldsymbol{X}^\top \boldsymbol{X}$. Using L^2 regularization replaces this matrix with $(\boldsymbol{X}^\top \boldsymbol{X} + \alpha \boldsymbol{I})^{-1}$. 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. We can see that L^2 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.

TL;DR. The L^2 regularization adds a penalty equal to the sum of the squared value of the coefficients. The L^2 regularization will force the **parameters to be relatively small**, the bigger the penalization, the smaller (and the more robust) the coefficients are.

□

Exercise 3. What is L^1 regularization? Why does L^1 regularization result in sparse models?

Proof. Formally put, L^1 regularization on the model parameter $\boldsymbol{w}$ is defined as:

$$\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1 = \sum_i |w_i|, \tag{1}$$

that is, as the sum of absolute values of the individual parameters.

Note. As with L^2 regularization, we could regularize the parameters towards a value that is not zero, but instead towards some parameter value $\boldsymbol{w}^{(o)}$. In that case the L^1 regularization would introduce the term $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w} - \boldsymbol{w}^{(o)}||_1 = \sum_i |w_i - w_i^{(o)}|$.

Similar to L^2 regularization, the strength of the regularization by scaling the penalty Ω using a positive hyperparameter α. Thus, the regularized objective function $\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$ is given by

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

with the corresponding gradient (actually, sub-gradient):

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

where $\operatorname{sign}(\boldsymbol{w})$ is simply the sign of $\boldsymbol{w}$ applied element-wise.

By inspecting the above equation, we can see immediately that the effect of L^1 regularization is quite different from that of L^2 regularization. Specifically, we can see that the regularization contribution to the gradient no longer scales linearly with each w_i ; instead it is a constant factor with a sign equal to $\operatorname{sign}(w_i)$. One consequence of this form of the gradient is that we will not necessarily see clean algebraic solutions to quadratic approximations of $J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$ as we did for L^2 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

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

where, again, $\boldsymbol{H}$ is the Hessian matrix of J with respect to $\boldsymbol{w}$ evaluated at $\boldsymbol{w}^*$.

Because the L^1 penalty does not admit clean algebraic expressions in the case of a fully general Hessian, we will also make the further simplifying assumption that the Hessian is diagonal, $\boldsymbol{H} = \operatorname{diag}([H_{1,1}, \ldots, H_{n,n}])$, where each $H_{i,i} > 0$. 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^1 regularized objective function decomposes into a sum over the parameters:

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

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

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

Consider the situation where $w_i^* > 0$ for all i. There are two possible outcomes:

1. The case where $w_i^* \leq \frac{\alpha}{H_{i,i}}$. Here the optimal value of w_i under the regularized objective is simply $w_i = 0$. This occurs because the contribution of $J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$ to the regularized objective $J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y})$ is overwhelmed—in the direction i— by the L^1 regularization which pushes the value of w_i to zero.

2. The case where $w_i^* > \frac{\alpha}{H_{i,i}}$. In this case, the regularization does not move the optimal value of w_i to zero but instead it just shifts it in that direction by a distance equal to $\frac{\alpha}{H_{i,i}}$.

A similar process happens when $w_i^* < 0$, but with the L^1 penalty making w_i less negative by $\frac{\alpha}{H_{i,i}}$, or 0.

In comparison to L^2 regularization, L^1 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 of L^1 regularization is a qualitatively different behavior than arises with L^2 regularization. While $\tilde{\boldsymbol{w}} = \boldsymbol{Q}(\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1}\boldsymbol{\Lambda}\boldsymbol{Q}^\top \boldsymbol{w}^*$ gave the solution $\tilde{w}$ for L^2 regularization, if we revisit that equation using the assumption of a diagonal and positive definite Hessian $\boldsymbol{H}$ that we introduced for our analysis of L^1 regularization, we find that $\tilde{w}_i = \frac{H_{i,i}}{H_{i,i}+\alpha} w_i^*$. If w_i^* was nonzero, the $\tilde{w}_i$ remains nonzero. This demonstrates that L^2 regularization does not cause the parameters to become sparse, while L^1 regularization may do so for large enough α.

Note. The sparsity property induced by L^1 regularization has been used extensively as a feature selection mechanism. Feature selection simplifies a machine learning problem by choosing which subset of the available features should be used. In particular, the well known LASSO (Tibshirani , 1995) (least absolute shrinkage and selection operator) model integrates an L^1 penalty with a linear model and a least squares cost function. The L^1 penalty causes a subset of the weights to become zero, suggesting that the corresponding features may safely be discarded.

TL;DR. The L^1 regularization adds a penalty equal to the sum of the absolute value of the coefficients. The L^1 regularization will **shrink some parameters to zero**. Hence some variables will not play any role in the model, L^1 regression can be seen as a way to select features in a model. More on this here and here too.

□

Exercise 4. Compare L^1 and L^2 regularization.

Proof. In a nutshell,

L^2 Regularization	L^1 Regularization
Computationally efficient due to having analytic solutions	Computationally inefficient on non-sparse cases
Non-sparse outputs	Sparse outputs
No feature selection	Built-in feature Selection

Table 1: Differences between L^1 and L^2 Regularization

This geometric interpretation shows how with L^1 regularization one of the weights β_1 in this case will be 0, while with L^2 regularization β_1 will be close to 0.

□

Exercise 5. What is Elastic-net regularization?

Proof. The elastic-net regularization linearly combines the L^1 and L^2 penalties of the lasso and ridge regularization methods. Formally put, the elastic-net regularization on the model parameter $\boldsymbol{w}$ is defined as:

$$\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1 + \frac{1}{2}||\boldsymbol{w}||_2^2, \tag{2}$$

which has the following objective function:

$$\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \frac{\alpha_2}{2}\boldsymbol{w}^\top \boldsymbol{w} + \alpha_1||\boldsymbol{w}|| + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}),$$

with the corresponding gradient (technically, sub-gradient):

$$\nabla_{\boldsymbol{w}}\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \alpha_2\boldsymbol{w} + \alpha_1 \operatorname{sign}(\boldsymbol{w}) + \nabla_{\boldsymbol{w}} J(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{w}).$$

The elastic-net regularization method overcomes the limitations of the LASSO. When LASSO is used with high-dimensional data with few examples—large p and small n, it will selects at most n variables before saturating. In addition, if there is a group of highly correlated variables, then the LASSO tends to select one variable from a group and ignore the others. To overcome these limitations, the elastic net adds a quadratic part to the penalty (the ridge regression term $\frac{1}{2}||\boldsymbol{w}||_2^2$), which makes the loss function strongly convex and therefore, has a unique minimum.

Meanwhile, the naive version of elastic net method finds an estimator in a two-stage procedure: first for each fixed α_2 it finds the ridge regression coefficients, and then does a LASSO type shrinkage. This kind of estimation incurs a double amount of shrinkage, which leads to increased bias and poor predictions. To improve the prediction performance, the authors rescale the coefficients of the naive version of elastic net by multiplying the estimated coefficients by $(1 + \alpha_2)$.

Examples of where the elastic net regularization method has been applied are:

- Support vector machine
- Metric learning
- Portfolio optimization

□

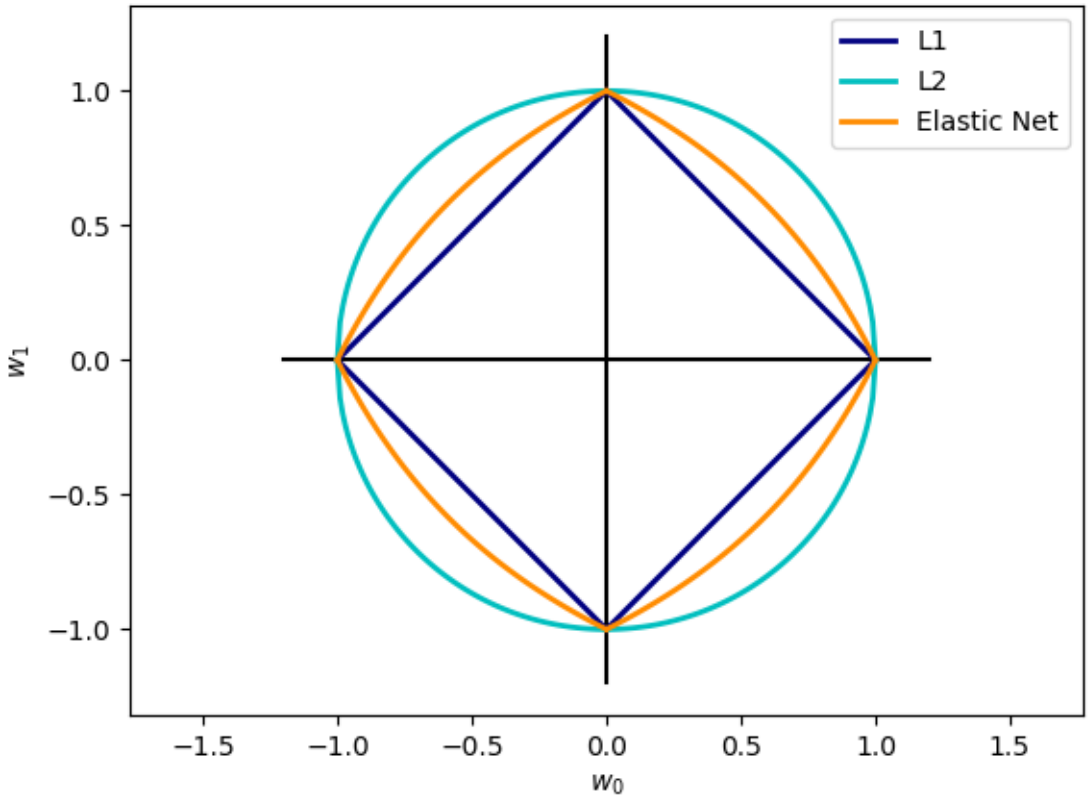

Figure 3: LASSO, Ridge and Elastic-Net geometric interpretations

2 Extras

Exercise 6. When to use Regularization for overfitting?

Proof. Traditional methods like cross-validation, stepwise regression to handle overfitting and perform feature selection work well with a small set of features, but these techniques (regularization) are a great alternative when we are dealing with a large set of features. □

Exercise 7. What does Regularization achieve?

Proof. A standard least squares model tends to have some variance in it, i.e. this model won't generalize well for a data set different than its training data. **Regularization, significantly reduces the variance of the model, without substantial increase in its bias**. So the tuning parameter α, used in the regularization techniques described above, controls the impact on bias and variance. As the value of α rises, it reduces the value of coefficients and thus reducing the variance. **Till a point, this increase in α is beneficial as it is only reducing the variance(hence avoiding overfitting), without loosing any important properties in the data**. But after certain value, the model starts loosing important properties, giving rise to bias in the model and thus underfitting. Therefore, the value of α should be carefully selected. □