Lecture 4

Leila Wehbe

10-701 Spring 2019

1 Announcements

- HW1 is due Thursday. Remember it takes about 10 minutes to submit it so don't wait till the deadline.
- You have to do both a GradeScope and an Autolab submission.
- Waitlisted students should have received an email with instructions from Diane Stidle.

2 Regression

We have been interested so far in predicting y where y is discrete. We now move to predicting y where y is continuous. We assume that y is a function of $\mathbf{x}$ plus noise:

$$y = f(\mathbf{x}) + \epsilon$$

We want to learn $f : \mathbf{x} \to y$ given $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2)...(\mathbf{x}_n, y_n)\}$.

In this section of the course we will focus on parametric methods, and therefore consider parametric regression. But one can use other methods such as kernel regression or KNN regression which are not parametric. We thus specify a form for the function f with parameters θ and then we estimate θ using the data and some assumption on the distribution of the noise ϵ. For example if we assume that $\epsilon \sim \mathcal{N}(0, \sigma^2)$, then $y \sim \mathcal{N}(f(\mathbf{x}), \sigma^2)$, and we estimate the parameters of f from data.

3 Linear Regression

This is a parametric method in which we consider that the function f is linear:

$$y = \mathbf{x}^\top \beta + \epsilon$$

Where $\beta = (\beta^1, \beta^2 ... \beta^p)$. One way to learn estimate the parameters β is the minimize the Minimum Squared Error (MSE), leading to the ordinary least squares solution (OLS):

$$\widehat{\beta} = \arg\min_\beta \sum_i (y_i - \mathbf{x}_i^\top \beta)^2 = \arg\min_\beta ||\mathbf{y} - \mathbf{X}\beta||_2^2 = \arg\min_\beta (\mathbf{y} - \mathbf{X}\beta)^\top (\mathbf{y} - \mathbf{X}\beta)$$

where $\mathbf{y} = (y_1, y_2 ..., y_n)$ and $\mathbf{X} = (\mathbf{x}_1^\top, \mathbf{x}_2^\top ... \mathbf{x}_n^\top)$, and $||.||_2$ is the L_2 norm.

One way to check if $MSE(\beta) = (\mathbf{y} - \beta\mathbf{X})^\top (\mathbf{y} - \beta\mathbf{X})$ is a strictly convex function of β is to compute the second derivative:

$$\begin{aligned} MSE'(\beta) &= \frac{\partial (\mathbf{y} - \mathbf{X}\beta)^\top (\mathbf{y} - \mathbf{X}\beta)}{\partial \beta} \\ &= -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) \quad \text{(from homework 1)} \\ MSE''(\beta) &= 2\mathbf{X}^\top \mathbf{X} \end{aligned}$$

$\mathbf{X}^\top\mathbf{X}$ is positive definite or positive semi-definite depending on our input $\mathbf{X}$. We might have less samples than dimensions for $\mathbf{X}$, or the samples are more or less copies or linear combinations of each other. If $\mathbf{X}^\top\mathbf{X}$ is positive definite, $MSE(\beta)$ is strictly convex and has one minimum. We minimize with respect to β to obtain β_{OLS}:

$$\begin{aligned} MSE'(\beta_{OLS}) &= 0 \\ -2\mathbf{X}^\top\mathbf{y} + 2\beta_{OLS}\mathbf{X}^\top\mathbf{X} &= 0 \\ \beta_{OLS}\mathbf{X}^\top\mathbf{X} &= \mathbf{X}^\top\mathbf{y} \\ \beta_{OLS} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} \end{aligned}$$

Inverting $\mathbf{X}^\top\mathbf{X}$

If $\mathbf{X}^\top\mathbf{X}$ if not positive definite, it would not be invertible. One solution is to add a diagonal element: $\beta_{MSE} = (\mathbf{X}^\top\mathbf{X} + \gamma\mathbf{I_p})^{-1}\mathbf{X}^\top\mathbf{y}$. This is related to the Ridge Regression solution below.

Multiple output regression

Now imagine that each output is actually multi-dimensional: given $\mathbf{x}_i$, we want to predict $\mathbf{y}_i$ where $\mathbf{y}_i = (y_i^1, y_i^2 ... y_i^m)^\top$. Now the problem becomes:

$$\mathbf{y}_i = \mathbf{x}_i^\top\boldsymbol{\beta} + \boldsymbol{\epsilon}_i$$

where $\boldsymbol{\beta} = ({\beta^1}^\top, {\beta^2}^\top, ... {\beta^m}^\top)^\top$ and $\boldsymbol{\epsilon}_i = (\epsilon_i^1, \epsilon_i^2 ... \epsilon_i^m)^\top$ and $\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \sigma^2\mathbf{I}_m)$.

The MSE becomes:

$$\sum_i (\mathbf{y}_i - \mathbf{x}_i^\top\boldsymbol{\beta})^2 = ||\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}||_2^2 = (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^\top(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})$$

where $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n)$. You can go through the derivation and find that $\boldsymbol{\beta}_{OLS} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}$.

Are the values of the kth output $\mathbf{y}^k$ used in the computation of β_{OLS}^j for a given output $j \neq k$? Think about how matrix multiplication works.

4 Bias Variance Decomposition

In regression, the decision step consists in picking a specific estimate $\widehat{f}(x)$ of the value of $\mathbf{y}$ for each input x. This has a loss $L(y, \widehat{f}(\mathbf{x}))$. A common loss function is the squared loss, and the average loss is given by:

$$E[L] = \int (\widehat{f}(x) - y)^2 p(y, x) dy dx,$$

We have:

$$y = f(x) + \epsilon$$

such that $\epsilon \sim \mathcal{N}(0, \sigma^2)$. If we have infinite data, our model assumptions were correct, and we were able to learn $\widehat{f}(x) = f(x)$, but we would still have some error in predicting y due to the noise ϵ:

$$\begin{aligned} E[L] &= \int (\widehat{f}(x) - y)^2 p(y,x) dy dx \\ &= \int \epsilon^2 p(y,x) dy dx = \sigma^2. \end{aligned}$$

In real applications we have finite data of size n.
You can see in **[CB] 1.5.5** how to write the average loss as :

$$\begin{aligned} E[L] &= \int (\widehat{f}(x) - E[y|x])^2 p(x) dx + \int (E[y|x] - y)^2 p(x,y) dy dx && (1) \\ &= \int (\widehat{f}(x) - f(x))^2 p(y,x) dy dx + \int \epsilon^2 p(x) dx && (2) \\ &= \int (\widehat{f}(x) - f(x))^2 p(x) dx + \sigma^2, && (3) \end{aligned}$$

(in [CB] the variables are named differently).
Let's rewrite $(\widehat{f}(x) - f(x))^2$, the loss for a given x in another way, by adding and removing $E_{\mathcal{D}}[\widehat{f}(x)]$ where D is a variable denoting a training dataset of size n paired x_i and y_i samples:

$$\begin{aligned} (\widehat{f}_D(x) - f(x))^2 &= (\widehat{f}_D(x) - E_D[\widehat{f}_D(x)] + E_D[\widehat{f}_D(x)] - f(x))^2 \\ &= (\widehat{f}_D(x) - E_D[\widehat{f}_D(x)])^2 + (E_D[\widehat{f}_D(x)] - f(x))^2 + 2(\widehat{f}_D(x) - E_D[\widehat{f}_D(x)])(E_D[\widehat{f}_D(x)] - f(x)). \end{aligned}$$

The above formulation is for a single dataset D, however, we want the mean over all datasets D:

$$\begin{aligned} E_D\left[(\widehat{f}_D(x) - f(x))^2\right] &= E_D\left[(\widehat{f}_D(x) - E_D[\widehat{f}_D(x)])^2\right] + E_D\left[(E_D[\widehat{f}_D(x)] - f(x))^2\right] + E_D\left[\text{cross-term}\right] \\ &= E_D\left[(\widehat{f}_D(x) - E_D[\widehat{f}_D(x)])^2\right] + (E_D[\widehat{f}_D(x)] - f(x))^2 \quad \text{(cross-term's expected value is 0)} \\ &= \mathrm{Var}(\widehat{f}(x)) + \mathrm{Bias}(\widehat{f}(x))^2 \end{aligned}$$

Where:

$$\mathrm{Bias}\left[\widehat{f}(x)\right] = \mathrm{E_D}\left[\widehat{f}_D(x)\right] - f(x),$$

and

$$\mathrm{Var}\left[\widehat{f}(x)\right] = E_D\left[(\widehat{f}_D(x) - E_D[\widehat{f}_D(x)])^2\right].$$

this is for a single x. Going back to 3, when considering all x, the expected loss is:

$$\begin{aligned} E(L) &= \int \mathrm{Bias}[\widehat{f}(x)]^2 p(x) dx + \int \mathrm{Var}[\widehat{f}(x)]^2 p(x) dx + \sigma^2 \\ &= \mathrm{Bias}^2 + \mathrm{Variance} + \mathrm{noise}. \end{aligned}$$

You can read more about this in **[CB] 3.2** and **[HTF] 2.5**.
The average loss can therefore be decomposed in terms of bias, variance and unavoidable loss (error). We will see that picking a type of estimator and fitting its hyperparameters to data is akin to performing a bias-variance tradeoff. Some methods (such as adding L_1 or L_2 regularization) can achieve a useful reduction in variance but incur bias, which might be acceptable for our use.

5 Regularized Linear Regression

To reduce variance and improve generalization, one can penalize the regression weights. Various regularization penalties can be used.

5.1 Ridge regularization

The objective includes an L_2 penalty on the weights:

$$\beta_{ridge} = \arg\min_{\beta} = \arg\min_{\beta} ||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda_r ||\beta||_2^2$$

with $\lambda_r > 0$.

Ridge regression has a closed form solution:

$$\beta_{ridge} = (\mathbf{X}^\top \mathbf{X} + \lambda_r \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{y}. \tag{4}$$

Take the SVD decomposition of $\mathbf{X}$:

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\top$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and Σ is a diagonal matrix where the diagonal entries correspond to the singular values of $\mathbf{X}$ $\sigma_1, \sigma_2 ... \sigma_p$.

Then

$$\begin{aligned} \beta_{ridge} &= (\mathbf{X}^\top \mathbf{X} + \lambda_r \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{y} & (5)\\ &= (\mathbf{V}\Sigma\Sigma\mathbf{V} + \lambda_r \mathbf{I}_p)^{-1} \mathbf{V}\Sigma\mathbf{U}^\top \mathbf{y} & (6)\\ &= \mathbf{V}\Sigma_2 \mathbf{U}^\top \mathbf{y} & (7) \end{aligned}$$

where Σ_2 is a diagonal matrix where each entry i is:

$$\frac{\sigma_i}{\sigma_i^2 + \lambda_2}$$

The ridge penalty has the effect of shrinking the regression coefficients towards zero. This effect is more pronounced for small singular values as we see in the next section. Larger values of λ_r lead to more shrinkage.

Cross-validation and performance In order to obtain a good value for the parameter λ_r one can perform cross-validation on the training set. In each fold, different values of λ_r are used to estimate β on the training portion of that fold, then this estimate is used to predict the validation portion. The average performance for each λ_r is obtained and then the best value of λ_r over all the folds is used to retrain β using all the training data.

At a given CV fold, if p, the dimension of $\mathbf{x}_i$, is very big, it might be too expensive to use 4 at every λ_r because of the matrix inversion. It is usually faster to use 7 instead and recompute Σ_2 for every λ_r, then perform the required multiplications.

Tikhonov Regularization

$$\beta_{tikhonov} = \arg\min_{\beta} = \arg\min_{\beta} ||\mathbf{y} - \mathbf{X}\beta||_2^2 + ||\Gamma\beta||_2^2$$

of which Ridge Regression is a special case. The solution is:

$$\beta_{tikhonov} = (\mathbf{X}^\top \mathbf{X} + \Gamma^\top\Gamma)^{-1} \mathbf{X}^\top \mathbf{y}.$$

5.2 Lasso

(least absolute shrinkage and selection operator)

$$\beta_\lambda = \arg\min_\beta = \arg\min_\beta ||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda_L ||\beta||_1$$

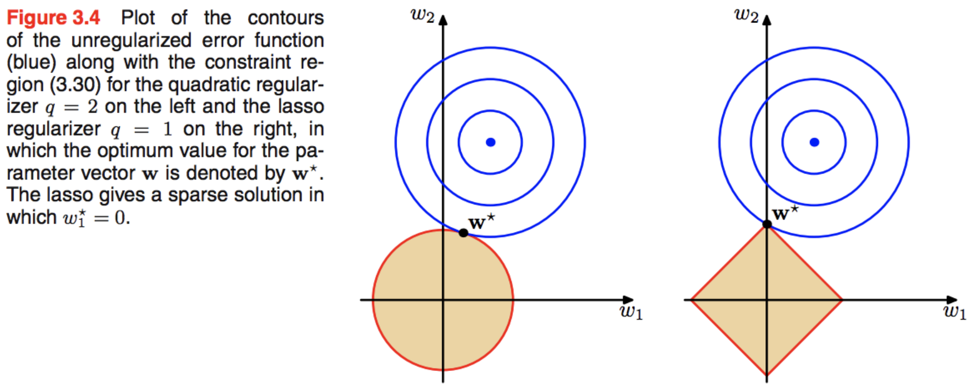
Figure 3.4 Plot of the contours of the unregularized error function (blue) along with the constraint region (3.30) for the quadratic regularizer $q = 2$ on the left and the lasso regularizer $q = 1$ on the right, in which the optimum value for the parameter vector $\mathbf{w}$ is denoted by $\mathbf{w}^\star$. The lasso gives a sparse solution in which $w_1^\star = 0$.

Figure 1: from [CB] page 146

The lasso penalty encourages sparsity. The level is determined by λ_L.

β_λ has no closed form solution. Furthermore the objective function is not differentiable. There exist several optimization approaches to approximate it, such as using the proximal gradient method.

The lasso problem can be formulated as:

$$\beta_r = \arg\min_\beta = \arg\min_\beta ||\mathbf{y} - \mathbf{X}\beta||_2^2$$
$$\text{subject to} \quad ||\beta||_1 \leq r$$

The sets $\{\beta_\lambda\}_{\lambda\in(0,\infty)}$ and $\{\beta_r\}_{r\in(0,\infty)}$ have a one to one mapping (the specific mapping depends on $\mathbf{X}$ and $\mathbf{y}$). Figure 1 shows the unregularized MSE contours, as well as one constraint region with a level r for both Ridge and Lasso. The optimization will try to find the point within the constraint region that has the smallest MSE. For Ridge, this is not encouraged to be a sparse point (i.e. have zero coordinates). For Lasso, the optimum is more likely to be sparse, especially as we move to higher dimensions and there are many corners and edges that can satisfy the smallest MSE in the constrain region.

Lasso Path as λ_L is decreased more and more elements of β_{lasso} become non-zero. The elements that became non-zero at higher λ_L can also change value as well, but remain non-zero. The coefficients do not change monotonically, they can increase or decrease.

6 Note on assumptions

We have started here from the assumption that $y = \mathbf{x}\beta + \epsilon$. However, this assumption being correct is not a requirement for using OLS, Ridge or Lasso. One can always use these methods to perform *predictions*: how good the predictions are will vary. For example if you use OLS and the true model is not linear, you will just get the best linear fit. The concepts of Bias and Variance tradeoff still apply in those cases.

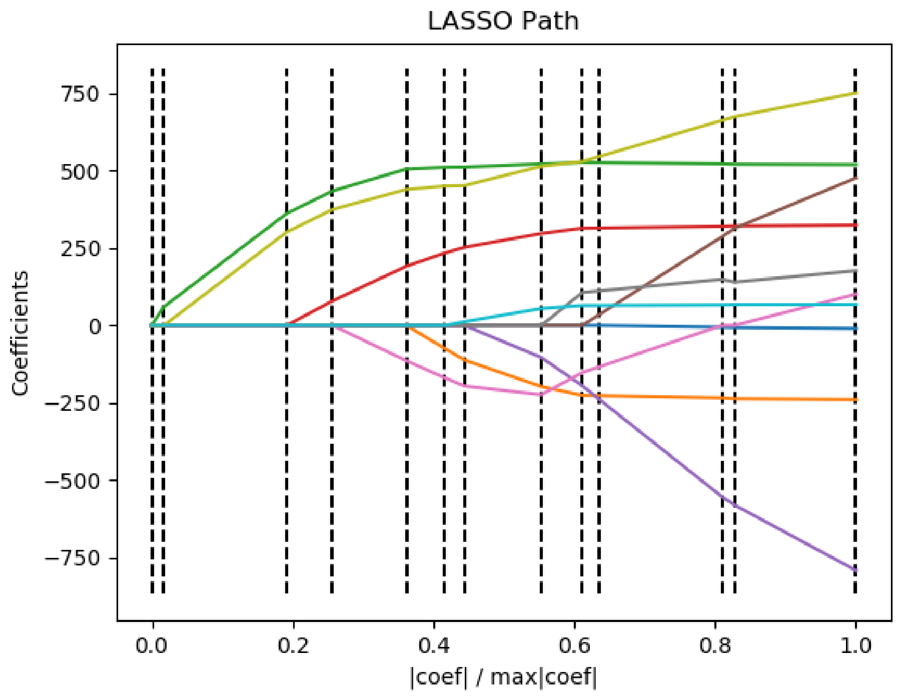

Figure 2: from https://scikit-learn.org/stable/auto_examples/linear_model/plot_lasso_lars.html the effect of different settings of the regularization parameter on the coefficients of different variables (different colors). 1 on the x-axis corresponds to the OLS solution. Decreasing values of x correspond to additional regularization.

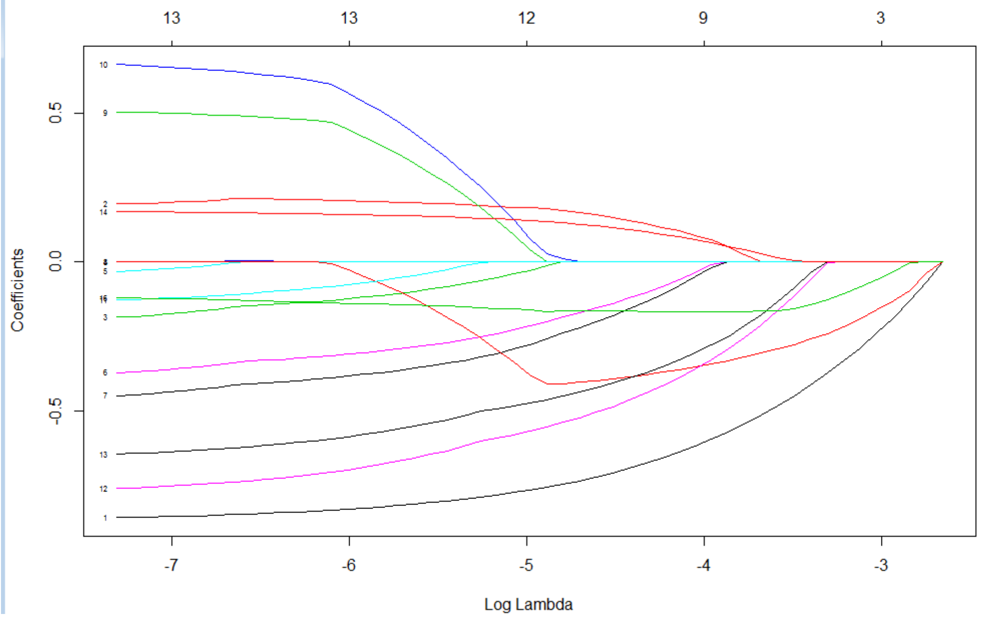

Figure 3: Another common way to plot the Lasso Path, from https://stats.stackexchange.com/questions/177788/generating-lasso-path-for-feature-selection. The coefficients for different variables are shown with different colors as λ_L varies. As λ_L gets reduced (right to left), more variables have a non-zero coefficient.