Least Squares and Friends

Duke Course Notes
Cynthia Rudin

We are going to discuss several important and interesting facts about least squares, ridge regression and L1-regularized regression (also called lasso). Perhaps you will find they are more unusual and interesting than you thought.

First, I need to explain the idea of generative models. If you are already familiar with Bayesian analysis, you can skip this. But if you are only familiar with the optimization perspective, this might be helpful. It will show you the important point that the prior in Bayesian analysis is analogous to the regularization term in the optimization-based approach, whereas the likelihood term is the loss term.

Equivalence between frequentist perspective and generative perspective

Warning: the term "model" here is used for several different things! I usually use "model" to refer to a function that predicts y from x. It can also be used as a set of probabilistic assumptions for how the data were generated. Unfortunately since there's no other word, I use the word "model" for both of these! Sorry about that!

The generative perspective is to think about a probabilistic approach for generating the *model* before the data are seen (this is your *prior* knowledge of the model). Then think about how the *data* are generated from the model (if the data were very likely to be generated from this process, the *likelihood* is large). The example we will give is for linear regression in 1D. We have some prior knowledge that the slope of the line, $\boldsymbol{\lambda}$, is small, so we define a prior that depends on $\|\boldsymbol{\lambda}\|$. We also have knowledge that if we know the true value of $\boldsymbol{\lambda}$, the data are generated close to the line defined by $\boldsymbol{\lambda}$. So in that case, our likelihood term would depend on the distance between y and $f_{\boldsymbol{\lambda}}(\mathbf{x})$. The likelihood and prior are both functions of the model.

Here's Bayes Rule.

$$\begin{aligned} P(\text{model}|\text{data}) &\propto P(\text{data}|\text{model}) \quad P(\text{model}) \\ \text{posterior} &\propto \text{likelihood} \cdot \text{prior}. \end{aligned}$$

You might be wondering why I left out a term that should look like $P(\text{data})$. It is because our goal is to figure out what the model is. $P(\text{data})$ doesn't depend on the model, so we don't need it in our whole discussion.

Taking the log of both sides:

$$\log \text{posterior} \propto \log \text{likelihood} \cdot \log \text{prior}.$$

If I want to create a good model, it should agree both with my prior knowledge and with my data, so I would like a model that maximizes the posterior. This is called a *maximum a posteriori* (MAP) model.

$$\max_{\text{model}} \log \text{posterior}$$

If I maximize the log posterior, it's the same as minimizing the negative log likelihood plus negative log prior:

$$\min_{\text{model}} (-\log \text{likelihood} - \log \text{prior})$$

In supervised learning, we have:

$$\min_{\text{model}} \text{loss} + \text{regularization}$$

Maximum likelihood is a special case. This is where we don't use a prior. (Or equivalently, the prior is uniform so log prior is constant, meaning that it isn't a function of the model.)

So here, we can see the equivalence between the -log likelihood term and the loss (both of which depend on data), as well as the equivalence of the -log prior term with the regularization (neither of which depend on data).

Before we move on, I would like to address the elephant in the room, which is whether we need to actually believe that the data were generated from a probabilistic process, and whether we actually have prior beliefs about that process. In

statistical learning theory we avoided these kinds of statements at all costs! The truth is that you can happily use the tools of Bayesian analysis without believing that these are realistic mechanisms for the creation of your dataset. As long as you think it's ok to maximize both the likelihood and the prior you've created, then it's fine. That's why they call it a "model" rather than a "hypothesis" I suppose. But no, belief is definitely not necessary to use the tools of Bayesian analysis!

1 Ridge regression has a frequentist and a generative interpretation

Ridge regression uses the squared loss and ℓ_2 regularization:

$$\min_{\boldsymbol{\lambda}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f_{\boldsymbol{\lambda}}(\mathbf{x}_i))^2 + C\|\boldsymbol{\lambda}\|_2^2 \text{ where } f_{\boldsymbol{\lambda}}(\mathbf{x}_i) = \sum_j \lambda_j x_{ij}.$$

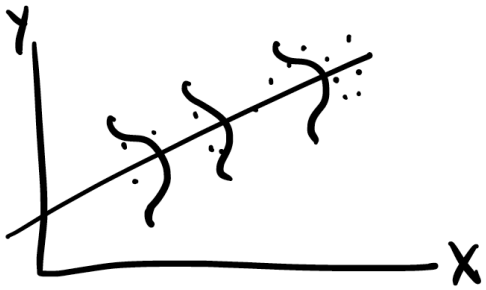
Let's consider a Bayesian version of it. Usually they use notation $\boldsymbol{\beta}$ rather than notation $\boldsymbol{\lambda}$ in statistics, so I'll switch over to that just for this section. Here is the process for generating the model $\boldsymbol{\beta}$, and then the data $\mathbf{Y}$ given the model.

$$\begin{aligned} \boldsymbol{\beta} &\sim N(\mathbf{0}, \tau^2 \mathbf{I}), \\ \mathbf{Y} &\sim N(\mathbf{X}\boldsymbol{\beta}, \sigma^2 \mathbf{I}) \end{aligned}$$

where τ is a number and $\mathbf{I}$ is a matrix that is 1 along the diagonals. This covariance matrix just says that each β_j is chosen independently of the other ones. We have no prior knowledge that any of the β_j's are naturally connected to each other, so a diagonal covariance matrix makes sense.

You might be wondering why $\mathbf{X}$ seems to be given and not random. Interestingly, the traditional statistical perspective does not view the $\mathbf{X}$ matrix as random, they view it as given. So we will too for this lecture.

This figure shows the data generation process. First we have prior knowledge about the slope of the line, then once that is known, we use it to generate data from a gaussian surrounding the line.



The posterior is

$$P(\boldsymbol{\beta}|\mathbf{Y},\mathbf{X}) = P(\mathbf{Y}|\boldsymbol{\beta},\mathbf{X}) \cdot P(\boldsymbol{\beta}) \cdot \frac{1}{Z},$$

where, as I mentioned earlier, the Z doesn't depend on the model. It is called the "evidence," and it is $\int P(\mathbf{Y}|\boldsymbol{\beta}\mathbf{X})p(\boldsymbol{\beta})d\boldsymbol{\beta}$. Now I will substitute in the gaussian distributions I have above. I will put all of the terms that are constants with respect to $\boldsymbol{\beta}$ in light gray.

$$\begin{aligned}
P(\boldsymbol{\beta}|\mathbf{Y},\mathbf{X}) &= \frac{1}{Z}\frac{1}{(2\pi)^{n/2}\sigma^n}\exp\left(-\frac{1}{2\sigma^2}\|\mathbf{Y}-\mathbf{X}\boldsymbol{\beta}\|_2^2\right)\cdot\frac{1}{(2\pi)^{p/2}\tau^p}\exp\left(-\frac{1}{2\tau^2}\|\boldsymbol{\beta}\|_2^2\right)\\
&= \frac{1}{Z}\frac{1}{(2\pi)^{n/2}\sigma^n}\frac{1}{(2\pi)^{p/2}\tau^p}\exp\left(-\frac{1}{2\sigma^2}\|\mathbf{Y}-\mathbf{X}\boldsymbol{\beta}\|_2^2\right)\cdot\exp\left(-\frac{1}{2\tau^2}\|\boldsymbol{\beta}\|_2^2\right)\\
&= \text{stuff}\cdot\exp\left(-\frac{1}{2\sigma^2}\|\mathbf{Y}-\mathbf{X}\boldsymbol{\beta}\|_2^2\right)\cdot\exp\left(-\frac{1}{2\tau^2}\|\boldsymbol{\beta}\|_2^2\right).
\end{aligned}$$

Here the stuff doesn't depend on the model, so it won't influence our optimization for the model.

$$-\log p(\boldsymbol{\beta}|\mathbf{Y},\mathbf{X}) = -\log \text{stuff} + \frac{1}{\sigma^2}\|\mathbf{Y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 + \frac{1}{\tau^2}\|\boldsymbol{\beta}\|_2^2.$$

Now we are back to ridge regression! The σ and τ trade off between the loss term (a.k.a. the likelihood term) and the regularization term (a.k.a. the prior term), so they together effectively act as the regularization term.

Hence, ridge regression has both a frequentist and a Bayesian interpretation.

2 Least squares regression has a closed-form solution

You might think that least squares regression is nothing special, but it really is! And the reason it is so special is because it has a closed-form solution. I will go

back to the $\boldsymbol{\lambda}$ notation now for the coefficient vector of my linear model. Here is least squares.

$$\min_{\boldsymbol{\lambda}} F(\boldsymbol{\lambda}), \text{ where } F(\boldsymbol{\lambda}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f_{\boldsymbol{\lambda}}(\mathbf{x}_i))^2 \text{ and } f_{\boldsymbol{\lambda}}(\mathbf{x}_i) = \sum_j \lambda_j x_{ij}.$$

Written in matrix notation,

$$F(\boldsymbol{\lambda}) = \|\mathbf{Y} - \mathbf{X}\boldsymbol{\lambda}\|_2^2.$$

To minimize this, I set the gradient to 0. I like to calculate one component of the gradient rather than working the calculation out in vectors, just to make sure I did it right.

$$\frac{\partial F(\boldsymbol{\lambda})}{\partial \lambda_j} = -\sum_i 2(y_i - \mathbf{x}_i\boldsymbol{\lambda})x_{ij} = -2\mathbf{X}_{\cdot j}^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\lambda})$$

where $\mathbf{X}_{\cdot j}$ is the jth feature vector. Now it is easy to see what the gradient would be, just by stacking the partial derivatives for each j:

$$\nabla F(\boldsymbol{\lambda}) = -2\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\lambda}) = -2(\mathbf{X}^T\mathbf{Y} - \mathbf{X}^T\mathbf{X}\boldsymbol{\lambda}).$$

Setting this to $\mathbf{0}$, the solution is called $\boldsymbol{\lambda}^*$.

$$\mathbf{X}^T\mathbf{Y} = \mathbf{X}^T\mathbf{X}\boldsymbol{\lambda}^*$$

and inverting the big matrix,

$$\boldsymbol{\lambda}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}.$$

Thus, least squares has an analytical solution!

I will introduce a small bit of notation before moving on. Let's make predictions on all of the $\mathbf{X}$ values using the linear model with coefficients $\boldsymbol{\lambda}^*$. These predictions are:

$$\hat{\mathbf{Y}} = \mathbf{X}\boldsymbol{\lambda}^* = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}\mathbf{Y} =: \mathbf{H}\mathbf{Y},$$

where here I have defined the "hat" matrix $\mathbf{H}$. You can think of $\mathbf{H}$ as smoothing out the values of $\mathbf{Y}$ since it is a map from their original values (which may be a bit scattered) to their predictions $\hat{\mathbf{Y}}$, which are all on a line.

3 Ridge regression has a closed form solution

Let us repeat the same calculations we did for least squares but with the regularization term.

$$F(\boldsymbol{\lambda}) = \|\mathbf{Y} - \mathbf{X}\boldsymbol{\lambda}\|_2^2 + C\|\boldsymbol{\lambda}\|_2^2.$$

Now for the gradient, one term at a time and then all together:

$$\frac{\partial F(\boldsymbol{\lambda})}{\partial \lambda_j} = -2\mathbf{X}_{\cdot j}^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\lambda}) + 2C\lambda_j$$

Stacking the terms for each j to form a vector:

$$\nabla F(\boldsymbol{\lambda}) \;=\; -2\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\lambda}) + 2C\boldsymbol{\lambda} = 2(-\mathbf{X}^T\mathbf{Y} + \mathbf{X}^T\mathbf{X}\boldsymbol{\lambda} + C\boldsymbol{\lambda}).$$

Setting this to $\mathbf{0}$ again at the solution $\boldsymbol{\lambda}^*$,

$$\mathbf{X}^T\mathbf{Y} = (\mathbf{X}^T\mathbf{X} + C\mathbf{I})\boldsymbol{\lambda}^*.$$

Inverting

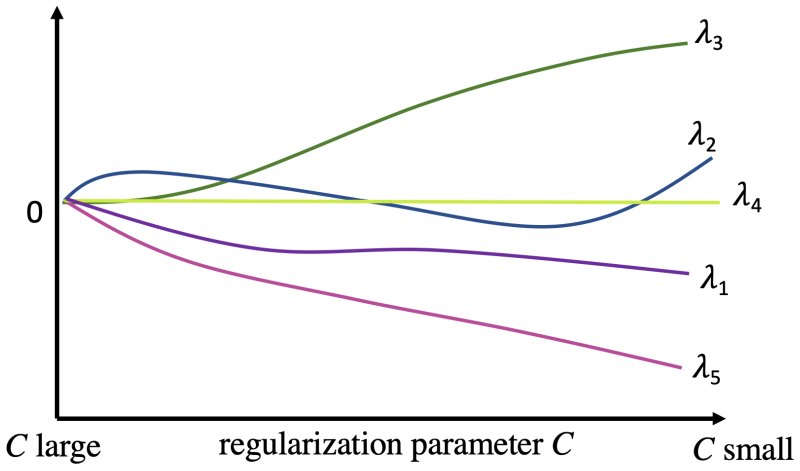
$$\boldsymbol{\lambda}^* = (\mathbf{X}^T\mathbf{X} + C\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}.$$

Thus, ridge regression also has an analytical solutions. It really is not common at all to have analytical solution problems for machine learning algorithms, so ridge regression is quite unique.

If you are familiar with numerical analysis, the ridge regression solution might look somewhat familiar. In particular, the Moore-Penrose pseudoinverse from the 1970's is

$$\mathbf{X}^+ = \lim_{\delta \to 0}(\mathbf{X}^T\mathbf{X} + \delta\mathbf{I})^{-1}\mathbf{X}^T$$

though that was designed to help with numerical instability. Since there could be many inverses to $\mathbf{X}^T\mathbf{X}$, we should pick one that makes sense, and adding a little noise to the diagonals to make it invertible and then sending that noise to zero would give it a proper stable answer. Here we don't want to set the regularization term to 0. In fact, we often like to explore what happens to the coefficients $\boldsymbol{\lambda}^*$ as we change the regularization constant C. One way to do this is by plotting "regularization paths." The horizontal axis is the regularization parameter. We plot all of the coefficients $\boldsymbol{\lambda}^*$ that come from ridge regression at each value of C.



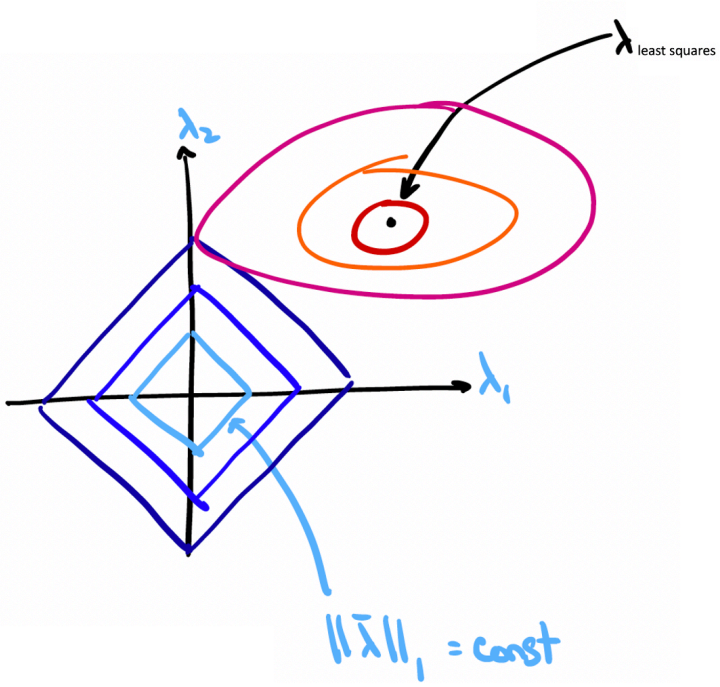
This doesn't have any particular uses, but it can be useful to see what happens to the coefficients as the regularization changes.

4 ℓ-1 penalized regression has no closed form solution

None of the above calculations work for ℓ_1 regularization (lasso). However, ℓ_1 penalties are often useful for something else, namely inducing sparsity. The reason they induce sparsity is shown by considering level sets. There are two terms:

$$F_1(\boldsymbol{\lambda}) = \frac{1}{n}\sum_i (y_i - f_{\boldsymbol{\lambda}}(\mathbf{x}_i))^2 + C\|\boldsymbol{\lambda}\|_1,$$

where $\|\boldsymbol{\lambda}\|_1$ is $\sum_i |\lambda_j|$. Let's consider what the level sets of these two terms look like. The level sets of $\|\boldsymbol{\lambda}\|_1$ are diamonds. To see this, think about 2 dimensions, and see that points on the level set $\|\boldsymbol{\lambda}\|_1 = 1$ include the points (0,1), (1,0), (-1,0), and (0,-1), as well as the diagonal lines connecting them (e.g., (1/2,1/2), (1/3,2/3), etc.). Those are shown in shades of blue in the picture. To get a low value of the ℓ_1 norm, we would like to choose level sets that are closer to 0 (lighter blue in the picture).



The level sets of the squared loss are shown in shades of red in the picture, they are ovals centered around the least squares solution. With no regularization at all, we would minimize the sum of the terms by choosing the least squares solution. With regularization, we would choose a point that has both low squared loss and low regularization, which might be the point on the λ_2 axis in the image where the level sets are shown to meet. This point has a low value of both terms. The fact that the minimizer is shown on an axis is not a coincidence. With the corners of the diamond jutting out, it becomes more likely that level sets of the loss will hit those corners. It doesn't always happen, but it happens fairly often that many of the solutions fall on various axes, where some of the values of λ_j equal 0. This is how the solution ends up being sparse.

Written another way, if we ask: to obtain the minimum squared loss for $\|\boldsymbol{\lambda}\|_1 = 1$, where along this diamond would we go? The answer will often be at a corner, just because these are the extreme points of the shape.

If we consider this perspective with ℓ_2 regularization, the level sets are circles and not diamonds. The axes are not special (they are not extreme points of the shape), and the solutions of ridge regression are not usually sparse.

5 Kernel Ridge Regression / Kernel Least Squares

Interestingly, we can kernelize ridge regression. The representer theorem states that since we are using ℓ_2 regularization, we can do it. It is not as easy as you might think though.
Here is the ridge objective again:

$$F(\boldsymbol{\lambda}) = \|\mathbf{Y} - \mathbf{X}\boldsymbol{\lambda}\|_2^2 + C\|\boldsymbol{\lambda}\|_2^2$$

Here, the objective is not written in terms of dot products on the $\mathbf{X}$'s. There is a trick, namely to replace $\boldsymbol{\lambda}$ with $\mathbf{Xr}$. This way, the objective is in terms of $\mathbf{r}$. (Once we solve for $\mathbf{r}$ we can get $\boldsymbol{\lambda}$ back using $\boldsymbol{\lambda} = \mathbf{Xr}$.) Thus, our objective becomes

$$F(\mathbf{r}) = \|\mathbf{Y} - \mathbf{X}\mathbf{X}^T\mathbf{r}\|_2^2 + C\|\mathbf{X}^T\mathbf{r}\|_2^2$$

Here, matrix $\mathbf{X}\mathbf{X}^T$ is a Gram matrix of inner products. (The matrix multiplication takes each row of $\mathbf{X}$, which is an observation, and applies it to each column of $\mathbf{X}^T$, which is again an observation. The result is $n \times n$.) Thus, we now have inner products, and we can replace the matrix of inner products by inner products in our new space, and the new matrix is called $\mathbf{K}$. Thus:

$$F(\mathbf{r}) = \|\mathbf{Y} - \mathbf{K}\mathbf{r}\|_2^2 + C\|\mathbf{X}^T\mathbf{r}\|_2^2$$

To simplify the regularization term, we write out the norm as an inner product (The inner product is just vector multiplication, $\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^T\mathbf{b}$.) This term is:

$$\|\mathbf{X}^T\mathbf{r}\|_2^2 = \langle \mathbf{X}^T\mathbf{r}, \mathbf{X}^T\mathbf{r} \rangle = (\mathbf{X}^T\mathbf{r})(\mathbf{X}^T\mathbf{r}) = \mathbf{r}^T(\mathbf{X})^{T^T}\mathbf{X}^T\mathbf{r} = \mathbf{r}^T\mathbf{X}\mathbf{X}^T\mathbf{r} = \langle \mathbf{r}, \mathbf{K}\mathbf{r} \rangle,$$

where we used that $(\mathbf{ab})^T = \mathbf{b}^T\mathbf{a}^T$. Now we have everything in terms of inner products $\mathbf{K}$:

$$F(\mathbf{r}) = \|\mathbf{Y} - \mathbf{K}\mathbf{r}\|_2^2 + C\langle \mathbf{r}, \mathbf{K}\mathbf{r} \rangle.$$

Thus, we can replace the matrix of inner products $\mathbf{K}$ with a matrix of inner products from any other valid RKHS.

Now let us solve for $\mathbf{r}$ by setting the gradient of $F(\mathbf{r})$ to 0.

$$\nabla F(\mathbf{r}) = -2\mathbf{K}(\mathbf{Y} - \mathbf{K}\mathbf{r}) + C2\mathbf{K}\mathbf{r} = \mathbf{0} \text{ at } \mathbf{r}^*.$$

Removing the $2\mathbf{K}$ multiplier terms on the left side:

$$-\mathbf{Y} + \mathbf{K}\mathbf{r}^* + C\mathbf{r}* = \mathbf{0}$$

$$(\mathbf{K} + C\mathbf{I})\mathbf{r}^* = \mathbf{Y}.$$

Inverting,

$$\mathbf{r}^* = (\mathbf{K} + C\mathbf{I})^{-1}\mathbf{Y}$$

and remember, $\boldsymbol{\lambda}^* = \mathbf{X}^T\mathbf{r}^*$.

There are two questions we need to answer. First, how do we make predictions? We cannot use $\boldsymbol{\lambda}^* = \mathbf{X}^T\mathbf{r}^*$, because $\mathbf{X}$ would need to turn into $\Phi(\mathbf{X})$ and we do not have access to Φ, only kernels. The second question is whether the solution we just found for $\boldsymbol{\lambda}^*$ is equal to the one we found earlier, before we were working in feature space? Neither of these two questions are hard, but they both require some work.

6 Making predictions

First, let us make a prediction at a new point $\tilde{\mathbf{x}}$. We will use that $\boldsymbol{\lambda}^* = \mathbf{X}^T\mathbf{r}^* = \sum_i \mathbf{x}_i^T r_i^*$, which we got from writing out the inner product.

$$\begin{aligned} f(\tilde{\mathbf{x}}) &= \tilde{\mathbf{x}}\boldsymbol{\lambda}^* = \tilde{\mathbf{x}}\sum_i \mathbf{x}_i^T r_i^* = \sum_i \tilde{\mathbf{x}}\mathbf{x}_i^T r_i^*, \text{ and with kernels,} \\ f(\tilde{\mathbf{x}}) &= \sum_i k(\tilde{\mathbf{x}}, \mathbf{x}_i) r_i^* \end{aligned}$$

Define $k(\tilde{\mathbf{x}}, \mathbf{x}_i) =: K_{\tilde{\mathbf{x}},i}$, and so the vector $\mathbf{K}_{\tilde{\mathbf{x}}}$ is the vector of inner products of $\tilde{\mathbf{x}}$ with all of the $\mathbf{x}_i$'s.

$$f(\tilde{\mathbf{x}}) = \sum_i K_{\tilde{\mathbf{x}},i} r_i^* = \mathbf{K}_{\tilde{\mathbf{x}}}^T\mathbf{r}^* = \mathbf{K}_{\tilde{\mathbf{x}}}^T(\mathbf{K} + C\mathbf{I})^{-1}\mathbf{Y}.$$

Now we can compute everything in terms of inner products, so we can use this formula to make predictions.

7 Reconciling the solutions

We now have two solutions for ridge regression that look completely different from each other: the original solution we derived, and the one using kernels. Even if you use the plain linear kernel, the solutions looks quite different, so

we need to show that they are actually equivalent. This actually requires some calculation.

Lemma. For invertible matrices,

$$\mathbf{P}\mathbf{B}^T(\mathbf{B}\mathbf{P}\mathbf{B}^T + \mathbf{R})^{-1} = (\mathbf{P}^{-1} + \mathbf{B}^T\mathbf{R}^{-1}\mathbf{B})^{-1}\mathbf{B}^T\mathbf{R}^{-1}$$

Proof: Let's work backwards, starting from the statement of the lemma. We will multiply both sides on the right by $(\mathbf{B}\mathbf{P}\mathbf{B}^T + \mathbf{R})$ so that the $(\mathbf{B}\mathbf{P}\mathbf{B}^T + \mathbf{R})^{-1}$ term will vanish:

$$\mathbf{P}\mathbf{B}^T = (\mathbf{P}^{-1} + \mathbf{B}^T\mathbf{R}^{-1}\mathbf{B})^{-1}\mathbf{B}^T\mathbf{R}^{-1}(\mathbf{B}\mathbf{P}\mathbf{B}^T + \mathbf{R}).$$

Multiplying both sides by $\mathbf{P}^{-1}+\mathbf{B}^T\mathbf{R}^{-1}\mathbf{B}$ on the left, the term $(\mathbf{P}^{-1}+\mathbf{B}^T\mathbf{R}^{-1}\mathbf{B})^{-1}$ will vanish:

$$\begin{aligned}(\mathbf{P}^{-1} + \mathbf{B}^T\mathbf{R}^{-1}\mathbf{B})\mathbf{P}\mathbf{B}^T &= \mathbf{B}^T\mathbf{R}^{-1}(\mathbf{B}\mathbf{P}\mathbf{B}^T + \mathbf{R}) \\ \mathbf{B}^T + \mathbf{B}^T\mathbf{R}^{-1}\mathbf{B}\mathbf{P}\mathbf{B}^T &= \mathbf{B}^T\mathbf{R}^{-1}\mathbf{B}\mathbf{P}\mathbf{B}^T + \mathbf{B}^T.\end{aligned}$$

Now both sides are the same, so we are done.

Let us look at the two solutions to ridge regression. The first one we derived is:

$$(\mathbf{X}^T\mathbf{X} + C\mathbf{I})^{-1}\mathbf{X}^T. \quad \text{(ridge)}$$

The second one (using a linear kernel for ridge regression) is:

$$\mathbf{X}\mathbf{r}^* = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T + C\mathbf{I})^{-1}. \quad \text{(kernel ridge)}$$

Are they equal? Let us use the lemma with $\mathbf{P} = \mathbf{I}$, $\mathbf{R} = C\mathbf{I}$ and $\mathbf{B} = \mathbf{X}$.

$$
\begin{aligned}
\mathbf{P}\mathbf{B}^T(\mathbf{B}\mathbf{P}\mathbf{B}^T + \mathbf{R})^{-1} &= (\mathbf{P}^{-1} + \mathbf{B}^T\mathbf{R}^{-1}\mathbf{B})^{-1}\mathbf{B}^T\mathbf{R}^{-1} \\
\mathbf{I}\mathbf{X}^T(\mathbf{X}\mathbf{I}\mathbf{X}^T + C\mathbf{I})^{-1} &= \left(\mathbf{I}^{-1} + \mathbf{X}^T\frac{1}{C}\mathbf{I}\mathbf{X}\right)^{-1}\mathbf{X}^T\frac{1}{C}\mathbf{I}^{-1} \\
\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + C\mathbf{I})^{-1} &= \left(\mathbf{I} + \frac{1}{C}\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\frac{1}{C}\mathbf{I} \\
&= \left(\frac{1}{C}\left(C\mathbf{I} + \mathbf{X}^T\mathbf{X}\right)\right)^{-1}\mathbf{X}^T\frac{1}{C}\mathbf{I} \\
&= C\left(C\mathbf{I} + \mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\frac{1}{C}\mathbf{I} \\
&= \left(C\mathbf{I} + \mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T \\
\text{(kernel ridge)}\quad \mathbf{X}\mathbf{r}^* &= \left(C\mathbf{I} + \mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T \quad \text{(ridge)}.
\end{aligned}
$$

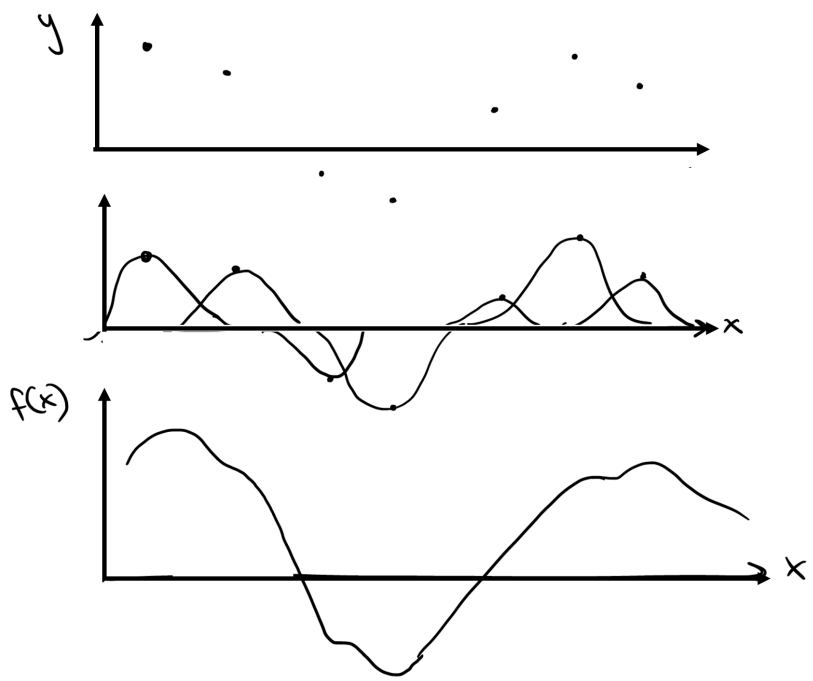
We have now shown that the two solutions are equivalent.

8 Kernel regression

There is another technique that is separate from kernel ridge regression that is called "kernel regression." It is a different technique than kernel ridge regression! Kernel regression is a locally-weighted average. The estimate of y given $\mathbf{x}$ is:

$$\hat{\mathbb{E}}(y|\tilde{\mathbf{x}}) = f(\tilde{\mathbf{x}}) = \frac{\sum_{i=1}^n K_h(\tilde{\mathbf{x}} - \mathbf{x}_i)y_i}{\sum_{i=1}^n K_h(\tilde{\mathbf{x}} - \mathbf{x}_i)}.$$

This is called the Nadaraya-Watson estimator. An easy way to understand this is that it places a bump function K_h centered on each $\mathbf{x}_i$ with height y_i, adds these bumps together, and then normalizes (so that denser areas with lots of points nearby do not have higher values than less-dense areas with fewer points nearby). The kernel functions K_h here are not learned, they are chosen beforehand. Here is an illustration.



The data is on top. A bump is placed on each point, with height y_i. These are summed and normalized.

As you probably noticed, kernel regression is *much* simpler than kernel ridge regression. Kernel ridge regression requires no optimization for the heights of the bumps. Kernel regression should generally perform better, because the heights of the bumps are globally optimized.