6.867 Section 2: Regression

Reading:

- Lecture 3: Bishop 3.1
- Lecture 4: Bishop 3.3–3.4; 3.5[1]

Contents

[1] Optional

1 Intro

Now we will look at the supervised learning problem of *regression*, in which the data given is a set of pairs

$$\mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\} \ ,$$

with $x^{(i)} \in \mathbb{R}^D$ and $y^{(i)} \in \mathbb{R}^k$, but we will mostly focus on the case in which $y^{(i)} \in \mathbb{R}$.

In the last section, we focused mostly on a simple estimation problem, in which we had a set of data that consisted only of $y^{(i)}$ values, and we had to predict $y^{(n+1)}$. The main issues we faced were understanding the properties of the process generating the y values and making the best prediction we could, given our loss function. Now, when we add a dependency on x values, we have to worry about *generalization*: that is, predicting a y value associated with an x value that we have never seen before. In the case where $x^{(i)} \in \mathbb{R}^D$, we can't really expect *ever* to have seen the value $x^{(n+1)}$ previously in the training set.

There is a beautiful mathematical story about the theory of generalization. A very informal caricature of two critical results is as follows:

- The *no free lunch* theorem: You can only obtain generalization from finitely many training examples if the algorithm searches a limited hypothesis space.
- We desire a learning algorithm to perform generalization and to be stable. It *generalizes* if the training error on a data set will converge to the expected error (on new data) as the size of the training set increases. It is *stable* if small perturbations in the data result in only small perturbations in the output hypothesis.

We can characterize generalization error by analyzing how the error of algorithm, such as an empirical risk minimization (choosing the hypothesis that has the least average loss on the data) method, converges to the expected error. It can be shown (for classification problems) that empirical error is at most $O(\sqrt{d/n})$ from the expected error, where d is the VC dimension of the hypothesis space.

Regularization is a strategy for both decreasing the VC dimension (hence increasing generalization ability) and increasing stability, by penalizing some form of complexity in the process of empirical risk minimization.

We will also see that using a Bayesian approach and averaging over the posterior on models when it is time to make a prediction can lead to stability and effective generalization.

We will look at learning prediction rules, probabilistic models, and distributions over models, and focus on the case in which the model represents a linear relationship between inputs and outputs. In later parts of the course, we will return to regression and look at non-linear and non-parametric approaches.

2 Representation

We will focus on linear models. They are mathematically straightforward and also relatively easy to handle algorithmically.

What does it mean to have a linear model? Generally, it will be that we can express the output value $y^{(i)}$ as a linear function of $x^{(i)}$. That is, that there are some weight values w_0 and $w = (w_1, \ldots, w_D)$ such that

$$y = w_0 + w_1 x_1 + \ldots + w_D x_D \ .$$

Such a model is known as a *linear regression* model.

As such, it is capable of representing a fairly limited class of relationships between input and output. We can extend the reach of these models by doing a fixed non-linear change

of representation. We can define a set of functions $\phi_j : \mathbb{R}^D \to \mathbb{R}$ that take an input x and compute a *feature value* $\phi_j(x)$. We will refer to ϕ_j as *basis functions* or sometimes *features*. Given a set of M basis functions, we can find a regression model that is linear in the feature values:

$$y = w_0 + w_1\phi_1(x) + \ldots + w_M\phi_M(x) \ .$$

We can, of course, capture our original model by defining a set of basis functions

$$\phi_i(x) = x_i \ .$$

But we can extend it considerably by making polynomial basis functions, augmenting the original input vector with all products of k of the original features, with k ranging from 2 to K. New features would include

$$\phi_a(x) = x_i^k \ \textit{and} \ \ \phi_b(x) = x_i \cdot x_j \ ,$$

or trigonometric or logistic basis functions, such as

$$\phi(x) = \sin(x_i) \ \textit{or} \ \ \phi(x) = \frac{1}{1 + \exp(-x_i)} \ .$$

Radial basis functions capture a notion of distance from some set of canonical points $\mu_i \in \mathbb{R}^D$:

$$\phi(x) = \exp\left(-\frac{(x - \mu_i)^2}{2s^2}\right) \ .$$

Although using non-linear basis functions means that the output y is a non-linear function of x, it is still easy to work with this representation because we will seek a linear function of the feature values of x.

In the following, to simplify notation, we will omit the possibility of using basis function to transform the input values, but the extension is completely straightforward.

3 Prediction rule

	No model	Prediction rule	Prob model	Dist over models
Regression		❋		

The most direct thing we can do is try to find weights w_0, w that define a regression function that optimizes some criterion on the training data. A popular criterion is mean squared error between the value the linear model with weights w_0, w would have us predict, $w \cdot x^{(i)} + w_0$, and the training output, $y^{(i)}$. So we would want to find

$$w^*, w_0^* = \arg\min_{w, w_0} \sum_{i=1}^{n} \left(w \cdot x^{(i)} + w_0 - y^{(i)}\right)^2 \ .$$

So, how can we find the minimizing weight vector w? Taking the gradient of the error function, setting to 0, and solving for w.

$$\text{Err}(w, w_0) = \sum_{i=1}^{n} \left(w \cdot x^{(i)} + w_0 - y^{(i)}\right)^2 \ .$$

Things will get easier if we do this in matrix notation. Let $\mathbf{X}$ be the *design matrix* or sometimes *data matrix*, W be the weight vector, and Y be the vector of training outputs:

$$\mathbf{X} = \underbrace{\begin{pmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \cdots & \cdots \\ 1 & x^{(n)} \end{pmatrix}}_{n \times (D+1)} \quad W = \underbrace{\begin{pmatrix} w_0 \\ w_1 \\ \cdots \\ w_D \end{pmatrix}}_{(D+1)\times 1} \quad Y = \underbrace{\begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \cdots \\ y^{(n)} \end{pmatrix}}_{n \times 1} \Bigg\} \; n \text{ input points}$$

Adding the column of 1s to the data matrix means we don't have to treat w_0 specially.

At least until we do regularization.

In an ideal world, we would have $\mathbf{X}W = Y$. But, we almost certainly can't do that. So, we will minimize MSE. Back to taking the derivative and setting it to 0.

You don't need to be able to do matrix derivatives for this class; but in case you do, the Wikipedia article has a great table of identities.

$$\begin{aligned} \text{Err}(W) &= (\mathbf{X}W - Y)^T (\mathbf{X}W - Y) \\ \nabla_{\mathbf{w}} \text{Err}(W) &= \mathbf{X}^T (\mathbf{X}W - Y) + \mathbf{X}^T (\mathbf{X}W - Y) \\ \mathbf{0} &= 2\mathbf{X}^T (\mathbf{X}W - Y) \\ \mathbf{X}^T Y &= \mathbf{X}^T \mathbf{X} W \\ W &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T Y \end{aligned}$$

So, our prediction rule is $h(x) = W_0 + W_{1:d} \cdot x$.
We will see, later, some cures for singular $\mathbf{X}^T\mathbf{X}$.
This is typically called *ordinary least squares* regression.

4 Probabilistic model

It might be good to make a probabilistic model, to help understand what assumptions we're making and what criterion we really want to optimize; also because we could use $\Pr(Y \mid X)$ to make decisions more effectively than with a decision rule $y = h(x)$.

	No model	Prediction rule	Prob model	Dist over models
Regression			❋	

4.1 Conditional Gaussian

Since we are trying to do regression, and we know that we will be given queries x and asked to predict the associated y, then one reasonable model is to estimate $\Pr(Y \mid X)$, which will let us immediately determine $\Pr(Y \mid X = x)$, given a query.

To approach this problem, we need to make an assumption about the form of this conditional distribution. Let us assume that:

- There is some underlying deterministic linear function h that relates X to Y and
- The observed values of Y are equal to $h(X; w, w_0)$, perturbed with additive Gaussian noise.

That is, that

$$Y \mid X \sim \text{Normal}(h(X; w, w_0), \beta^{-1}) \quad .$$

Here β is the precision (inverse variance) of the conditional distribution, and

$$h(x; w, w_0) = w_0 + w \cdot x \quad .$$

Given this model, we can try to find the *maximum likelihood estimators* for w, w_0.

$$\begin{aligned} w_{\mathbf{ml}}, w_{0\mathbf{ml}} &= \arg\max_{w,w_0} \prod_{i=1}^{n} \Pr(y^{(i)} \mid x^{(i)}; w, w_0) \\ &= \arg\max_{w,w_0} \sum_{i=1}^{n} \log \Pr(y^{(i)} \mid x^{(i)}; w, w_0) \\ &= \arg\max_{w,w_0} -\sum_{i=1}^{n} (w \cdot x^{(i)} + w_0 - y^{(i)})^2 \end{aligned}$$

Hey! This is the same as the least squares criterion! We already know the answer to this!

A story is that Gauss liked squared error as a criterion, and then "invented" the Gaussian distribution because its MLE is the minimizer of squared error.

We can also find the maximum-likelihood estimate of the variance parameter to be:

$$\frac{1}{\beta_{\mathbf{ml}}} = \frac{1}{n} \sum_{i=1}^{n} (w \cdot x^{(i)} + w_0 - y^{(i)})^2 \ .$$

4.2 Joint Gaussian

Another approach we might think of is to estimate $\Pr(X, Y)$ from the data. Then, given a particular query x, we could compute $\Pr(Y \mid X = x)$ using that distribution.

To approach this problem, we need to make an assumption about the form of this joint distribution. Let us assume that the joint distribution of X, Y is a Gaussian in $D + 1$ dimensions, of random variable Z that is obtained by concatenating Y and X, so that $z_0^{(i)} = y^{(i)}$ and $z_{1,\ldots,n}^{(i)} = x^{(i)}$. So

$$Z \sim \mathbf{Normal}(\mu, \Sigma) \ .$$

The estimates $\mu_{\mathbf{ml}}$ and $\Sigma_{\mathbf{ml}}$ are just the standard ML Gaussian parameter estimates (sample mean and sample covariance).

We are particularly interested in the conditional distribution $\Pr(Y \mid X)$, because we are solving a regression problem. Using standard results about the Gaussian, we can express

$$\Sigma_{\mathbf{ml}} = \begin{pmatrix} \Sigma_{YY} & \Sigma_{YX} \\ \Sigma_{XY} & \Sigma_{XX} \end{pmatrix}$$

Then

$$Y \mid X = x \sim \mathbf{Normal}(\mu_{Y|X=x}, \Sigma_{Y|X=x}) \ ,$$

where

$$\mu_{Y|X=x} = \mu_Y + \Sigma_{YX} \Sigma_{XX}^{-1} (x - \mu_x) \ ,$$

and

$$\Sigma_{Y|X=x} = \Sigma_{YY} - \Sigma_{YX} \Sigma_{XX}^{-1} \Sigma_{XY} \ .$$

Notice that $\mu_{Y|X=x}$ is *linear* in x! This gives us a model in which

$$w = \Sigma_{YX} \Sigma_{XX}^{-1} \ \text{ and } \ w_0 = \mu_Y - \Sigma_{YX} \Sigma_{XX}^{-1} \mu_X \ .$$

Furthermore, it's the same predictive hypothesis that you would have gotten if you had estimated the conditional distribution! This will not generally be the case—it relies on special properties of the Gaussian.

4.3 Regularization

Okay, but what happens if we have a lot of features and not a lot of data? The ML estimates can have a lot of variance. There are many strategies for reducing variance (at the cost of some bias, usually). Surprisingly many of them turn out to be the same thing.

4.3.1 L2 norm

Here is a *penalized* or *regularized* error function, using the squared L_2 norm of the weight vector as a penalty.

$$\text{Err}_{\text{ridge}}(w, w_0) = \sum_{i=1}^{n} \left(w \cdot x^{(i)} + w_0 - y^{(i)} \right)^2 + \lambda \|w\|_2^2 \ .$$

The parameter λ governs the bias/variance trade-off. Any value of $\lambda > 0$ induces bias and reduces the norm of the weight vector, which reduces variance.

Note that we are not penalizing w_0. If we did that, it would be harder for the line to translate along the Y axis, which would be a very strange kind of bias.

Another way to write it is:

$$\begin{aligned} w_{\text{ridge}}, w_{0\text{ridge}} &= \arg\min_{w, w_0} \sum_{i=1}^{n} \left(w \cdot x^{(i)} + w_0 - y^{(i)} \right)^2 \\ &\text{subject to } \ \|w\|_2^2 < \eta \end{aligned}$$

This method is also called *shrinkage* because we are trying to shrink the weight vector.

See appendix A for a discussion of the relationship between η and λ.

It's also called *ridge regression*. Here's why. We are going to make a *centered* data matrix $\mathbf{Z}$ by letting:

$$z_j^{(i)} = x_j^{(i)} - \bar{x}_j \ ,$$

and leaving off the column of 1's. Define the mean output

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y^{(n)} \ ,$$

and center the y values, as well, to define Y_c so that

$$y_c^{(i)} = y^{(i)} - \bar{y} \ .$$

Now we can write the ridge criterion as:

$$\begin{aligned} E_{\text{ridge}}(W) &= (Y_c - \mathbf{Z}W)^T (Y_c - \mathbf{Z}W) + \lambda W^T W \\ \nabla_W E_{\text{ridge}}(W) &= \mathbf{Z}^T (\mathbf{Z}W - Y_c) + \lambda W = 0 \\ W_{\text{ridge}} &= (\mathbf{Z}^T \mathbf{Z} + \lambda \mathbf{I})^{-1} \mathbf{Z}^T Y_c \ . \end{aligned}$$

We set W_0 to minimize the remaining prediction error once we have committed to weights W:

$$W_{0\text{ridge}} = (\bar{y} - W_{\text{ridge}}^T \bar{X}) \ .$$

Exercise: Given a new input x how do you use the W_{ridge} and $W_{0\text{ridge}}$ obtained from ridge regression to predict a y value?

Not surprisingly, increasing λ increases bias but decreases variance. The variance of the ridge-regression estimate is

$$\text{Var}(W_{\text{ridge}}) = \beta^{-1} (X^T X + \lambda I)^{-1} X^T X (X^T X + \lambda I)^{-1} \ ;$$

the bias is

$$\text{Bias}(W_{\text{ridge}}) = -\lambda (X^T X + \lambda I)^{-1} W \ ,$$

where W are the OLS weights. It can be shown that the total variance $\sum_j \text{Var}(W_{\text{ridge}}[j])$ is a monotone decreasing squence with respect to λ, while the total squared bias $\sum_j \text{Bias}(W_{\text{ridge}}[j])$ is a monotone increasing sequence with respect to λ.[2]

There are a couple of other interesting ways to think about regularization in this setting. They were developed independently, but turn out to do the same thing as ridge regression.

[2] Notes from UKY BST 764 by Patrick Breheny.

- **Noisy training data** If you take each input point $x^{(i)}$ and add a random value ϵ_{ij} drawn from a Gaussian with zero mean and variance σ^2 to each component $x_j^{(i)}$, then it's equivalent to regularizing with $\lambda = \sigma^2$.
- **Adding "ridge" data** If you center the data, as described above, and then add an additional fictitious example, of the form $x^{(f)} = (\sqrt{\lambda}, \ldots, \sqrt{\lambda})$, $y^{(f)} = 0$, and do ordinary least squares regression, it is equivalent to doing ridge regression with parameter λ.

4.3.2 Lasso

There is another shrinkage method, similar to ridge regression, but using the L_1 norm rather than the squared L_2 norm of the weight vector as the penalty term. It is called "the lasso." Actually, it stands for *least absolute selection and shrinkage*.

Yippee-ay-yay!

$$w_{\text{lasso}}, w_{0\text{lasso}} = \arg\min_{w,w_0} \sum_{i=1}^{n} \left(w \cdot x^{(i)} + w_0 - y^{(i)} \right)^2 + \lambda \|w\|_1 \quad .$$

Or, in Lagrangian form:

$$\begin{aligned} w_{\text{lasso}}, w_{0\text{lasso}} \quad = \quad & \arg\min_{w,w_0} \sum_{i=1}^{n} \left(w \cdot x^{(i)} + w_0 - y^{(i)} \right)^2 \\ & \text{subject to} \quad \|w\|_1 < \eta \end{aligned}$$

It is of interest, because it tends to drive some of the w_j values to 0, thus resulting in solutions that are sparse (that is, the output may only depend on a subset of the input features). One way to think about it is that the optimal solution happens when the highest level set intersects the constraint surface. Because the constraint is "pointier," the intersection is more likely to be on an axis, which means some weight values will be 0. For lasso $w = (1, 0)$ is cheaper than $(1/\sqrt{2}, 1/\sqrt{2})$.

See Bishop figures 3.3 and 3.4.

Unfortunately, there is no closed form solution. The objective is not differentiable, but it is convex. That means the gradient doesn't exist, per se, but there is a set of *subderivatives* that characterize the optimality conditions for a weight vector.

Elastic nets use a convex combination of the ridge and lasso regularizers.

The *regularization* path is the weight of a feature plotted as a function of λ. Generally it starts at 0 for large λ and grows in magnitude to be the value it would have for OLS. Using the lasso, we find that weights "pop out" of their zero values one at a time.

See Murphy (or papers by Hastie, Tibshirani, and Friedman) for more about how to optimize with L1 regularization.

Lasso has become a method of choice. In particular, regularizing with the L_1 norm has sample complexity logarithmic in number of features, whereas using the quadratic penalty it has sample complexity linear in the number of features.[3]

We'll talk more about this later...for now, it's a rough measure of the number of training examples needed to, with high probability, return a hypothesis with a high degree of accuracy.

4.3.3 Model selection

We might, instead, want to perform l_0 regularization.

$$w_{L0}, w_{0L0} = \arg\min_{w,w_0} \sum_{i=1}^{n} \left(w \cdot x^{(i)} + w_0 - y^{(i)} \right)^2 + \lambda \|w\|_0 \quad .$$

The explicit goal here is to drive some weights to 0. This is a very hard criterion to optimize: it is neither differentiable nor convex. To optimize it, there is no choice but to enumerate all possible subsets of features!

Two reasonable greedy alternatives are:

[3] A. Ng, *Feature selection, L_1 vs L_2 regularization, and rotational invariance.*

- Forward selection (also called *orthogonal least squares*):
  - Do OLS on all D problems with feature sets of size 1. Pick the one that performs best on validation data. Call the associated feature f_1.
  - Do OLS on all $D - 1$ problems with feature sets of size 2, including feature f_1 with each possible other feature. Pick the one that performs best on validation data. Call the associated feature f_2.
  - Continue until error on the validation starts to go up. Stop and use the set of features that generated the best performance on validation set.
- Backward deletion: Like forward selection, but consider one feature on each stage for deletion from the working set of features.

More efficient than forward selection is *orthogonal matching pursuits*, in which you freeze each new weight when you add it, so you only need to optimize one new weight in each round. Lots of other variations.

Lasso also effectively performs model selection, although it also shrinks the weights that are non-zero. It is sometimes useful to do *debiasing*: Use lasso to decide which weights to set to 0; then do OLS with only those weights. Lasso's choice of which weights to set to 0 is not very stable. A correction for this is *bolasso*:

- Make a set of "bootstrap" subsets of $\mathcal{D}$ by sampling with replacement.
- Run lasso on all of them.
- Include a feature in your final model if it had a non-zero coefficient in at least 90% of the lasso runs.

Can be shown under some conditions to be *model-selection consistent* (that is, recover the original model in the limit of large data.)

4.3.4 Early stopping

If you are using an iterative method to find the OLS fit, the standard thing is to run until convergence: some point when the value of the objective function is only changing by a very small amount. But, if you plot the test-set error against iteration number, it is generally the case that if you stopped the iterative process before it had converged, you would have gotten a solution with better generalization ability. Bishop discusses this in 5.5.2. We won't go into any more detail, but it's interesting.

5 Distribution over models

	No model	Prediction rule	Prob model	Dist over models
Regression				❋

We will continue with the basic assumptions we made about the form of the conditional probabilistic model:

- There is some underlying deterministic linear function h that relates X to Y and
- The observed values of Y are equal to $h(X; w, w_0)$, perturbed with additive Gaussian noise.

That is, that

$$Y \mid X \sim \text{Normal}(h(X; W), \beta^{-1}) \ .$$

But now, instead of trying to find a maximum-likelihood estimate of W, we will try to find a distribution over the $\Pr(W \mid \mathcal{D})$. We will use W to stand for all the weights: w and w_0.

Then, to make a prediction, given an input x, we will integrate the loss function over the weight vectors:

$$\arg\min_g \int_W \int_a L(g, a) \Pr(a \mid x, W) \Pr(W \mid \mathcal{D}) \ .$$

5.1 Prior

We need to look for a conjugate family, if we want this to work out nicely. We know that the *data likelihood*, $\Pr(y \mid x, w)$ has the form of $\exp(-c(w \cdot x + w_0 - y)^2)$, so to keep it *all in the family* we will want a prior with a similar form. This will work out if we let

$$W \sim \text{Normal}(\mathbf{m_0}, \mathbf{S}_0) \ ,$$

where $\mathbf{m}_0$ is a $D + 1$-dimensional mean vector and $\mathbf{S}_0$ is a $(D + 1) \times (D + 1)$ covariance matrix.

5.2 Update

Now, we can do the multi-dimensional version of the Bayesian Gaussian update (see Bishop section). If we incorporate n new data points, then

$$W \mid \mathcal{D} \sim \text{Normal}(\mathbf{m_n}, \mathbf{S}_n) \ ,$$

where

$$\mathbf{m}_n = \mathbf{S}_n \left(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta \mathbf{X}^T Y\right) \ ,$$

and

$$\mathbf{S}_n^{-1} = \mathbf{S}_0^{-1} + \beta \mathbf{X}^T \mathbf{X} \ .$$

If we make a simplifying assumption that:

$$W \sim \text{Normal}(\mathbf{0}, \alpha^{-1}\mathbf{I}) \ ,$$

that is, that our prior over the weights has zero mean, and is round, with standard deviation radius $1/\sqrt{\alpha}$, then something very familiar will happen:

$$\mathbf{m}_n = \beta \mathbf{S}_n \mathbf{X}^T Y \ ,$$

and

$$\mathbf{S}_n^{-1} = \alpha \mathbf{I} + \beta \mathbf{X}^T \mathbf{X} \ .$$

So

$$\log \Pr(W \mid \mathcal{D}) = -\frac{\beta}{2}(\mathbf{X}W - Y)^T(\mathbf{X}W - Y) - \frac{\alpha}{2} W^T W + c \ .$$

So, to find the maximum *a posteriori* probability (MAP) weights, we optimize the same equations as for OLS with a quadratic regularizer, with $\lambda = \alpha/\beta$.

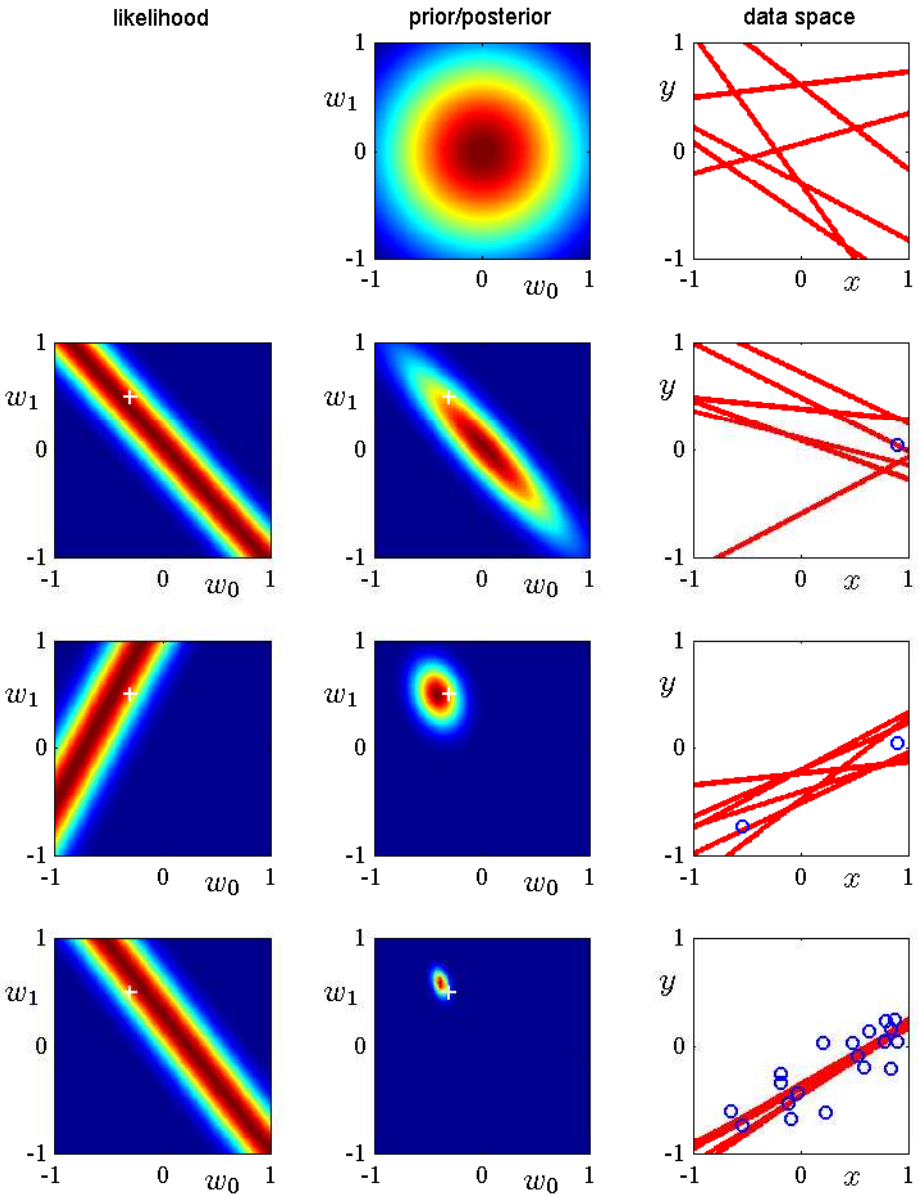


Figure 1: Figure 3.7 from Bishop

5.3 Example from Bishop

Look at figure 1. Simple model:

$$h(x; W) = w_0 + w_1 x \ .$$

Data generated from model

$$h(x; W) = -0.3 + 0.5x \ .$$

Assume noise variance is known, $\beta = 25$, and let $\alpha = 2.0$.

- **Column 1**: Plot of likelihood function $\Pr(y \mid x, W)$ for the training data point. This is the (smoothed) locus of lines that run through the training point (which is shown in blue in the right-hand column). True parameter values are shown with a white cross.
- **Column 2**: Distribution over W.
- **Column 3**: Samples from $\Pr(W \mid \mathcal{D})$

Now, we watch it go.

- **Row 1**: Round zero-mean prior on weights, and some samples.
- **Row 2**: Data point at about $(1, 0)$. We see the likelihood function for the data, multiply it by the prior and get the posterior. Samples now go near that point.
- **Row 3**: Another data point, near $(-0.6, -0.8)$. Likelihood function. Multiply by the new prior (old posterior) and the posterior gets tight. Samples now go near both points.
- **Row 4**: After many more samples, distribution has tightened considerably; mean is near the truth; samples are clustered and sensible.

5.3.1 Other norms

There is a whole family of prior distributions parameterized by q:

$$\Pr(w; \alpha) = \left(\frac{q}{2} \left(\frac{\alpha}{2} \right)^{1/q} \frac{1}{\Gamma(1/q)} \right)^D \exp\left(-\frac{\alpha}{2} \sum_{j=1}^{D} |w_j|^q \right) \ .$$

These are *not* conjugate with Gaussian observations unless $q = 2$. But the MAP estimate of $\mathbf{w}$ has the form of the regularized likelihood, with penalty $\sum_{j=1}^{D} |w_j|^q$.

5.4 Predictive distribution

But, really, why bother being Bayesian if you are just going to find the MAP estimate and be done! If we want to make a prediction, we will take expected risk, with the expectation taken with respect to the *posterior predictive distribution*

$$\Pr(y \mid \mathcal{D}, x) = \int_W \Pr(y \mid x, W) \Pr(W \mid \mathcal{D}) \ .$$

Writing out the densities and completing the square in the numerator, for the Gaussian case, we have

$$Y \mid X, \mathcal{D} \sim \text{Normal}(\mathbf{m}_n \cdot x, \sigma_n^2(x)) \ ,$$

where

$$\sigma_n^2(x) = \frac{1}{\beta} + x^T \mathbf{S}_n x \ .$$

So, variance in predictions is the sum of the inherent noise in the process plus variance due to uncertainty in the estimate of the weights.

Note! The variance in the posterior is dependent on x! The example in Bishop 3.3.2 is using **Gaussian basis functions**. These are *local* features centered around some canonical points x_p that go quickly to zero as you move away. So, in places where there is more data, we tend to have a tighter estimate.

5.5 Equivalent kernel

Let's imagine that our loss function is squared loss, so that the best prediction is the mean of the predictive distribution. (Actually, because this is the Gaussian, and the mean is equal to the median, the mean is optimal for any symmetric loss function.) We can think of our hypothesis, h, as the predictions we will make as a function of x, parameterized by the posterior $\mathbf{m}, \mathbf{S}$.

$$\begin{aligned} h(x; \mathbf{m}, \mathbf{S}) &= \mathbf{m}^T x = \beta x^T \mathbf{S} \mathbf{X}^T Y \\ &= \sum_{i=1}^{n} \beta x^T \mathbf{S} x^{(i)} y^{(i)} \\ &= \sum_{i=1}^{n} k(x, x^{(i)}) y^{(i)} \ ; \end{aligned}$$

where

$$k(x, x') = \beta x^T \mathbf{S} x' \ .$$

This is kind of cool! We can see the prediction that we make for some point x as a weighted function of the output values $y^{(i)}$ in the training data. They are weighted according to a *kernel* function k, which you can think of as measuring the distance between x and x'. In this case, the distance depends on $\mathbf{S}$, which is the posterior covariance matrix of distribution on weights, which itself depends on the data. Generally, the y values associated with x' values that are close to query point x are weighted more highly.

We will see later that this kernel idea is very important. Sometimes, instead of thinking about basis functions as a way of transforming our inputs, we will find it easier to think about a kernel, directly.

6 Bayesian model comparison for regression

In the Bayesian approach to regression, we have choices to make: we have both the model class (which basis functions to use) and the parameters of the prior. How should we select them? Cross-validation doesn't really fit nicely into the Bayesian worldview. But there is an elegant alternative.

Let's first consider the model class; let M_1 and M_2 be regression models with two different sets of basis functions. In the usual Bayesian inference process we will find a posterior distribution on the parameters for each model, given the data:

$$\Pr(\theta_1 \mid \mathcal{D}, \mathcal{M}_1) \ \text{ and } \ \Pr(\theta_2 \mid \mathcal{D}, \mathcal{M}_2) \ .$$

To see how good a "fit" we have, we can look at the likelihood of the data; but rather than looking at the likelihood conditioned on one particular set of parameters, we have to integrate it over the posterior distribution on the parameters. So, we'll be interested in

$$\Pr(\mathcal{M} \mid \mathcal{D}) \quad \propto \quad \Pr(\mathcal{D} \mid \mathcal{M}) \Pr(\mathcal{M})$$

You might imagine that the way we will control overfitting is by putting a distribution on $\mathcal{M}$ that prefers lower-complexity models. But the really interesting thing is that the *marginal likelihood*, $\Pr(\mathcal{D} \mid \mathcal{M})$, will do that for us to a significant degree.

This is also called the *model evidence*.

We can expand it out as:

$$\Pr(\mathcal{D} \mid \mathcal{M}) = \int_{\theta} \Pr(\mathcal{D} \mid \theta) \Pr(\theta \mid \mathcal{M}) \quad .$$

There are two things that can make the marginal likelihood be low:

- If θ describes a very large class of models and we don't have a lot of data, then our posterior on θ for a particular model will be very spread out. If it is, there will be lots of θ values with non-negligible probability, and they will not, generally, all be good models of $\mathcal{D}$, which will make the marginal likelihood low. Another way to think about it is that there will generally be a very large number of small data sets that are consistent with a complex model, and since $\Pr(\mathcal{D} \mid \mathcal{M})$ has to integrate to 1, it won't be able to assign very high probability to any one of these data sets. This is a problem of *high variance* in the Bayesian view.
- If θ describes a class of models, none of which fit the data very well, then all the $\Pr(\mathcal{D} \mid \theta)$ terms will be low. This is a problem of *high bias* in the Baysiean view.

If the true model is in the class of models, then we will on average prefer the correct model.

But even Bishop admits it is wise to keep a held-out set of data to see whether your models are roughly right.

Let's see what happens in a couple of examples.

6.1 Picking a model class: Bayesian model selection

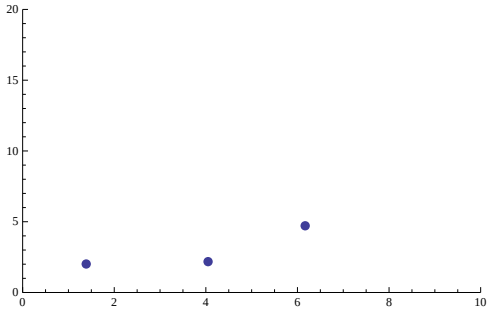
We will look at a regression problem with one-dimensional input and output. $\mathcal{M}_1$ will just use the basis $(1, x)$ as input. $\mathcal{M}_2$ will use the basis $(1, x, x^2, x^3)$ as input. In each case, we use a prior on W with zero mean, and diagonal covariance with $\alpha = 0.1$ (recall that α is a precision parameter, so it's like having variance 10.) Furthermore we assume the data is generated with known precision parameter $\beta = 1$.

This experiment is shown in detail in figure 2. We show two data sets, drawn from the same distribution with $\beta = 1$. We then plot the predictive distribution for each model, when trained on each data set. The blue line is the mean, and the red lines show the values of the predictive distributions one standard deviation away from the mean on each side. We can see that, on the small data set, the linear model makes reasonable predictions and the variability of the weights is not too high; however, the cubic model on that same data over-fits and has very high variance. This is borne out in the evidence computation: the evidence is higher for the simple model on the small data set. On the large data set, however, the situation changes. For both models, there is very little variance in the predictions; however, the predictions of the linear model have a lot of error. In this case the evidence for the cubic model is much higher.

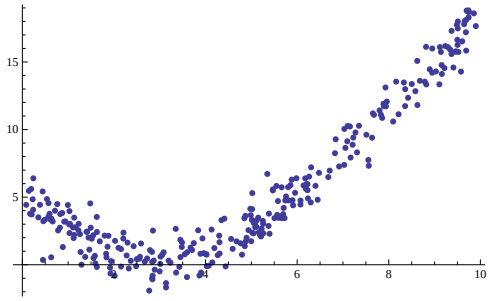
The truly Bayesian strategy would be to do model averaging, and make predictions

$$\Pr(y \mid \mathcal{D}) = \Pr(y \mid x, \mathcal{D}, \mathcal{M}_1) \Pr(\mathcal{M}_1 \mid \mathcal{D}) + \Pr(y \mid x, \mathcal{D}, \mathcal{M}_2) \Pr(\mathcal{M}_2 \mid \mathcal{D}) \quad .$$

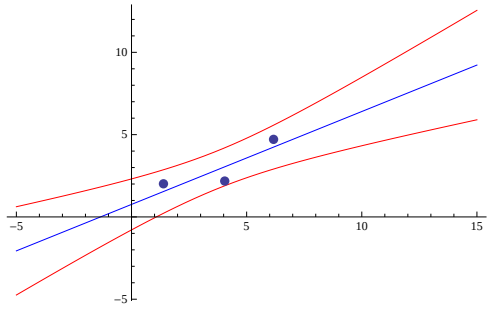
The result of that is shown in the bottom of the figure. This is particularly useful when one model is more confident in one part of the space and another is more confident in another part of the space.



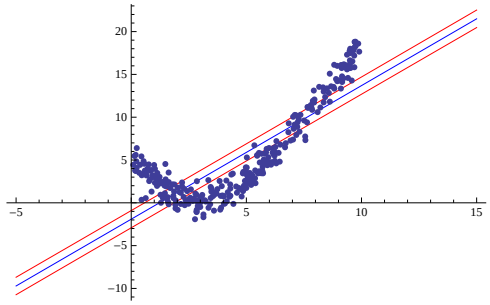
(a) Small data set ($n = 3$)



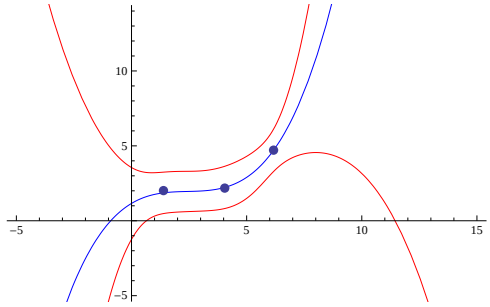
(b) Large data set ($n = 300$)



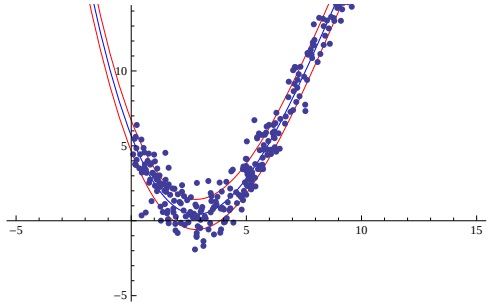
(c) Predictive distribution for $\mathcal{M}_1$ on small data set. Log Evidence: -4.



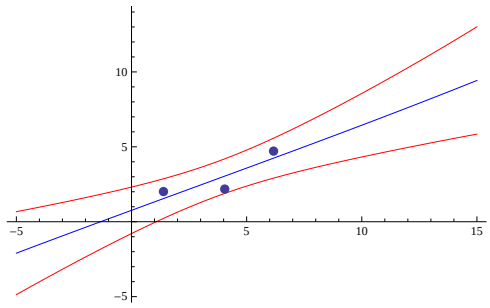
(d) Predictive distribution for $\mathcal{M}_1$ on large data set: Log Evidence: -738.



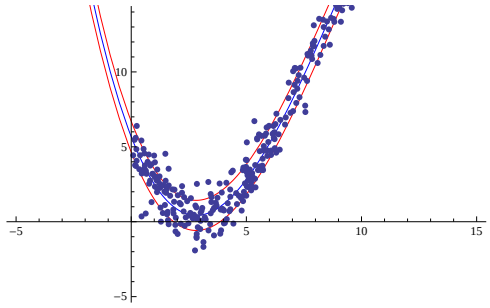
(e) Predictive distribution for $\mathcal{M}_2$ on small data set. Log Evidence: -6.



(f) Predictive distribution for $\mathcal{M}_2$ on large data set. Log Evidence: -204



(g) Predictive distribution averaged over models on small data set.



(h) Predictive distribution averaged over models on large data set.

Figure 2: Bayesian model selection.

6.2 Picking parameters: empirical Bayes

It is also possible to do this same thing for selecting parameters. For instance, you might not know how to pick a good value of α. For a single model, you can consider evidence as a function of α, and either average models over the α values (though this is hard to do analytically) or pick the best α value and use that.

The idea of using the data to pick the prior is pretty weird from a Bayesian philosophical perspective, but can be pragmatically useful. This strategy is called *empirical Bayes*.

A Relationship unconstrained and constrained forms of ridge regression

Here's the original set-up in the constrained form.

For simplicity in this, assume the constraint is going to be tight: that is that the maximum likelihood solution violates the constraint.

$$w_{\text{ridge}}, w_{0\text{ridge}} = \arg\min_{w,w_0} \sum_{i=1}^{n} \left(w \cdot x^{(i)} + w_0 - y^{(i)}\right)^2 \quad (1)$$

$$\text{subject to } \|w\|_2^2 \leqslant \eta \quad (2)$$

Let's assume that the data have already been centered, so we don't need to worry about w_0, and we'll let $\mathbf{Z}$ be our data matrix. Now the Lagrangian is:

$$\mathcal{L}(W, \lambda) = (\mathbf{Z}W - Y_c)^T(\mathbf{Z}W - Y_c) + \lambda(W^TW - \eta) \quad (3)$$

Set gradient of $\mathcal{L}$ with respect to W to 0, and solve for W:

$$\begin{aligned} \nabla_W \left((\mathbf{Z}W - Y_c)^T(\mathbf{Z}W - Y_c) + \lambda(W^TW - \eta)\right) &= 0 \\ 2(\mathbf{Z}W - Y_c) + 2\lambda W &= 0 \\ W &= (\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I})^{-1}\mathbf{Z}^TY_c \end{aligned}$$

Take the gradient of $\mathcal{L}$ with respect to λ and set to 0:

$$W^TW - \eta = 0 \ .$$

Now, we have η in terms of λ and the data.

$$\eta = W^TW \quad (4)$$

$$= \left((\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I})^{-1}\mathbf{Z}^TY_c\right)^T \left((\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I})^{-1}\mathbf{Z}^TY_c\right) \quad (5)$$

So, if you wanted to solve the unconstrained version of the problem in equation 3 with a fixed λ, that would be equivalent to solving the constrained version of the problem in equations 1 and 2 with the η determined in equation 5.

I had trouble inverting this to find λ in terms of η. But, a plot of the relationship between them for a simple data set, in figure 3.

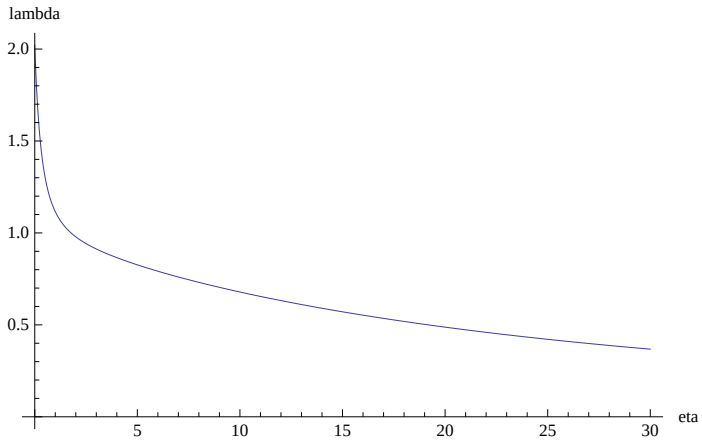
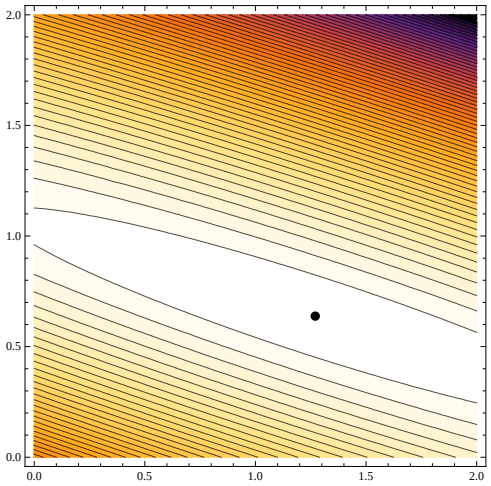
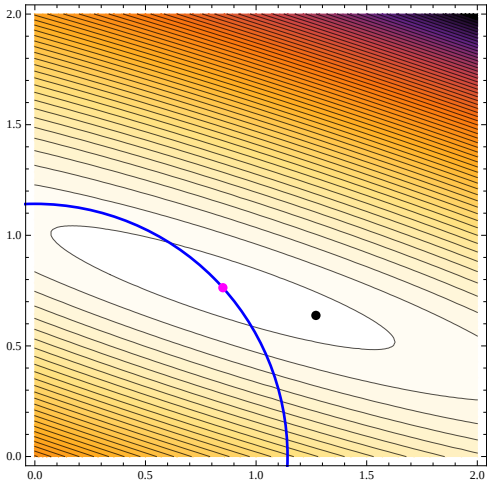


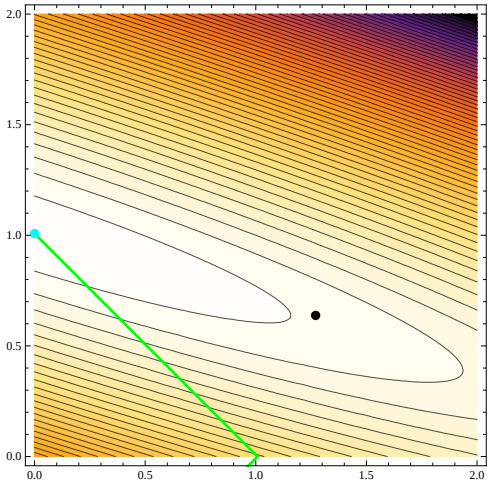
Figure 3: The hard constraint η corresponding to a regularization penalty λ for a simple data set.



(a) Maximum likelihood error contours



(b) Ridge error contours with $\lambda = 0.5$, $\eta \approx 1.3$; blue circle has radius $\sqrt{\eta}$; magenta dot is w_0, w_1 optimizing the ridge criterion



(c) L1 error contours with $\lambda = 3$; green line with half diagonal $\eta = 1.08$; cyan dot is w_0, w_1 optimizing the Lasso

Figure 4: Error contours in weight space.