**Introduction to Machine Learning** **Fall Semester**

Lecture 9

*Lecturer: Eran Halperin* *Scribe: Eran Halperin*

# 9.1 Introduction to Linear Regression

So far, we focused on classification problems. In a classification problem, we get as an input a multidimensional point $x$, which is typically a vector over the real, or another field, and the output is a class $y \in \{0, 1\}$. If there are more than two classes we may want to have $y \in \{0, 1, 2, \ldots, K-1\}$, where $K$ is the number of classes. In the training data we are given examples $(x_i, y_i)$, and in the test set we are given the value of $x$ and we need to find our best 'guess' for the value of $y$. As a side note, the point $x$ does not necessarily have to be a vector, it could be a vector over discrete values, or it can also be a more complex structure such as a graph or another combinatorial structure. We have seen examples for classifiers in the last few lectures, including the perceptron, support vector machines, and finally boosting.

In some cases, we are interested in giving a more detailed classification of the data. Particularly, in the limit we want to assign every value $x$ a value $y \in R$, thus we can think of it as a classification with an infinite number of classes. Based on the training data we need to learn the relation between $x_i$ and $y_i$, and then in the test data we can assign each point to a value based on an extrapolation of the relation $(x_i, y_i)$ to the entire domain of $x$. In the most general case, we can assume that the output is also a vector, i.e., $y \in R^k$, however in this lecture we will focus on the case where $y \in R$. One of the main techniques used for this extrapolation is termed regression analysis, which will be the topic of this lecture.

There are many natural scenarios in which regression analysis is useful. Consider for example the case in which we want to predict the risk of having high cholesterol based on the weight and height of a person. In this case, physicians usually use a measure called Body Mass Index (BMI), which is defined as $BMI = \frac{weight(kg)}{height^2(m)}$. We can measure the cholesterol level of a number of individuals, and measure their BMI values as well (Figure 9.1). It is clear from the figure that there is a monotonic relation between the BMI and the cholesterol level. In linear regression, we search for a line that in some sense fits this relation as best as we can. We will explain below what criterion we use in order to find this line.

Another example for regression is the prediction of the time a person spends in front of the television based on the person's age and gender. Such information could be useful for marketing purposes. Regression can also be used to predict the preference score that a person gives a book or a movie (e.g., in Netflix) based on some set of features describing the person's past choices (e.g., what is the average score the customer gave to action movies, to movies starring Nicholas Cage, etc.). In another example from the world of computation,

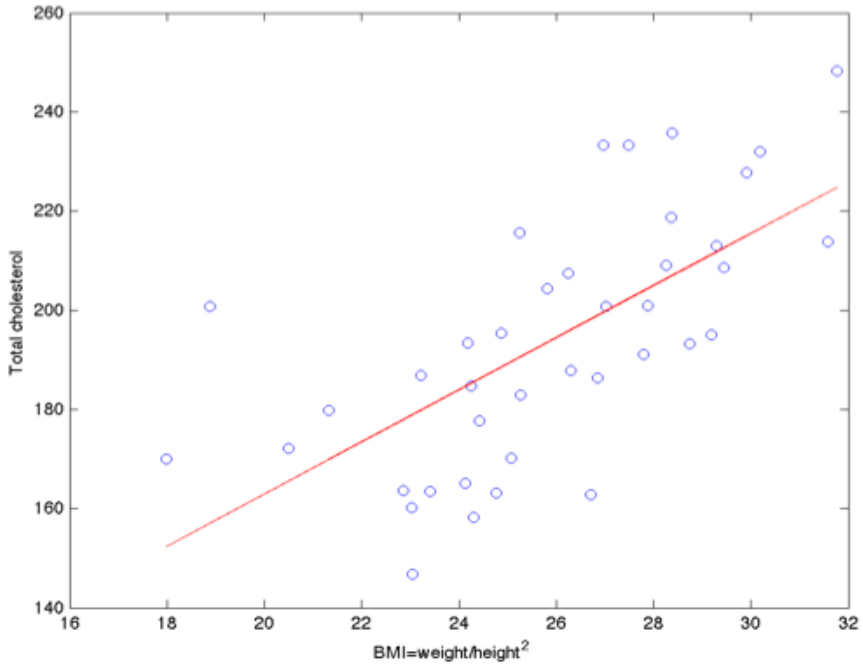


Figure 9.1: Regression line of cholesterol vs. BMI

we can try to predict the memory usage of a specific job based on some features of the job, for example the user submitting the job, his/her group, which type of machine the user is using (for example what the RAM of the machine is), and the time of day. There are many more natural examples and indeed regression is a widely applied tool that has been studied extensively.

## 9.2 Regression - Formal Definition

The input for a regression problem is a set of points $(x_i, y_i)$. Consider for now the case where $x_i, y_i \in R$. Assume there is a linear relation between $x$ and $y$, so that

$$y \approx ax + b$$

In linear regression, we are searching for such a line, by finding $a, b$ that minimize the following:

$$(\hat{a}, \hat{b}) = \arg\min_{a,b} \sum_i (y_i - ax_i - b)^2$$

In Figure 9.2 we illustrate the optimization. The values $r_i = y_i - \hat{a}x_i - \hat{b}$ are called the

residuals of the regression, and the objective of the regression is to find $a, b$ such that the sum of squares of the residuals is minimized.

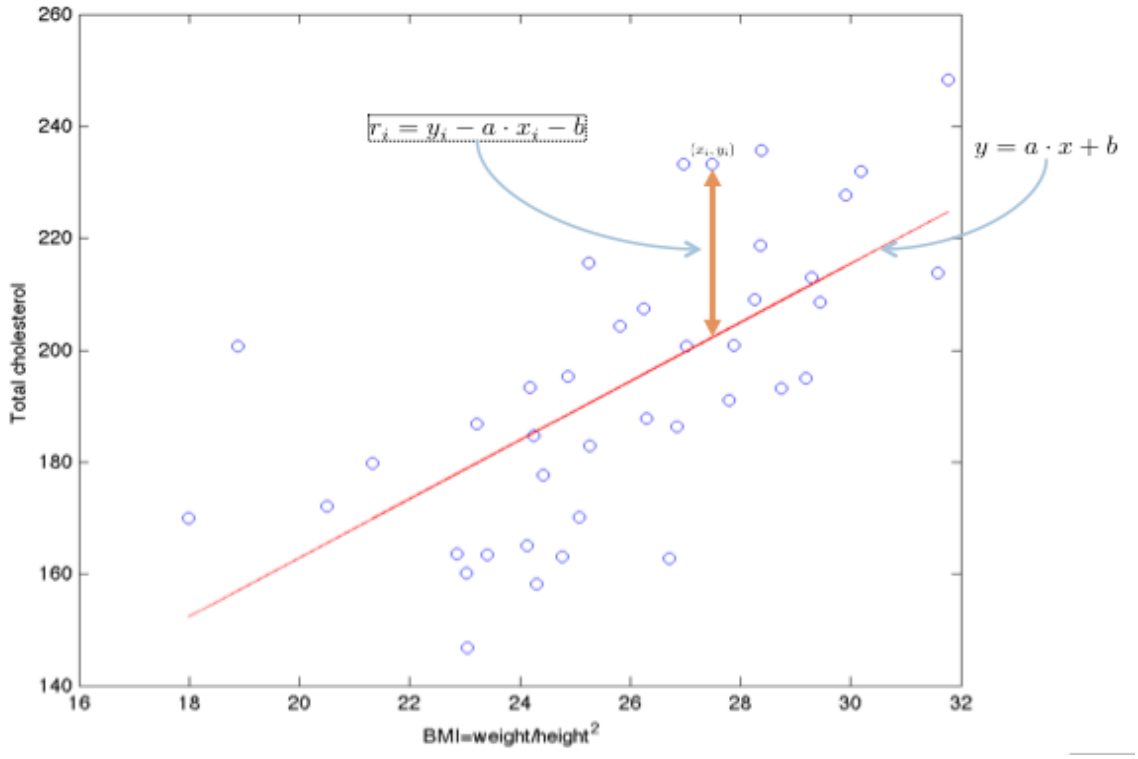


Figure 9.2: Regression line of cholesterol vs. BMI

The regression minimization criterion can be interpreted as a likelihood maximization criterion. We assume a model in which the relation between $x$ and $y$ can be described as

$$y = ax + b + \epsilon,$$

where $a$ and $b$ are unknown constants, and $\epsilon \sim N(0, \sigma^2)$, for some unknown value $\sigma$. Note that under this model the mean of $y$ changes as a function of $x$, however the variance of $y$ is constant, when considered as a function of $x$. This assumption often breaks in practice.

Given a set of training data points $(x_1, y_1), \ldots, (x_n, y_n)$, we can write the log likelihood of the model as

$$\log \mathcal{L}(a, b, \sigma; \{(x_i, y_i)\}) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - ax_i - b)^2 - \frac{n}{2} \log(2\pi\sigma^2)$$

It is easy to see that the likelihood is maximized at the linear regression solution.

## 9.3 The Normal Equations

So far we considered the case where $x \in R$. In general, we can assume that $x \in R^d$, thus it is a $d$-dimensional vector. We will think of $x$ as a row vector. In that case, we can still consider the model

$$y = xa + b + \epsilon,$$

where $\epsilon \sim N(0, \sigma^2)$, and $a \in R^d$ is a column vector. Note that we can increase the dimension of $x$ by adding another variable $x_{d+1} = 1$. Thus, without loss of generality $b = 0$, as it is represented by $a_{d+1}$. From now on, our model of interest is

$$y = xa + \epsilon,$$

where $\epsilon \sim N(0, \sigma^2)$, $a, x \in \mathbb{R}^d$. Our goal is to minimize the following:

$$f(a) = \sum_{i=1}^{n} (y_i - x_i a)^2.$$

In matrix notations, we can denote

$$a = \begin{pmatrix} a_1 \\ \vdots \\ a_d \end{pmatrix}, \quad y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad x_i = (x_{i1}, \ldots, x_{id}), \; X = \begin{pmatrix} - x_1 - \\ \cdots \\ \cdots \\ \cdots \\ - x_n - \end{pmatrix} = \begin{pmatrix} | & \cdots & | \\ X_1 & \cdots & X_d \\ | & \cdots & | \end{pmatrix}.$$

Then, the minimization function can be written as

$$f(a) = (y - Xa)^t (y - Xa)$$

Taking the derivative we get:

$$f'(a) = 2a^t X^t X - 2y^t X$$

Equating the derivative to zero, and after taking the transpose, we get

$$X^t X a = X^t y, \tag{9.1}$$

and therefore we can solve the regression problem analytically as a set of linear equations and get

$$a = (X^t X)^{-1} X^t y.$$

The linear equations described in Equation 9.1 are called the Normal Equations. These equations have a natural geometric interpretation. Since the equations are equivalent to the identity $X^t(y - Xa) = 0$, and since the residuals are defined as $r = y - Xa$, the Normal

Equations require the residuals to be orthogonal to each of the columns of $X$. The columns of $X$, denoted by $X_1, \ldots, X_d$, correspond to the variables used by the regression (e.g., BMI in the examples above). Therefore, the solution of the linear regression is a projection of $y$ onto the subspace spanned by $X_1, \ldots, X_d$ (see Figure 9.3). Algebraically, in the special case where the columns of $X$ are orthogonal ($X_i^t X_j = 0$ for $i \neq j$), we get $a_i = \frac{X_i^t y}{X_i^t X_i}$.

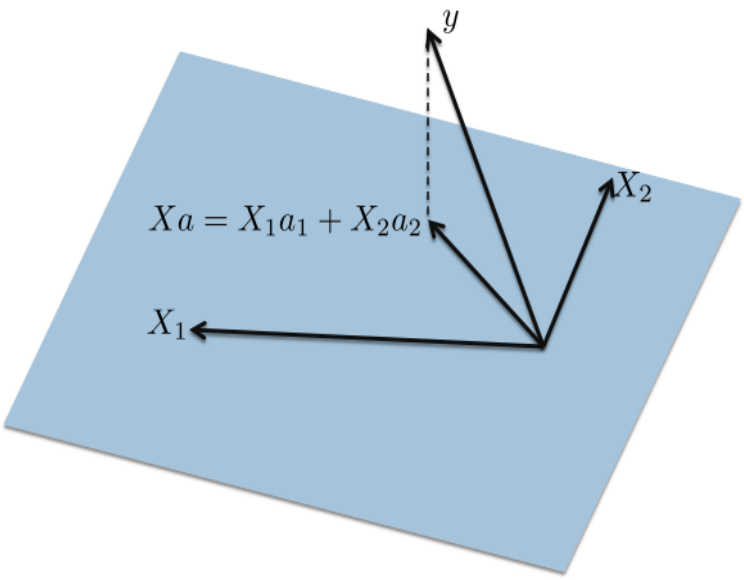


Figure 9.3: Geometric Interpretation of Linear Regression

## 9.4 Gradient Descent

We will take a short break from linear regression and discuss the gradient descent algorithm. We will later show the application of the gradient descent algorithm for linear regression.

Let $f(x_1, \ldots, x_n)$ be a scalar-value function over $n$ variables. The gradient of $f$, denoted by $\nabla f$ is a natural generalization of the derivative of functions over one dimension, and is defined as

$$\nabla f(x) = \left( \frac{\partial f(x)}{\partial x_1}, \ldots, \frac{\partial f(x)}{\partial x_n} \right).$$

The basic idea is that the gradient gives the direction of $f$ in each dimension. Given a minimization problem in which we are interested in minimizing $f(x_1, \ldots, x_n)$, we can use the gradient descent algorithm:

The Gradient Decent Algorithm

Objective: Minimize $f(x_1, \ldots, x_n)$.

1. Start from a point $x^0 = (x_1^0, \ldots, x_n^0)$.
2. Compute the gradient $u^i = \nabla f(x^i)$.
3. Update $x^{i+1} := x^i - \alpha u^i$.
4. Return to step (2) until the algorithm converges.

Thus, the gradient descent algorithm always advances in the direction in which the function changes the most (the largest slope). The initial guess $x^0$ is often chosen randomly from the space, or in some cases it is chosen based on the structure of the problem. The choice of the parameter $\alpha$ is important. A very small $\alpha$ will result in very slow progress, while a large $\alpha$ can 'shoot too far', thus in some cases it will not converge. It is also possible to change $\alpha$ in each iteration of the algorithm, and often the value of $\alpha$ can be optimized in each iteration so that $f(x)$ is minimized in that step. The convergence rule of step (4) may be, for example, that the change in $f(x)$ is smaller than a given parameter $\epsilon$.

In the case of linear regression, the function that we are interested in minimizing is $f(a) = \sum_{i=1}^{n}(y_i - x_i a)^2$. The gradient is $\nabla f(a) = -X^t(y - Xa)$. Therefore, we start from a guess $a_0$, and in each iteration we define $a_{k+1} = a_k + \alpha X^t(y - Xa_k)$. In the case of linear regression, we can even find the best possible value for $\alpha$, since $f(a_{k+1})$ is a quadratic function of $\alpha$.

The gradient descent algorithm is very efficient. Note that $\alpha X^t y$ is fixed across all iterations, and $X^t X$ can be computed once. Then, each iteration costs $O(d^2)$. On the other hand, solving the Normal Equations requires inverting the matrix $X^t X$, which takes $O(d^3)$. In both algorithms the matrix itself needs to be computed in time $O(nd^2)$, so the gradient descent algorithm becomes beneficial when $d$ is large, assuming a small number of iterations will be sufficient. The algorithm is also very simple and can be easily implemented.

**Online Least Squares.** The least squares problem can be viewed as an online learning problem, where we get the samples $(x_i, y_i)$ one at a time. In this case, we start from an arbitrary solution, usually $a_0 = 0$, and then we perform updates based on the gradient of the function restricted to the new sample. Thus,

$$a_{k+1} = a_k + \alpha(y_{k+1} - x_{k+1}a_k)x_{k+1}^t.$$

Clearly, each iteration is highly efficient and costs $O(d)$. The algorithm is similar to the perceptron. In the perceptron, we are also searching for a vector $a$, such that $x_i a$ will be

close to $y \in \{-1, 1\}$. Recall that in the perceptron, if $sign(x_{k+1}a_k) \neq sign(y_{k+1})$ then we set $a_{k+1} = a_k + y_{k+1}x_{k+1}^t$.

## 9.5 Singularity

Recall that the solution to the Normal Equations is $a = (X^tX)^{-1}X^ty$. If $X^tX$ is nonsingular, then there is a unique solution. However, if $X^tX$ is a singular matrix, there is an infinite number of solutions, basically the set of solutions corresponds to the kernel of $X^tX$. Even if $X^tX$ is nonsingular, the solution may be arbitrary. Consider for example the set of points in Figure 9.4. In this case, the value of $z$ is determined by $x, y$, however all the data points collected have $x_i \approx y_i$. Therefore, all we can really say is that the regression solution should correspond to a line, since every plane that intersects with the regression line is a potential solution. In practice, however, a specific solution will be chosen based on the small variance captured in the space orthogonal to that line. If all the points were on the line $y = x$ then the matrix $X^tX$ would be singular and all planes intersecting with the regression line correspond to valid solutions.

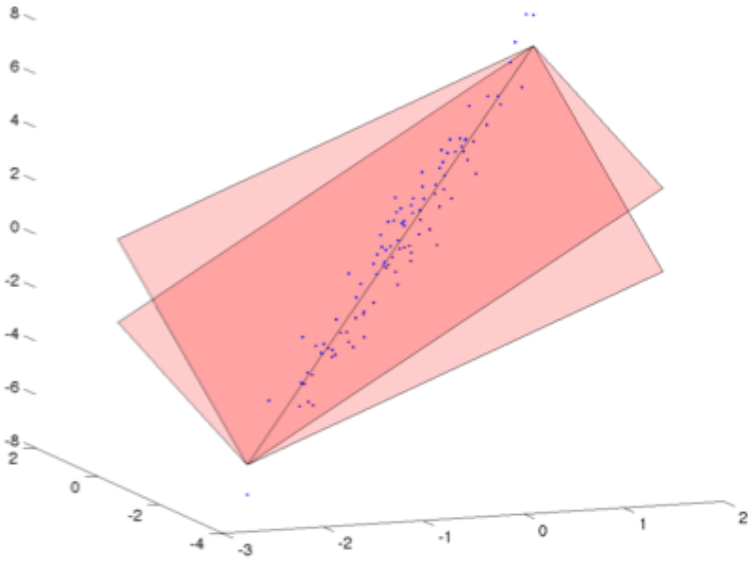


Figure 9.4: An example for a regression in an almost singular scenario

In which cases would we expect to have a nonsingular matrix $X^tX$. Note that the singularity of $X^tX$ depends on its rank. $X^tX$ is a $d \times d$ matrix, and it is singular if its rank is strictly smaller than $d$. Since $rank(X^tX) = rank(X)$, the matrix will be singular if the columns of $X$ are linearly dependent. When will this happen? Typically, when the number

of variables is large (i.e., $d$ is large), we get into situations in which the columns of $X$ are either linearly dependent or close to co-linearity. This often happens since we tend to collect many correlated measurements.

In practice, however, the matrix will typically not be singular since the different measurements taken are noisy, and the noise adds a matrix of full rank to $X$. However, we would not want to predict the noise because we will end up with overfitting. Consider for example the case where $d = n$. This case is extreme, and here you can find $a$ directly by solving $a = X^{-1}y$, and thus $(y - Xa)^t(y - Xa) = 0$. Clearly the solution over-fits here and intuitively we would want to have a small number of variables explaining $y$. In order to resolve this issue we use regularization, as explained in the next section.

## 9.6 Linear Regression with Regularization

Intuitively we want to solve the least squares problem using a small number of variables. We therefore want to penalize our objective function if the number of variables used is large. To do so, we introduced a constant $\lambda$, which is set as an input to our problem. We call $\lambda$ the regularization parameter. We are now interested in solving the following problem:

$$\hat{a} = \arg\min_a \{(y - Xa)^t(y - Xa) + \lambda \|a\|_0\}. \tag{9.2}$$

Here, $\|a\|_0$ is the $L_0$ norm of $a$, which is simply the number of non-zero entries. In what follows we will use the notations $\|a\|_p = \left(\sum_{i=1}^{d} |a_i|^p\right)^{\frac{1}{p}}$ for the $L_p$ norm of a vector (with $p > 0$).

The formulation described in 9.2 involves the choice of a parameter $\lambda$. When $\lambda = 0$ the problem is reduced to the standard linear regression formulation. When $\lambda \to \infty$ it becomes better to set $a_i = 0$ for all $i$. The choice of $\lambda$ can be made using model selection techniques, for instance using cross validation.

Generally, the problem described in 9.2 is NP-hard (if you studied complexity try proving it, it's a good exercise) , meaning that it is probably very difficult, if possible at all, to find a polynomial time algorithm that finds the optimum. Therefore people use different heuristics. For instance, you can start by setting $a_i = 0$ for all entries, and add each time another variable $(a_1, a_2, \ldots, a_d)$. In each iteration you test whether setting $a_i \neq 0$ can improve the current result. Obviously the order in which we traverse the variables may affect the results, and typically one can decide in each iteration to add the variable that provides the best improvement for the objective function. We can also go backwards - start with the solution involving all entries and in each step remove one variable (for instance the one with the smallest coefficient in terms of absolute value), as long as the objective function increases. These algorithms are variants of the forward and backward stepwise regression algorithm.

# 9.7 Shrinkage Methods

Since the $L_0$ formulation is hard, we can consider other approximations. Two of the most widely used approximations are based on the $L_1$ and $L_2$ approximation. We will first describe the $L_2$ approximation, termed Ridge regression.

## 9.7.1 Ridge Regression

In Ridge regression we are interested in optimizing the following for a specified parameter $\lambda$:

$$\hat{a} = \arg\min_a \{(y - Xa)^t(y - Xa) + \lambda \|a\|_2^2\}. \tag{9.3}$$

We will now try to generalize the Normal Equations to the case of Ridge regression. Note that the optimization function can be written as

$$\begin{aligned} f(a) &= (y - Xa)^t(y - Xa) + \lambda a^t a \\ &= y^t y - a^t X^t y - y^t X a + a^t X^t X a + \lambda a^t a \end{aligned}$$

Therefore,

$$\frac{\partial f(a)}{\partial a} = 2(X^t X + \lambda I)a - 2X^t y$$

and we get that the solution to the Ridge regression is given by

$$\hat{a} = (X^t X + \lambda I)^{-1} X^t y. \tag{9.4}$$

The solution given by Equation 9.4 has a few interesting properties. First, it is easy to see that when $\lambda = 0$ we get the Normal equations, which is what we should expect. Second, note that the matrix $X^t X + \lambda I$ is positive definite and therefore non-singular for $\lambda > 0$. Therefore Ridge regression can be viewed as a numerical correction to the standard linear regression - by adding a small positive $\lambda$ times the identity to $X^t X$ we implicitly change the set of columns of $X$ so that it has full rank and is far from singular.

The motivation for applying Ridge regression goes beyond the numerical issues. We can view Ridge regression as a Bayesian generalization of the standard linear regression. Consider the following model:

$$y = \sum_{j=1}^{d} a_j X_j + \epsilon,$$

where $X_j$ is the $j - th$ variable of the regression, and we assume $\epsilon \sim N(0, \sigma^2)$ where $\sigma$ is unknown. This is the standard regression model. In Ridge regression, we use a hierarchical model in which we assume that the coefficients are not fixed but instead they are randomly

picked from a normal prior, i.e., $a_j \sim N(0, \tau^2)$. If we now write down the log of the posterior under the assumption that $\sigma, \tau$ are known (ignoring the normalizing constant), we get:

$$\begin{aligned} \log Posterior(a \mid \sigma, \tau, Data) &= -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - a \cdot x_i)^2 - \frac{1}{2\tau^2}\sum_{i=1}^{d} a_i^2 \\ &\quad -\frac{n}{2}\log(2\pi\sigma^2) - \frac{d}{2}\log(2\pi\tau^2) \end{aligned}$$

Therefore, maximizing the posterior is equivalent to solving Ridge regression with $\lambda = \frac{\sigma^2}{\tau^2}$. This is a very natural Bayesian extension of standard linear regression. Clearly more complicated Bayesian priors could be introduced, for example the prior of $a$ could be a multivariate normal distribution and in this case we will obtain a similar analysis.

### 9.7.2 Lasso Regression

In Lasso regression we are interested in optimizing the following for a specified parameter $\lambda$:

$$\hat{a} = \arg\min_a\{(y - Xa)^t(y - Xa) + \lambda\|a\|_1\}. \tag{9.5}$$

Note that the only difference between Lasso and Ridge is that the penalty is an $L_1$ penalty instead of $L_2$. Simplifying the Lasso minimization problem we get an equivalent formulation:

$$\hat{a} = \arg\min\{a^t(X^tX)a - 2y^tXa + \lambda\sum_{i=1}^{d}|a_i|\}$$

This in turn can be written in the following equivalent form, as a constraint optimization problem:

$$\begin{aligned} & min_a\{a^t(X^tX)a - 2y^tXa + \lambda\sum_{i=1}^{d} b_i\} \\ s.t.\quad & b_i \geq a_i, \qquad i = 1, \ldots, d \\ & b_i \geq -a_i, \qquad i = 1, \ldots, d \end{aligned}$$

This constraint optimization program is a quadratic program, with a convex optimization function, since $X^tX$ is positive semidefinite. Recall that in quadratic programming the minimum is unique and can be reached in polynomial time when the objective is a combination of a linear function and a positive definite quadratic term. This makes the Lasso regression an attractive approach since it can be efficiently calculated.

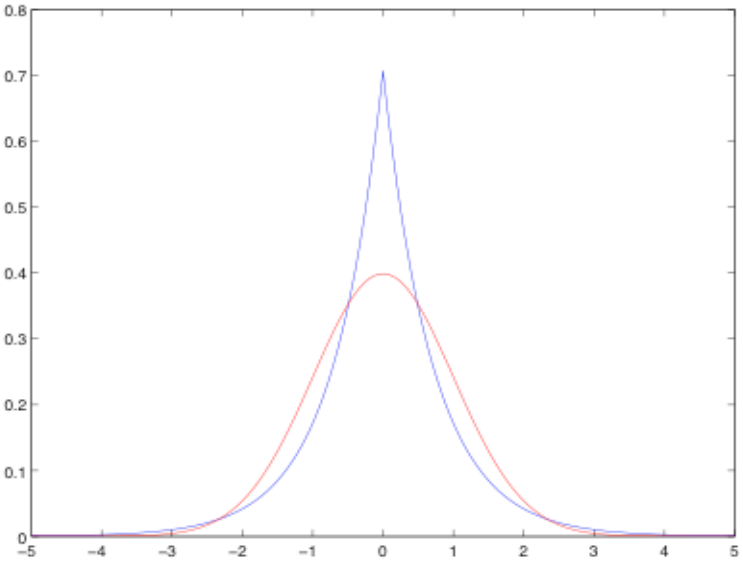


Figure 9.5: The Normal vs. Laplace priors (mean 0, variance 1)

Similarly to Ridge regression, also here we can have a Bayesian model that motivates the choice of Lasso. Consider the hirearchical model in which the coefficients $a_j$ are sampled from a Laplace distribution, $a_j \sim Laplace(0, \frac{2\sigma^2}{\lambda})$. The Laplace distribution, sometimes referred to as the double exponential distribution, has the following properties. The probability density function (pdf) of the Laplace distribution $Laplace(\mu, \sigma)$ is

$$pdf(x) = \frac{1}{2\sigma} e^{-\frac{|x-\mu|}{\sigma}}$$

The mean of the Laplace distribution is $\mu$ and the variance is $2\sigma^2$. Thus, in the model where $a_j \sim Laplace(0, \frac{2\sigma^2}{\lambda})$, we get

$$pdf(a_j) = \frac{\lambda}{4\sigma^2} e^{-\frac{\lambda}{2\sigma}|a_j|}$$

Therefore, given $\lambda, \sigma$, we can write down the log of the posterior (ignoring the normalizing factor):

$$\begin{aligned} \log Posterior(a \mid Data, \lambda, \sigma) &= -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - a \cdot x_i)^2 - \frac{n}{2} \log(2\pi\sigma^2) \\ &\quad + d \log\left(\frac{\lambda}{4\sigma^2}\right) - \frac{\lambda}{2\sigma^2} \sum_{i=1}^{d} |a_i| \end{aligned}$$

It is easy to verify that maximizing the a posteriori is equivalent to minimizing the Lasso minimization function.

The difference between Lasso and Ridge can therefore be viewed as a difference in the assumption on the priors used (see Figure 9.5). There is an interesting geometric interpretation for this difference. We can introduce the following constraint optimization program:

$$\begin{aligned} & min_a (y - Xa)^t (y - Xa) \\ s.t. \quad & \sum_{i=1}^{d} |a_i| \leq \lambda' \end{aligned}$$

We claim that for every $\lambda$ we can find $\lambda'$ such that the solution of Lasso and the above program is the same. Consider the solution $\hat{a}$ of the original Lasso optimization. Let $\lambda' = \sum_{i=1}^{d} |\hat{a}_i|$. Clearly, $\hat{a}$ is also a solution of the above program with $\lambda'$. If there is a better solution $a'$, it is easy to see that $a'$ will also provide a better solution for the Lasso optimization. Similarly, we can show that for every $\lambda$ there is $\lambda'$ so that the Ridge solution with the parameter $\lambda$ is obtained as the solution for the following constraint optimization program:

$$\begin{aligned} & min_a (y - Xa)^t (y - Xa) \\ s.t. \quad & \sum_{i=1}^{d} a_i^2 \leq \lambda' \end{aligned}$$

Thus, Lasso and Ridge correspond to the constraint optimization problem in which we are interested in minimizing the least squares of the residuals, so that the vector of coefficients is within a ball of radius $\lambda'$ (the ball is an $L_1$ ball in the case of Lasso, and an $L_2$ ball in the case of Ridge). The geometric interpretation of this view is given in Figures 9.6 and 9.7. The best result of the linear regression, denoted by $\hat{a}_{regression}$ is outside the ball. The counter lines, in which the value of the least squares is the same across the line, hit the ball at a specific point, which is the Ridge or Lasso solution. One of the advantages of Lasso is that the ball has a square-like form, and therefore the optimal solution is usually obtained in a vertex. In such cases, many of the entries are zero, and it is therefore usually a better approximation to the $L_0$ optimization criterion. In the example of Figure 9.7, the value of $a_1$ is zero.

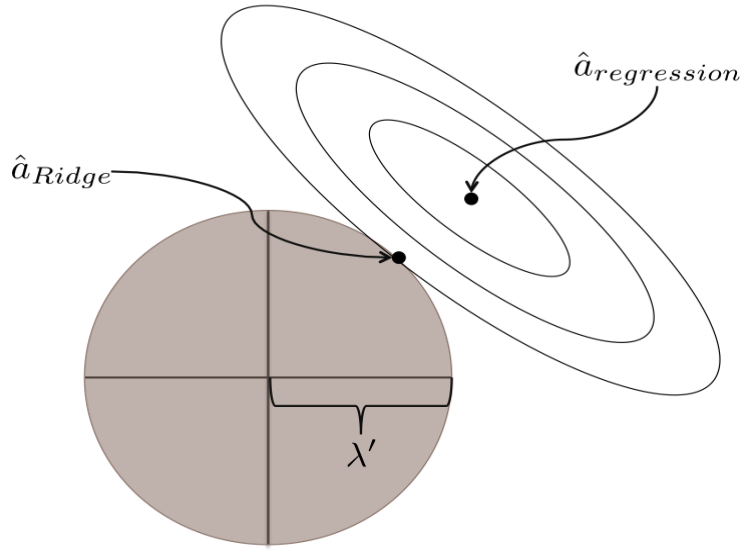


Figure 9.6: Geometric interpretation of Ridge regression

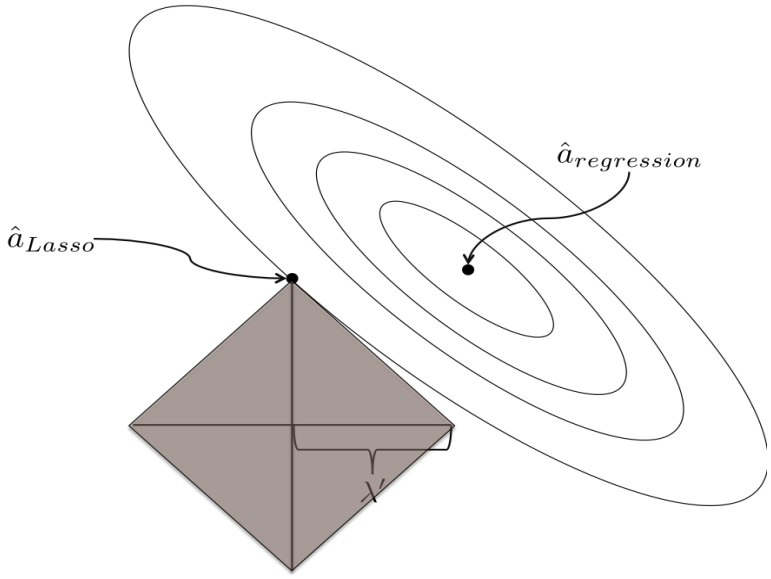


Figure 9.7: Geometric interpretation of Lasso regression