Lecture 3

Constantine Caramanis, Alex Dimakis
The University of Texas at Austin
Department of Electrical and Computer Engineering

1 Introduction

The previous two lectures have developed the key idea of Empirical Risk Minimization for classification, and we saw how to use this to find the "best" stump. For the classification problem, we were trying to find a labeling (or prediction) rule, that would use the height and width of a chip, to predict whether it would explode or not. Consider now a related problem: instead of trying to predict whether a specific chip with those measurements will explode, suppose we want to predict *the average time* that a chip will function before exploding. Thus, our data set now might look like

	height	**width**	**y=avg. time to explode**
chip 1	0.8	0.8	1.8
chip 2	0.3	0.25	0.45
chip 3	0.2	0.8	0.95
chip 4	0.3	0.7	1.02
chip 5	0.9	0.7	1.76

Table 1: Your dataset. As with the previous example, we have a special column (called y) that we are trying to predict using the other columns called features. Note, however, that unlike the previous example where we were trying to predict *categorical labels* $y \in \{0,1\}$ (explode/not explode), we are now trying to predict a *numerical quantity*: average time to explode. Every row corresponds to the size of a particular chip, and the label corresponds to the average time to explode. Recall that each row of this table corresponds to a sample. Thus we have $n = 4$ samples, $p = 2$-dimensional features.

We can again use the idea of Empirical Risk Minimization (ERM), to find a good labeling rule. Once we choose a loss function, $\ell(\cdot,\cdot)$, and a class of models, $\mathcal{H}$, our ERM looks very similar to what we had before: We want to find the $h \in \mathcal{H}$ that solves the problem:

$$\begin{aligned} \text{min}: &\quad \frac{1}{n}\sum_i \ell(h(\boldsymbol{x}_i), y_i) \\ \text{s.t.}: &\quad h \in \mathcal{H}. \end{aligned}$$

Thus we have described the Regression problem, and ERM for Regression. *Regression is the supervised learning task that seeks to find a rule, or mapping, from features to a numerical, or continuous label, y.*

In this case, two things are different: the labeling rules we choose from need to predict numerical labels, and not just 0 or 1. Second, we need a different loss function. For a 0/1 prediction problem, a prediction is either correct, or it is wrong. Here, though, what if our rule labels a chip of height and width $(0.3, 0.7)$ as 0.98? Certainly, $0.98 \neq 1.02$ which is the given label, but it is not too far off. Moreover, if another rule predicts 0.88 for that same chip, while both rules are "wrong" for this particular data point, the second labeling rule is "more wrong." We need a loss function that captures this.

As we did with classification, we note that if we choose $\mathcal{H}$ to be large enough, then we can always solve the ERM problem to make the training error zero. One way to do this is by a model h that memorizes the dataset $\mathcal{S}$ and produces labels as follows:

Stupid Memorization Model h_m

• For a given input $\mathbf{x}$, if the same feature vector $\mathbf{x}$is in the training set, output the training label as a prediction: $h_m(\mathbf{x}) = y$.
• For a given input $\mathbf{x}$that is not in the training set, make the prediction $h_m(\mathbf{x}) = -100$

Note that this memorization rule is so stupid, that it fails to even make physical sense: on any features that it has not seen before, it predicts that the time until the chip explodes is *negative*. Nevertheless, it attains the smallest possible loss on the training data, namely, zero. As with classification, therefore, we need to limit our set of rules, if we hope that ERM will give us something that *generalizes well to data that our model has not yet seen*. In other words, we need to limit $\mathcal{H}$ if we hope to obtain something with true predictive power from ERM.

2 Squared Error

One of the most often used loss functions is squared loss. In this case:

$$\ell(h(\boldsymbol{x}_i), y_i) = (h(\boldsymbol{x}_i) - y_i)^2.$$

Let's start with the most simple set of functions $\mathcal{H}$: the set of constants. That is, we limit our prediction to that most simple set of rules that always predict the same constant, regardless of what the features are. Not too clever, but still, a place to start. What constant should we choose? Using this choice of $\mathcal{H}$, and the squared error loss function, our ERM problem becomes:

$$\begin{aligned} \text{min} : &\quad \frac{1}{n}\sum_i (\beta_0 - y_i)^2 \\ \text{s.t.} : &\quad \beta_0 \in \mathbb{R}. \end{aligned}$$

We can find the best value, β_0^* by differentiating the loss function with respect to β_0 and setting equal to zero, and then solving for β_0:

$$\frac{d}{d\beta_0}\left(\frac{1}{n}\sum_i (\beta_0 - y_i)^2\right) = \frac{1}{n}\sum_i \frac{d}{d\beta_0}(\beta_0 - y_i)^2 = \frac{2}{n}\sum_i (\beta_0 - y_i).$$

Setting this equal to zero we find:

$$\beta_0^* = \frac{1}{n}\sum y_i.$$

That is, the β_0 that minimizes the squared error is the mean value of the $\{y_i\}$. For our data set, therefore, we have

$$\beta_0 = \frac{1}{5}(1.8 + 0.45 + 0.95 + 1.02 + 1.76) = 1.196.$$

The total empirical risk of this is therefore:

$$\begin{aligned} ERM &= \frac{1}{5}\left((1.8-1.196)^2 + (0.45-1.196)^2 + (0.95-1.196)^2 + (1.02-1.196)^2 + (1.76-1.196)^2\right) \\ &= 0.266184. \end{aligned}$$

Exercise 1

If we change the loss function, then the answer may change as well. Instead of squared error, $\ell(\hat{y}, y) = (\hat{y} - y)^2$, consider the absolute value of the error: $\ell(\hat{y}, y) = |\hat{y} - y|$. Therefore the ERM problem becomes:

$$\begin{aligned} \text{min}: &\quad \frac{1}{n}\sum_i |\beta_0 - y_i| \\ \text{s.t.}: &\quad \beta_0 \in \mathbb{R}. \end{aligned}$$

• Compute the empirical risk of this model on our dataset. The answer is not the mean, though it is very very close in this case.
• Can you find the general answer? In other words, if we have data $\{(\boldsymbol{x}_i, y_i)\}$, $i = 1, \ldots, n$, then what is the solution to the above problem?

3 Linear Regression

In the previous section and example, we considered the squared loss, and prediction only by constant functions. That is, we looked at the class of labeling rules of the form:

$$h(\boldsymbol{x}) = h(x_1, x_2; \beta_0) = \beta_0.$$

Note that we have changed, or rather *augmented*, our notation. While indeed each labeling rule h is a function of the feature vector, $\boldsymbol{x} = (x_1, x_2)$, we also want to make explicit its dependence on the coefficients that define the function. In the case above, there is a single coefficient, β_0, that defines h.

Let's now consider rules that are linear. This means that they have a linear dependence on each of the features:

$$h_{\text{linear}}(\boldsymbol{x}; \boldsymbol{\beta}) = h_{\text{linear}}(x_1, x_2; (\beta_0, \beta_1, \beta_2)) = \beta_0 + \beta_1 x_1 + \beta_2 x_2.$$

Using this class of rules, our ERM problem now becomes:

$$\begin{aligned} \min: \quad & \frac{1}{n}\sum_i (\beta_0 + \beta_1 x_1^{(i)} + \beta_2 x_2^{(i)} - y_i)^2 \\ \text{s.t.}: \quad & \beta_0, \beta_1, \beta_2 \in \mathbb{R}. \end{aligned}$$

4 Linear Regression: One Variable

Before we solve this general problem, let's consider a simpler version where we want to find the best (in terms of least square error) rule of the form:

$$h(\boldsymbol{x}; \beta_1) = h(x_1, x_2; \beta_1) = \beta_1 x_1,$$

i.e., we need to solve

$$\begin{aligned} \min: \quad & \frac{1}{n}\sum_i (\beta_1 x_1^{(i)} - y_i)^2 \\ \text{s.t.}: \quad & \beta_1 \in \mathbb{R}. \end{aligned}$$

This is again a problem in a single variable, so we can solve it using the same ideas as in the constant case: we take the derivative with respect to β_1, set it to zero and solve for β_1:

$$\frac{d}{d\beta_1}\left(\frac{1}{n}\sum_i (\beta_1 x_1^{(i)} - y_i)^2\right) = \frac{1}{n}\sum_i \frac{d}{d\beta_1}(\beta_1 x_1^{(i)} - y_i)^2 = \frac{2}{n}\sum_i x_1^{(i)}(\beta_1 x_1^{(i)} - y_i).$$

Setting this equal to zero we find:

$$\begin{aligned} & \sum_i x_1^{(i)}(\beta_1 x_1^{(i)} - y_i) = 0 \\ \Leftrightarrow \quad & \beta_1 \sum_i (x_1^{(i)})^2 - \sum_i x_1^{(i)} y_i = 0 \\ \Leftrightarrow \quad & \beta_1 \sum_i (x_1^{(i)})^2 = \sum_i x_1^{(i)} y_i \\ \Leftrightarrow \quad & \beta_1 = \frac{\sum_i (x_1^{(i)}) y_i}{\sum_i x_1^{(i)}}. \end{aligned}$$

Projection of y onto x1

For our data set, therefore, we have:

$$\beta_1^* = \frac{0.8 \cdot 1.8 + 0.3 \cdot 0.45 + 0.2 \cdot 0.95 + 0.3 \cdot 1.02 + 0.9 \cdot 1.76}{0.8^2 + 0.3^2 + 0.2^2 + 0.3^2 + 0.9^2} = \frac{3.655}{1.67} = 2.1886.$$

5 Linear Regression: Multiple Variables

The key idea in the case of the constant rule, and the single-variable linear rule, was the same: write down the ERM loss, take the derivative, and set it equal to zero. In both cases this was easy because the ERM loss was a function of a single variable – β_0 for the constant case, and β_1 for the linear case. For the general problem, we need to generalize our intuition from the single variable case. Fortunately, while we have to track notation a little more carefully, and visualization proves somewhat more challenging, conceptually the ideas are precisely the same. We illustrate the idea in Figure 1

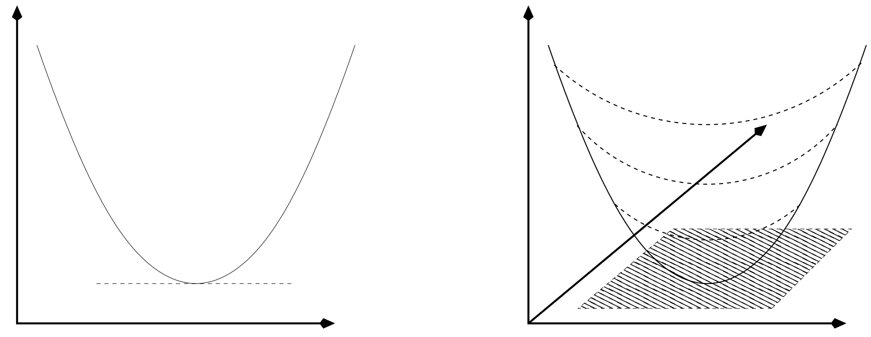

Figure 1: The first figure shows a function of one variable: $f(\beta)$. The minimum is attained at the point where the slope of the function is zero, i.e., the point where $f'(\beta) = 0$. The second figure shows a function of two variables: $f(\boldsymbol{\beta_1}, \boldsymbol{\beta_2}) = f(\beta_1, \beta_2)$. The minimum is attained when the derivatives in each direction are zero, i.e., when *both partial derivatives are equal to zero*: $\partial f(\boldsymbol{\beta})/\partial \beta_1 = 0$, and $\partial f(\boldsymbol{\beta})/\partial \beta_2 = 0$.

As shown in the figure, the minimum of a (convex) function is attained at a point where all partial derivatives are equal to zero. ***Note that we are applying these ideas to the entire ERM problem, i.e., the sum of the loss function over all the training data; and we are looking at this as a function of*** β_0***,*** β_1 ***and*** β_2***, and taking derivatives with respect to these, and not with respect to*** x_1 ***and*** x_2. In the case we have now, we therefore have to find $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2)$ that satisfy three equations, for the function

$$L(\boldsymbol{\beta}) = \frac{1}{n} \sum_i \ell(h(\boldsymbol{x}_i; \boldsymbol{\beta}), y_i).$$

Thus we have:

$$\frac{\partial L(\boldsymbol{\beta})}{\partial \beta_0} = 0, \quad \frac{\partial L(\boldsymbol{\beta})}{\partial \beta_1} = 0, \quad \frac{\partial L(\boldsymbol{\beta})}{\partial \beta_2} = 0.$$

Recall that the *gradient of a function* is the vector of its partial derivatives. Thus:

$$\nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}) = \begin{pmatrix} \frac{\partial L(\boldsymbol{\beta})}{\partial \beta_0} \\ \frac{\partial L(\boldsymbol{\beta})}{\partial \beta_1} \\ \frac{\partial L(\boldsymbol{\beta})}{\partial \beta_2} \end{pmatrix}.$$

Thus, in order to find an optimal solution $= (\beta_0, \beta_1, \beta_2)$ to the problem above, the analog to solving the single equation $dL(\beta_0)/d\beta_0$ for the single variable case, is to solve the system of three equations and three unknowns given by:

$$\nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}) = 0.$$

Let's apply this to our example above, and to our data. Recall that we are using squared loss, and our labeling rule h is linear. Therefore:

$$L(\boldsymbol{\beta}) = \frac{1}{n}\sum_i (\beta_0 + \beta_1 x_1^{(i)} + \beta_2 x_2^{(i)} - y_i)^2.$$

The three equations now read:

$$\begin{aligned}
\frac{\partial L(\boldsymbol{\beta})}{\partial \beta_0} &= \frac{2}{n}\sum_i (\beta_0 + \beta_1 x_1^{(i)} + \beta_2 x_2^{(i)} - y_i) = 0,\\
\frac{\partial L(\boldsymbol{\beta})}{\partial \beta_1} &= \frac{2}{n}\sum_i x_1^{(i)}(\beta_0 + \beta_1 x_1^{(i)} + \beta_2 x_2^{(i)} - y_i) = 0,\\
\frac{\partial L(\boldsymbol{\beta})}{\partial \beta_2} &= \frac{2}{n}\sum_i x_2^{(i)}(\beta_0 + \beta_1 x_1^{(i)} + \beta_2 x_2^{(i)} - y_i) = 0.
\end{aligned}$$

Note that while the second and third equations are not linear in x_1 and x_2, this is not of a concern to us because we are not solving for x_1 and x_2 – these are the data! We are solving for $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2)$, and all three equations are linear in these three variables. Therefore we have a linear system of three equations and three unknowns.

Exercise 2

Multiply out the expression above, using the numbers from our example.

- Show that the three equations we get are:

$$\begin{aligned}
5\beta_0 + 2.5\beta_1 + 3.25\beta_2 &= 5.98\\
2.5\beta_0 + 1.67\beta_1 + 1.715\beta_2 &= 3.655\\
3.25\beta_0 + 1.715\beta_1 + 2.3225\beta_2 &= 4.2585.
\end{aligned}$$

- Show that the solution to this system of three equations and three unknowns is

$$\boldsymbol{\beta}^* = \begin{pmatrix} -0.24750187 \\ 1.32602996 \\ 1.20074906 \end{pmatrix}.$$

6 Linear Regression – Multiple Variables, Take Two

If you do the exercise above, you will ask yourself why anybody bothers to solve problems with more than one variable, because it seems like quite a chore to obtain the above expression; we have to multiply out, collect terms, and so on. We now take a more direct approach, and show that we can obtain the solution much more quickly.

Let X_0 denote the data matrix of features, so for us,

$$X_0 = \begin{bmatrix} 0.8 & 0.8 \\ 0.3 & 0.25 \\ 0.2 & 0.8 \\ 0.3 & 0.7 \\ 0.9 & 0.7 \end{bmatrix}$$

Next, we augment X_0 by adding a vector one 1's as the first column:

$$X = \begin{bmatrix} 1 & 0.8 & 0.8 \\ 1 & 0.3 & 0.25 \\ 1 & 0.2 & 0.8 \\ 1 & 0.3 & 0.7 \\ 1 & 0.9 & 0.7 \end{bmatrix}$$

We can now write the squared error loss function much more compactly. For this, we recall that the Euclidean norm of a vector is the square root of the sum of squares of its coefficients:

$$\|\boldsymbol{z}\|_2 = \sqrt{z_1^2 + \cdots + z_n^2} = \sqrt{\boldsymbol{z}^\top \boldsymbol{z}}.$$

Then, using

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix},$$

our loss function becomes:

$$\begin{aligned} L(\boldsymbol{\beta}) &= \frac{1}{n}\sum_i (\beta_0 + \beta_1 x_1^{(i)} + \beta_2 x_2^{(i)} - y_i)^2 \\ &= \|X\boldsymbol{\beta} - \boldsymbol{y}\|_2^2. \end{aligned}$$

The next two exercises walk us through the steps to obtain the solution to the general linear least squares problem with multiple variables.

Exercise 3

Using the formula $\|z\|_2^2 = \boldsymbol{z}^\top \boldsymbol{z}$, expand $\|X\boldsymbol{\beta} - \boldsymbol{y}\|_2^2$.

Exercise 4

Use the expansion of $\|X\boldsymbol{\beta} - \boldsymbol{y}\|_2^2$ from the previous exercise, and the facts that:

$$\begin{aligned}
\nabla_{\beta}\boldsymbol{\beta}^\top X^\top X\boldsymbol{\beta} &= 2X^\top X\boldsymbol{\beta} \\
\nabla_{\beta}\boldsymbol{\beta}^\top X^\top\boldsymbol{y} &= X^\top\boldsymbol{y} \\
\nabla_{\beta}\boldsymbol{y}^\top\boldsymbol{y} &= 0,
\end{aligned}$$

to conclude that

$$\nabla_{\beta}\|X\boldsymbol{\beta} - \boldsymbol{y}\|_2^2 = 2X^\top X\boldsymbol{\beta} - 2X^\top\boldsymbol{y},$$

and hence that the optimal solution is given by setting the above expression to zero and solving for $\boldsymbol{\beta}$, i.e.,

$$\boldsymbol{\beta}^* = (X^\top X)^{-1}X^\top\boldsymbol{y}.$$

The two exercises above give us an important result in linear least squares regression: we obtain a *closed form expression* for the least squares solution:

$$\boldsymbol{\beta}^* = (X^\top X)^{-1}X^\top\boldsymbol{y}.$$

Recall that to implement this in Python, we first need to augment the data matrix of features, by adding a new "feature" that is all 1's. This new augmented matrix is the matrix X used above. The solution given above is sometimes denoted by $\boldsymbol{\beta}_{\text{ls}}$ for "least squares," or $\boldsymbol{\beta}_{\text{ols}}$ for "ordinary least squares," to allow for some more exotic versions of least squares regression that follow.

Exercise 5

Using this formula to obtain the optimal solution for our example. Check that you obtain the same result as what is given in the exercise from the previous section above.

7 Polynomial Regression

Suppose now, that our data are given by the following table:

A careful inspection shows that there is a very accurate (zero error, in fact) labeling rule, but it is not linear:

$$h(\boldsymbol{x}) = x_1^2.$$

	height	width	y=avg. time to explode
chip 1	0.8	0.8	0.64
chip 2	0.3	0.25	0.09
chip 3	0.2	0.8	0.04
chip 4	0.3	0.7	0.09
chip 5	0.9	0.7	0.81

Table 2: Your new dataset. Note that the features are unchanged, but the labels given are different.

By design, our linear least squares regression can never recover a solution of this form. If either by domain knowledge or perhaps by visualizing the data we realize that a linear fit does not seem appropriate, how can we proceed, and how can we use the ideas above?

It turns out that we can use the ideas for linear regression, exactly as outlined above, as long as we take one important pre-processing step: feature generation. Currently, a data point has the form $(\mathbf{x}, y) = (x_1, x_2, y)$. Suppose that we believe that a good labeling rule should depend not only on x_1 and x_2, but also on x_1^2. In other words, we believe that a good labeling rule should be of the form:

$$h(\boldsymbol{x};\boldsymbol{\beta}) = h(\boldsymbol{x};\beta_0,\beta_1,\beta_2,\beta_3) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2.$$

Note that this is again a linear decision rule! Therefore all we need is to appropriately augment our matrix X, and then we can proceed exactly as above. As before, we begin with our 5×2 matrix X_0. We augment this with the column of all 1′s. We now need to add a fourth column, that contains the squared values of x_1. Thus we have:

$$X = \begin{bmatrix} 1 & 0.8 & 0.8 & 0.64 \\ 1 & 0.3 & 0.25 & 0.09 \\ 1 & 0.2 & 0.8 & 0.04 \\ 1 & 0.3 & 0.7 & 0.09 \\ 1 & 0.9 & 0.7 & 0.81 \end{bmatrix}, \qquad \boldsymbol{y} = \begin{pmatrix} 0.64 \\ 0.09 \\ 0.04 \\ 0.09 \\ 0.81 \end{pmatrix}.$$

Exercise 6

Use the results from the previous section to compute the optimal decision rule of the form

$$h(\boldsymbol{x};\boldsymbol{\beta}) = h(\boldsymbol{x};\beta_0,\beta_1,\beta_2,\beta_3) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2.$$

The answer, as we can easily verify by inspection, is

$$h(\boldsymbol{x};\boldsymbol{\beta}) = h(\boldsymbol{x};\beta_0,\beta_1,\beta_2,\beta_3) = 0 + 0 \cdot x_1 + 0 \cdot x_2 + x_1^2,$$

but the goal of this exercise is to understand how the solution to the multiple-variable linear regression problem applies here.

Now we can extrapolate: fitting a polynomial to data is as easy as adding columns to our data matrix, corresponding to the monomials we want to use. In the example above, we added the monomial x_1^2. By a precisely similar approach, we could add x_2^2, or $x_1 x_2$, or $x_1 x_2^2$, etc.

As the next exercise demonstrates, we have to take care not to overfit. If we add too many terms, the expressive power of $\mathcal{H}$ becomes too great, and we can fit anything, resulting in rules that are overfitting the data instead of actually obtaining rules with valuable/useful predictive power.

Exercise 7

You will now fit functions of the form:

$$h(\boldsymbol{x};\boldsymbol{\beta}) = h(\boldsymbol{x};\beta_0,\beta_1,\beta_2,\beta_3\beta_4) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4^2.$$

To do this, take the data matrix X_0 above, and add the all 1's column, as well as columns for x_1^2 and x_2^2. Make up your own data for $\boldsymbol{y}$, and apply the above procedure to find the best fit. You should find that no matter what you chose for $\boldsymbol{y}$, you are able to fit with zero error.

Note what is happening in Exercise 7 above: you have five elements in your data set, and you also have five degrees of freedom: $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \beta_3, \beta_4)$. Typically, we want to have many more data points than degrees of freedom, otherwise we run the risk of overfitting, and may obtain decision rules that have no predictive power.