Week 3: Linear Regression

Instructor: Sergey Levine

1 The regression problem

We saw how we can estimate the parameters of probability distributions over a random variable x. However, in a supervised learning setting, we might be interested in *predicting* the value of some output variable y. For example, we might like to predict the salaries that CSE 446 students will receive when they graduate. In order to make an accurate prediction, we need some information about the students: we need some set of *features*. For example, we could try to predict the salaries that students will receive based on the grades they got on each homework assignment. Perhaps some assignments are more important to complete than others, or more accurately reflect the kinds of skills that employers look for. This kind of problem can be framed as *regression*.

Question. What is the data?

Answer. Like with decision trees, the data consists of tuples $(\mathbf{x}_i, y_i)$. Except now, $y \in \mathbb{R}$ is continuous, as are all of the attributes (features) in the vector $\mathbf{x}$. The dataset is given by $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$.

Question. What is the hypothesis space?

Answer. This is a design choice. A simple and often very powerful hypothesis space consists of *linear* functions on the feature vector $\mathbf{x}_i$, given by $f(\mathbf{x}_i) = \sum_{j=1}^{d} \mathbf{w}_j \mathbf{x}_{i,j} = \mathbf{x}_i \cdot \mathbf{w}$. The parameters of this hypothesis are the weights $\mathbf{w} \in \mathbb{R}^d$.

Question. What is the objective?

Answer. This is also a design choice. Intuitively, we would like $f(\mathbf{x}_i)$ to be "close" to y_i, so we can write our objective as:

$$\hat{\mathbf{w}} \leftarrow \arg\min_{\mathbf{w}} \sum_{i=1}^{N} D(f(\mathbf{x}_i), y_i),$$

where $D(a, b)$ is some measure of distance. Which distance measure to use? Well, a popular choice is to simply use the ℓ_2 norm (squared error), given by

$D(f(\mathbf{x}_i), y_i) = (f(\mathbf{x}_i) - y_i)^2$. This gives us the following objective:

$$\hat{\mathbf{w}} \leftarrow \arg\min_{\mathbf{w}} \sum_{i=1}^{N} (f(\mathbf{x}_i) - y_i)^2,$$

Given our definition of $f(\mathbf{x}_i)$, this is simply

$$\hat{\mathbf{w}} \leftarrow \arg\min_{\mathbf{w}} \sum_{i=1}^{N} (\mathbf{x}_i \cdot \mathbf{w} - y_i)^2,$$

As we will see later in the lecture, this choice of distance function corresponds to a kind of probabilistic model, which turns regression into a MLE problem.

Question. What is the algorithm?

Answer. Just like in the MLE lectures, we can derive the optimal $\hat{\mathbf{w}}$ by taking the derivative of the objective and setting it to zero. Note that we take the derivative with respect to a vector quantity $\mathbf{w}$ (this is also called the gradient):

$$\frac{d}{d\mathbf{w}} \sum_{i=1}^{N} (\mathbf{x}_i \cdot \mathbf{w} - y_i)^2 = \sum_{i=1}^{N} \mathbf{x}_i (\mathbf{x}_i \cdot \mathbf{w} - y_i) = 0.$$

So the gradient consists of the sum over all datapoints of the (signed) error (also called the residual), times the feature vector of that datapoint $\mathbf{x}_i$. To make it convenient to derive the solution $\mathbf{w}$, we can rearrange this in matrix notation, by defining

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \dots \\ \mathbf{x}_N^T \end{bmatrix} \qquad \mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{bmatrix}$$

Then we can see that the summation in the gradient equation above can be equivalently expressed as a matrix product:

$$\sum_{i=1}^{N} \mathbf{x}_i (\mathbf{x}_i \cdot \hat{\mathbf{w}} - y_i) = \mathbf{X}^T (\mathbf{X}\hat{\mathbf{w}} - \mathbf{Y}) = \mathbf{X}^T \mathbf{X} \hat{\mathbf{w}} - \mathbf{X}^T \mathbf{Y} = 0$$

Now we can rearrange terms and solve for $\mathbf{w}$ with a bit of linear algebra:

$$\mathbf{X}^T \mathbf{X} \hat{\mathbf{w}} = \mathbf{X}^T \mathbf{Y} \Rightarrow \hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}.$$

This gives us the optimal estimate $\hat{\mathbf{w}}$ that minimizes the sum of squared errors.

2 Features

In machine learning, it is often useful for us to make a distinction between features and inputs. This is particularly true in the case of linear regression: since the function we are learning is linear, choosing the right features is important to get the necessary flexibility. For example, imagine that we want to predict salaries y for students in CSE 446. All we have is their grades on the homeworks, arranged into a 4D vector $\mathbf{x}$. If we learn a linear function of $\mathbf{x}$, we will probably see that the higher the scores get, the higher the salaries. But what if in reality, there are diminishing returns: students who get bad scores get low salaries, but higher salaries don't result in a proportional increase, since these scores are already "good enough" – the resulting function might look a little bit like a quadratic. So perhaps in this case, it would be really nice if we had some quadratic features. We can do this by defining a feature function $h(\mathbf{x}_i)$. For example, if we want quadratic features, we might define:

$$h(\mathbf{x}_i) = \begin{bmatrix} \mathbf{x}_{i,1} \\ \mathbf{x}_{i,1}^2 \\ \mathbf{x}_{i,2} \\ \mathbf{x}_{i,2}^2 \\ \mathbf{x}_{i,3} \\ \mathbf{x}_{i,3}^2 \\ \mathbf{x}_{i,4} \\ \mathbf{x}_{i,4}^2 \end{bmatrix}$$

Let $\mathbf{h}_i = f(\mathbf{x}_i)$, then we can write the linear regression problem in terms of the features $\mathbf{h}_i$ as

$$\hat{\mathbf{w}} \leftarrow \arg\min_{\mathbf{w}} \sum_{i=1}^{N} (\mathbf{h}_i \cdot \mathbf{w} - y_i)^2$$

We can then solve for the optimal $\hat{\mathbf{w}}$ exactly the same way as before, only using $\mathbf{H}$ instead of $\mathbf{X}$:

$$\hat{\mathbf{w}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{Y}$$

So we can see that in this way, we can use the same exact least squares objective and the same algorithm to fit a quadratic instead of a linear function. In general, we can fit any function that can be expressed as a linear combination of features.

Question. In the case of standard linear regression, expressed as $f(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i$, what is the data, hypothesis space, objective, and algorithm? What is the data, hypothesis space, objective, and algorithm in the case where we have $f(\mathbf{x}_i) = \mathbf{w} \cdot h(\mathbf{x}_i)$, where $h(\dots)$ extracts linear and quadratic features?

Answer. The data, objective, and algorithm are the same in both cases. The only thing that changes is the hypothesis space: in the former case, we have the class of all lines, and in the latter case, we have the class of all (diagonal) quadratic functions.

In fact, we can choose any class of features $h(\mathbf{x}_i)$ we want to learn complex functions of the input. A few choices are very standard for linear regression and are used almost always. For example, we often want to include a constant feature, e.g.

$$h(\mathbf{x}_i) = \begin{bmatrix} \mathbf{x}_{i,1} \\ \cdots \\ \mathbf{x}_{i,d} \\ 1 \end{bmatrix}$$

so that the linear regression weights $\mathbf{w}$ include a constant bias (which is simply the coefficient on the last, constant feature).

3 Linear regression as MLE

At this point, we might wonder *why* we actually use the sum of squared errors as our objective. The choice seems reasonable, but a bit arbitrary. In fact, linear regression can be viewed as (conditional) maximum likelihood estimation under a particular simple probabilistic model. Unlike in the MLE examples covered last week, now our dataset $\mathcal{D}$ includes both inputs and outputs, since $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$. So we will aim to construct a *conditional* likelihood of the form $p(y|\mathbf{x}, \theta)$.

Question. What kind of likelihood can we construct such that $\log p(y|\mathbf{x}, \theta)$ corresponds to squared error?

Answer. We saw last week that for continuous variables (such as y), a good choice of distribution is the Gaussian distribution. Note that the logarithm of a Gaussian distribution is quadratic in the variable y:

$$\log p(y|\mu, \sigma) = -\log \sigma - \frac{1}{2\sigma^2}(\mu - y)^2 + \text{const.}$$

That looks a lot like the squared error objective in linear regression – we just need to figure out how to design σ and μ. If we simply set $\mu = f(\mathbf{x})$, we get:

$$\log p(y|\mathbf{x}, \mathbf{w}, \sigma) = -\log \sigma - \frac{1}{2\sigma^2}(\mathbf{x} \cdot \mathbf{w} - y)^2 + \text{const.}$$

So the (conditional) log-likelihood of our entire dataset is given by

$$\mathcal{L}(\mathbf{w}, \sigma) = -N \log \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^{N} (\mathbf{x}_i \cdot \mathbf{w} - y_i)^2 + \text{const.}$$

If we set the derivative with respect to $\mathbf{w}$ to zero, we get

$$-\frac{1}{2\sigma^2} \sum_{i=1}^{N} \mathbf{x}_i(\mathbf{x}_i \cdot \mathbf{w} - y_i) = 0 \Rightarrow \sum_{i=1}^{N} \mathbf{x}_i(\mathbf{x}_i \cdot \mathbf{w} - y_i) = 0,$$

which is exactly the same equation as what we saw for the sum of squared errors objective. Therefore, the solution under this model is given, as before, by

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y},$$

and we can estimate the variance σ^2 by plugging in the optimal mean:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i \cdot \hat{\mathbf{w}} - y)^2.$$

However, σ^2 won't affect the maximum likelihood prediction of y, which will always be $\mathbf{x}_i \cdot \hat{\mathbf{w}}$, the most probable value according to the Gaussian model, so we often disregard σ^2 when performing linear regression.

What is the practical, intuitive interpretation of this probabilistic model? This model states that the data comes from some underlying Gaussian distribution $y \sim \mathcal{N}(\mathbf{x} \cdot \mathbf{w}, \sigma^2)$, so the samples we observe are noisy realizations of the underlying function $\mathbf{x} \cdot \mathbf{w}$. Therefore, any deviation from this function is modeled as symmetric Gaussian noise.