Chapter 1

Learning via Optimization

How do we learn from data? An approach commonly used in data science and machine learning is to turn the problem of learning into a math problem – in particular, an **optimization problem**. In this section, we'll see an example of how this is done in a simple setting. The basic ideas of this section underlie almost every important machine learning method, from simple linear regression to deep neural networks.

1.1 Predicting Your Future Salary

How much can you expect to earn as a data scientist? To get an idea, you might survey working data scientists and ask them about their salary.

Luckily, you don't have to do much footwork. Each year since 2011, StackOverflow has surveyed their users, asking them about their pay, experience, education, etc. In 2018, there were 820 responses from full-time data scientists working in the U.S. A random selection of five of their salaries is shown below:

$$\begin{aligned}90{,}000\\94{,}000\\96{,}000\\120{,}000\\160{,}000\end{aligned}$$

Given this data, how might you predict your future salary? One way is to compute the **mean** of the data by adding up the salaries and dividing by five:

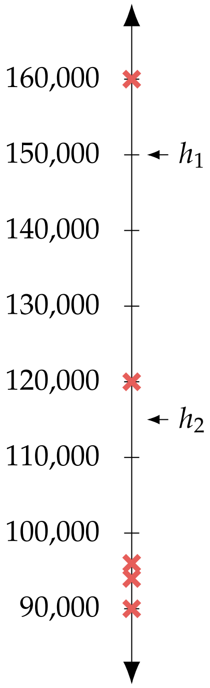
$$\frac{1}{5} \times (90{,}000 + 94{,}000 + 96{,}000 + 120{,}000 + 160{,}000) = 112{,}000$$

Another popular approach is to find the **median** salary. Recall that to find the median of a collection of numbers, we sort them and take the number in the middle (there are two "middle" numbers if the size of the collection is even – both are medians). The median of this data set is 96,000.

We already have two predictions for your future salary: \$112,000 (given by the mean) and \$96,000 (given by the median). Which prediction is more likely to be right? Is there a reason that the mean is higher? And are the mean and median salaries good ways of predicting your future salary? To answer these questions, we need to understand how the mean and the median are related to prediction. By the end of the chapter, we'll uncover some interesting connections.

1.1.1 Measuring the Error of a Prediction

What makes a prediction good? To get an intuition, let's plot the six salaries from the previous section on a number line:



Now consider two different predictions for your future salary: \$150,000, which we'll call h_1, and \$115,000, which we'll call h_2. These predictions are also shown on the number line above. Note that neither h_1 nor h_2 are the mean or the median of the data; they are just two arbitrary predictions.

Which is a *better* prediction of your future salary: h_1 or h_2? Most people would say that h_2 is better. When pressed to justify their answer, they might argue that h_2 is better because it is closer to the middle of the data points. This is fine reasoning, but let's make it more precise; let's find a way to *quantify* how good or bad a particular prediction is using this intuition.

Intuitively, a good prediction is one that is close to your future salary; the closer, the better. The distance between a prediction and your actual future salary is called the **error**, and can be calculated using the formula:

$$\text{error} = |\text{prediction} - (\text{actual future salary})| .$$

We use the absolute value here because a prediction that is one thousand dollars too high is just as wrong as a prediction that is one thousand dollars too low; they're both 1,000 dollars away from the actual salary.

Between h_1 and h_2, the better prediction is the one which is closer to your future salary; that is, the one with the smallest error. But there's a problem: we don't know your actual future salary, so we can't calculate the error of a prediction. We're stuck!

To get out of this predicament, we'll make use of an important assumption:

The future will look like the past.

Is this assumption always correct? No. Any number of things could happen: salaries could rise with inflation, there might be another recession, or the U.S. dollar might be replaced by bitcoin. But unless something drastic happens, it will not be too wrong.

If the future *does* look like the past (or is not too different), then a prediction that worked well *yesterday* will continue to work well *tomorrow*. This gives us a way out of our dilemma: to choose between two predictions, we pick that which has performed better in the past.

Consider again our data set of six salaries. Suppose that before we collected this data, we predicted that each of the data scientists surveyed makes $h_1 = \$150{,}000$ per year. We can then calculate just how wrong this prediction was for each of the observed salaries. For instance, one data scientist actually made 90,000; the error in estimating their salary was $|150{,}000 - 90{,}000| = 60{,}000$. Another data scientist actually made 160,000; for them, the error was $|150{,}000 - 160{,}000| = 10{,}000$, and so on. The table below shows the error in using h_1 to predict each salary, along with the total of all of the errors and the average error encountered.

salary	error of $h_1 = 150{,}000$
90,000	60,000
94,000	56,000
96,000	54,000
120,000	30,000
160,000	10,000
	total error: 210,000
	mean error: 42,000

As the table shows, the prediction h_1 is on average $42,000 away from the correct salary. If we were to use h_1 to predict your future salary, we would expect it to be about as wrong.

Now suppose we had used the prediction $h_2 = \$115{,}000$. In some cases, this prediction is worse; for instance, the error in predicting the salary of the data scientist who actually made 160,000 is 45,000 instead of just 10,000. But in aggregate, h_2 is closer to the data, as the table below demonstrates:

salary	error of $h_2 = 115{,}000$
90,000	25,000
94,000	21,000
96,000	19,000
120,000	5,000
160,000	45,000
	total error: 115,000
	mean error: 23,000

On average, h_2 is $23,000 away from the correct salary. This is much closer than h_1, and so h_2 is the better prediction.

We ended the last section by asking whether the mean of the data or the median is a better prediction of your future salary. We now have a way of comparing the two quantitatively: we'll compute the mean error of each and see which is lower. Recall that the mean salary is 112,000, while the median is 96,000. A quick calculation shows that the average error when the mean is used as the prediction is 22,400; this is lower than the mean error of both h_1 and h_2! But the mean error of the median is 19,200, which is ever-so-slightly smaller. In this view, the median is the best prediction found so far. But is it the best possible prediction? Is there a prediction which has even smaller average error?

1.1.2 Learning by Minimizing the Mean Error

Let's take a step back for a moment to remember our goal. We want to make an accurate prediction of your future salary. In this case, a prediction is simply a number – in fact, all (non-negative) real numbers are valid predictions. While there are a lot of possible predictions (infinitely-many, in fact), some are clearly worse than others. In the last section, we came up with a way of quantifying how bad a particular prediction is by computing the mean error it incurs on the data. In this view the *best* possible prediction is the real number which results in the smallest mean error.

Let's make this more precise by introducing some notation. Suppose we have gathered a data set of n salaries; let these be $y_1, y_2, \ldots y_n$. That is, y_1 is the salary of the first person, y_2 is the salary of the second, and so on. Suppose h is a prediction of your future salary. We'll write $R(h)$ for the mean error of the prediction h on the data $y_1, \ldots, y_n$; We can calculate $R(h)$ as follows:

$$\begin{aligned}
R(h) &= (\text{mean error in using } h \text{ to predict } y_1, \ldots, y_n) \\
&= \frac{1}{n}\left[(\text{error in using } h \text{ to predict } y_1) + \ldots + (\text{error in using } h \text{ to predict } y_n)\right] \\
&= \frac{1}{n}\left(|h - y_1| + |h - y_2| + \ldots + |h - y_n|\right)
\end{aligned}$$

Writing the summation of a bunch of terms using ... is sometimes imprecise, and isn't very concise. We'll typically use the equivalent **summation notation** below:

$$= \frac{1}{n}\sum_{i=1}^{n} |h - y_i|$$

You can check that, in reference to the example of the previous section, $R(h_1) = R(150{,}000) = 42{,}000$, whereas $R(h_2) = R(115{,}000) = 23{,}000$. Because $R(h_2) < R(h_1)$, we consider h_2 to be the better prediction. We can think of R as a function which takes in a prediction and outputs a score telling us how bad the prediction is; the bigger $R(h)$, the worse h is.

Our goal, again, to is to find the *best* prediction out of all possible predictions (that is, all non-negative real numbers). Our approach is to find the prediction h^* which results in the smallest possible value of $R(h)$. To put it another way: our goal is to **minimize** R over all non-negative real numbers. This is an example of an **optimiza-**

tion problem. We can write the problem more precisely using standard notation:

$$\begin{aligned} h^* &= \operatorname{arg\,min}_{h \in \mathbb{R}^+} R(h) \\ &= \operatorname{arg\,min}_{h \in \mathbb{R}^+} \frac{1}{n} \sum_{i=1}^{n} |h - y_i| \end{aligned}$$

This notation says that the best prediction, h^*, is the argument which minimizes $R(h)$ over all non-negative real numbers, $\mathbb{R}^+$.

1.1.3 Solving the Optimization Problem

Our goal is now precise: find the number h which minimizes the mean error, $R(h)$. This is an instance of a problem we've seen many times before in calculus: finding the minimum of a function.

Recall the trick for finding a local minimum or maximum of a function: we take the derivative, set it equal to zero, and solve. In essence, we are finding the places where the slope of the function is zero; these places must correspond to local minima or maxima.

Let's try this approach with the function R. First we must what variable to take the derivative in. In this case, we are concerned with how changing the prediction, h, changes the mean error, R, and so we will take the derivative with respect to h. We have:

$$\frac{dR}{dh} = \frac{d}{dh}\left[\frac{1}{n} \sum_{i=1}^{n} |y_i - h|\right]$$

Remember that constants don't phase derivative operators; we can move the d/dh right on past the $1/n$:

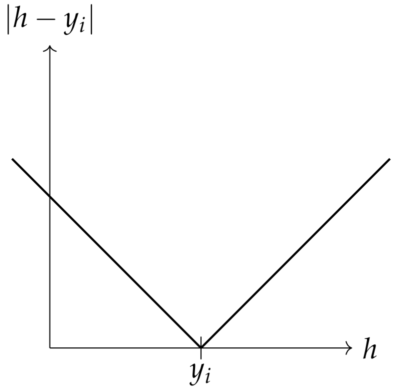
$$= \frac{1}{n} \frac{d}{dh} \sum_{i=1}^{n} |y_i - h|$$

Can we push the derivative inside of the summation symbol? It turns out that we can. Recall that the derivative of a sum is the sum of the derivatives; that is, $\frac{d}{dx}(f + g) = \frac{df}{dx} + \frac{dg}{dx}$. Our summation presents the same situation, just with n terms instead of two:

$$= \frac{1}{n} \sum_{i=1}^{n} \frac{d}{dh} |y_i - h|$$

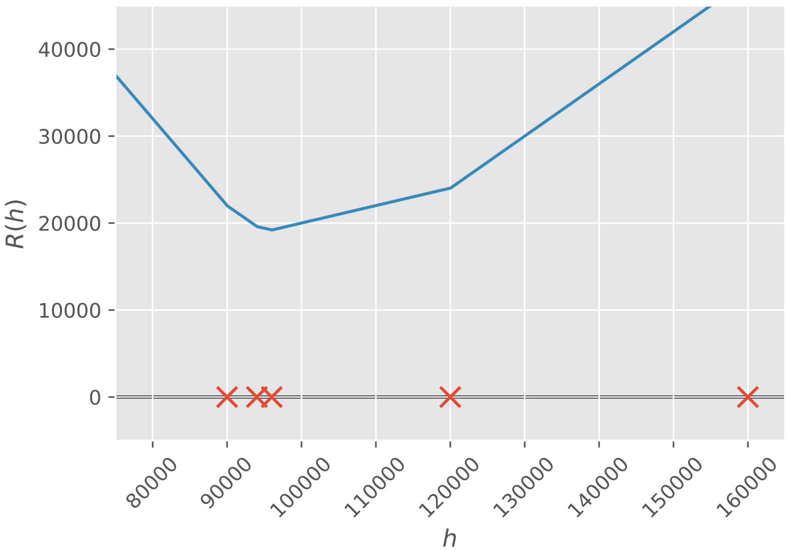
Now we're stuck. How do we take the derivative of an absolute value function?

Let's take a closer look at $|y_i - h|$. As a function of h, this is zero when $h = y_i$. When h is larger than y_i, the function looks like $h - y_i$; that is, it looks like a line with slope 1. When h is smaller than y_i, the function looks likes $y_i - h$; it is a line with slope -1. If we were to plot the function, we'd see:



One thing we notice in particular is that the slope of the function is not well-defined at $h = y_i$. This means that the function is not differentiable. We can't simply take the derivative, set to zero, and solve – there is no derivative!

Instead, we'll plot the function $R(h)$ and see what can be done. Below is the graph of $R(h)$ computed using the salaries from above as the data.



The first thing we notice about $R(h)$ is that it is **piecewise linear**. Indeed, this follows from the fact that R is the sum of absolute value functions, each of which are piecewise linear. A closer look shows that the points at which the function's slope changes are precisely the data points. The lowest value of $R(h)$ appears to coincide with the third data point, 96,000, which happens to be the median.

Notice that the minimum occurs where the slope goes from negative to positive. More generally, a minimizer is a point where the slope changes from negative to non-negative (possibly zero). This definition works for differentiable functions, too. Our goal now is to find a formula for the slope of R at a particular point, h, where we'll assume that h is not a data point, since the slope is not defined there.

Recall that $R(h) = \frac{1}{n}\sum_{i=1}^{n} |y_i - h|$. For any given h, we can divide the data points into those which are smaller than h and those which are larger. Furthermore, we can split the summation into two summations: one over the points smaller than h, and one over those larger:

$$R(h) = \frac{1}{n}\sum_{y_i<h} |y_i - h| + \frac{1}{n}\sum_{y_i>h} |y_i - h|$$

If $y_i < h$, then $|y_i - h| = h - y_i$. Likewise, if $y_i > h$, then $|y_i - h| = y_i - h$. Making these substitutions:

$$= \frac{1}{n}\sum_{y_i<h} (h - y_i) + \frac{1}{n}\sum_{y_i>h} (y_i - h)$$

The coefficient on h will tell us the slope of the line at h. We see that the coefficient is increased by $\frac{1}{n}$ for every data point less than h, and decreased by $\frac{1}{n}$ for every data point greater than h. Therefore, we have:

$$\text{slope at } h = \frac{1}{n}\left[(\text{\# of points} < h) - (\text{\# of points} > h)\right].$$

The slope will be negative when there are more points to the right of h than to the left of h. The slope will become non-negative precisely when there are an equal number of points to the left and right of h. This describes the median!

What we have shown, therefore, is that the mean error is indeed minimized the by the median. We have now solved the optimization problem posed by the previous section.

1.1.4 Learning by Minimizing Mean Squared Error

The fact that the median minimizes the mean error makes for a nice story, but we're not done yet. Remember that we couldn't use our favorite tool – calculus – to minimize the mean error because it is not differentiable. Instead, we had to do a more careful analysis. Is there a way to change the mean error – perhaps using a different way of measuring the accuracy of our prediction – so that it becomes differentiable?

Recall that the error of a prediction is given by:

$$\text{error} = |\text{prediction} - (\text{actual future salary})|\,.$$

The mean error is therefore the sum of a bunch of absolute values:

$$\text{mean error} = \frac{1}{n}\sum_{i=1}^{n} |h - y_i|$$

Remember that the absolute value presents problems when trying to minimize the above function because it is not differentiable.

What if we instead used a different way of measuring the accuracy of a prediction, the **squared error**:

$$\begin{aligned}\text{squared error} &= |\text{prediction} - (\text{actual future salary})|^2\\ &= (\text{prediction} - (\text{actual future salary}))^2\end{aligned}$$

Measuring the accuracy like this may not be as natural as using the error itself, but it works: the squared error is smaller the better the prediction is. Most importantly, however, it doesn't include the absolute value. The **mean squared error** is therefore:

$$\text{mean squared error} = \frac{1}{n}\sum_{i=1}^{n}(h - y_i)^2.$$

This is differentiable, and we can use calculus to find the minimizer!

To begin, we'll use $R_{\text{sq}}(h)$ to denote the mean squared error of the prediction h. That is:

$$R_{\text{sq}}(h) = \frac{1}{n}\sum_{i=1}^{n}(h - y_i)^2.$$

Our goal is to find the prediction with the smallest mean squared error. To do so, we will minimize R_{sq} by taking a derivative, setting it to zero, and solving for the best h:

$$\frac{dR_{\text{sq}}}{dh} = \frac{d}{dh}\left[\frac{1}{n}\sum_{i=1}^{n}(y_i - h)^2\right].$$

The derivative operator can be pushed inside of the summation:

$$= \frac{1}{n}\sum_{i=1}^{n}\frac{d}{dh}(y_i - h)^2.$$

A careful application of the chain rule results in:

$$= \frac{2}{n}\sum_{i=1}^{n}(h - y_i).$$

To minimize $R_{\text{sq}}(h)$ over all possible h, we set the derivative to zero and solve for h:

$$\begin{aligned}
&\frac{dR_{\text{sq}}}{dh} = 0 \\
\implies &\frac{2}{n}\sum_{i=1}^{n}(h - y_i) = 0.
\end{aligned}$$

Dividing both sides by two:

$$\implies \frac{1}{n}\sum_{i=1}^{n}(h - y_i) = 0.$$

The summation can be split into two summations:

$$\begin{aligned}
\implies &\frac{1}{n}\sum_{i=1}^{n}h - \frac{1}{n}\sum_{i=1}^{n}y_i = 0, \\
\implies &\frac{1}{n}\sum_{i=1}^{n}h = \frac{1}{n}\sum_{i=1}^{n}y_i.
\end{aligned}$$

Since h is a constant, we can pull it out of the first summation:

$$\begin{aligned}
\implies &\frac{h}{n}\sum_{i=1}^{n}1 = \frac{1}{n}\sum_{i=1}^{n}y_i, \\
\implies &\frac{h}{n}\cdot n = \frac{1}{n}\sum_{i=1}^{n}y_i. \\
\implies &h = \frac{1}{n}\sum_{i=1}^{n}y_i.
\end{aligned}$$

This calculation shows that the choice of hypothesis that minimizes the mean squared error is the **mean** of the data!

1.2 Loss Functions

We so far have two ways of predicting someone's future salary given observed salaries. First, we can find the prediction that minimizes the mean error; this results in the **median** of the data. Or we can find the prediction that minimizes the mean *squared* error; this gives the **mean** of the data.

Which is better? In one sense, the mean error is more natural than the mean squared error. It is also less sensitive to outliers in the data. To see this, consider the following salaries:

90,000 94,000 96,000 120,000 160,000

The median is 96,000, and the mean is 112,000. Now suppose the person who made 160,000 gets a raise and their salary doubles:

90,000 94,000 96,000 120,000 320,000

The mean of this new data set jumps, too, to 144,000, but the median stays at 96,000. It is arguable that the median remains a better description of a typical salary.

On the other hand, the median is harder to compute than the mean. Imagine how you would compute the median of 1,000 numbers without a computer. If you're like most people, you'd sort the numbers first and then find the one in the middle. But sorting 1,000 numbers is no easy task. Computing the mean of 1,000 numbers involves adding 1,000 numbers and dividing the result by 1,000. This may be tedious, but still seems easier than sorting so many numbers.

Of course, computers can sort very quickly, so finding the median is not much harder for them than finding the mean.[1] The larger point, however, is that a small change in how error is measured resulted in two different answers with very different properties. On one hand, the mean squared error is easy to minimize, but is sensitive to outliers. On the other, the mean error is robust to outliers, but it is harder to find its minimizer. As we'll see, this sort of tradeoff is typical in machine learning.

While we took different routes to minimizing the mean error and the mean squared error, the general strategy was the same. First, we settled on a way of measuring the difference between a prediction and the "right answer". We then tried to find the prediction which minimized this difference on average. This general paradigm is called **empirical risk minimization** (or **ERM**) and is at the heart of many of the machine learning methods that you've heard of.

The first step in ERM involves choosing a **loss function** $L(h, y)$. A loss function is simply a function that takes in a prediction and a correct answer and outputs a number quantifying how far h is from y. We have already seen two loss functions, implicitly: the **absolute loss**, defined by:

$$L_{\text{abs}}(h, y) = |h - y|$$

[1] It turns out that it is possible to find the median without sorting the data. The result is an algorithm for the median which is essentially as efficient as the normal algorithm for the mean.

and the **square loss**, defined by:

$$L_{\text{sq}}(h, y) = (h - y)^2.$$

Once we've picked a loss function, L, and gathered some data, $y_1, \ldots, y_n$, we can measure the average loss of a prediction, h, on the data:

$$\frac{1}{n} \sum_{i=1}^{n} L(h, y_i)$$

If the average loss is small, the prediction is good; if it is large, the prediction is bad.

In machine learning, the average loss on the data goes by the name of **empirical risk**. Our goal is therefore to pick a prediction which minimizes the empirical risk; thus the name, empirical risk minimization.

Suppose we choose our loss function to be the absolute loss, $L_{\text{abs}}(h, y) = |h - y|$. The empirical risk is then:

$$\begin{aligned} R_{\text{abs}}(h, y) &= \frac{1}{n} \sum_{i=1}^{n} L_{\text{abs}}(h, y_i) \\ &= \frac{1}{n} \sum_{i=1}^{n} |h - y_i| \end{aligned}$$

You'll recognize this as the mean (absolute) error! And if we use the square loss, $L_{\text{sq}}(h, y) = (h - y)^2$, the empirical risk is:

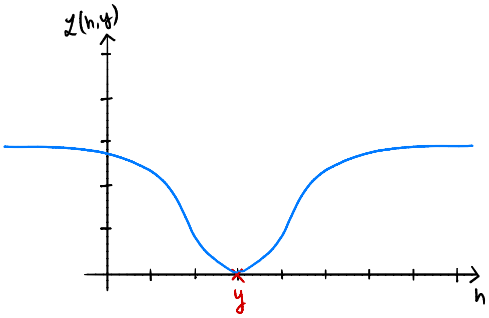
$$\begin{aligned} R_{\text{sq}}(h, y) &= \frac{1}{n} \sum_{i=1}^{n} L_{\text{sq}}(h, y_i) \\ &= \frac{1}{n} \sum_{i=1}^{n} (h - y_i)^2 \end{aligned}$$

This is, of course, simply the mean squared error.

1.2.1 Designing Our Own Loss Function

The absolute loss and square loss are fine choices of loss function, but let's design our own for fun. In this part, we'll design a loss function that is very insensitive to outliers. We'll then try to minimize the risk (a.k.a, the average loss), but run into some trouble. In the next section, we'll invent the method of **gradient descent** in order to resolve our difficulties.

One property that we might want from our loss function is an insensitivity to outliers. The square loss is sensitive because it grows faster and faster the further h is from y. What if our loss function takes the opposite approach, and grows very slowly. For instance, what if our loss function looked like that below:



This plot shows the right answer, y, as a red ×. The blue line shows what the loss might look like as h moves away from y. At first, the loss increases, signaling that h is becoming less and less accurate. At a point, however, the loss stops growing and essentially levels out. In essence, the loss function penalizes all outliers roughly the same; it is saying "an outlier is an outlier; whether it is 100 units away or 1,000,000".

Our loss function looks something like a bell curve that has been inverted and shifted up so that its minimum value is zero. As such, a function which describes the shape above is

$$L_{\text{UCSD}}(h, y) = 1 - e^{-(h-y)^2/\sigma^2},$$

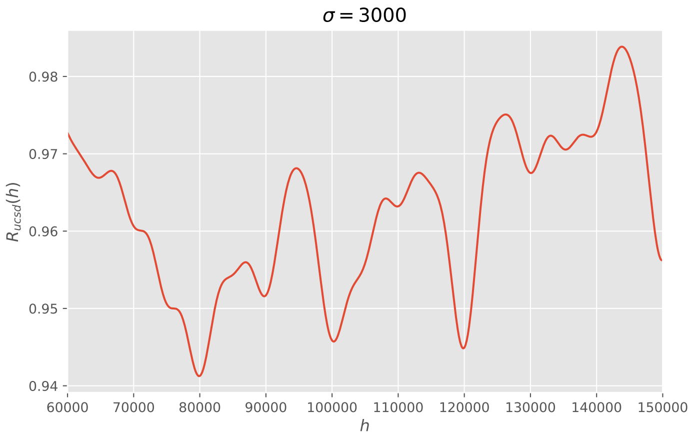
where σ is a **scale parameter**, to be discussed in a moment. This loss function isn't popular; it doesn't even have a name (so I've called it the UCSD loss). It isn't incredibly useful, either. But we'll explore its properties, and in doing so we'll get a better handle on empirical risk minimization.

As mentioned above, the σ in the definition of L_{UCSD} is a scale parameter. It determines the point where the function starts to level off. If σ is large, the level-off point occurs further from y. In essence, σ determines what an outlier is. If we're working with salary data, we might set σ to be in the tens of thousands of dollars. If we're working with temperatures, on the other hand, we might set σ to be around ten.

Let's use this loss function to make a prediction for your future salary. We start by writing down the risk:

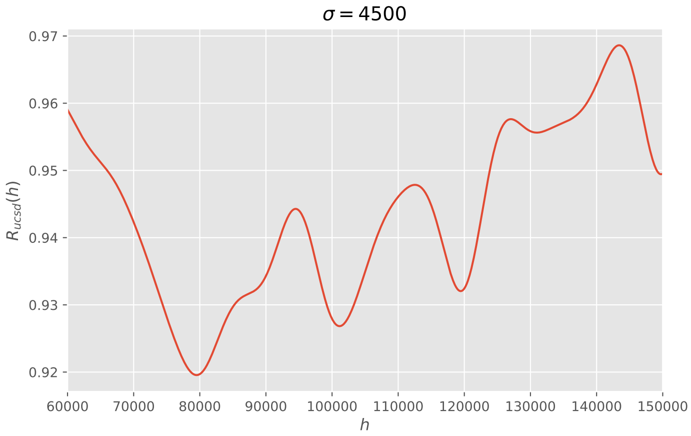
$$\begin{aligned} R_{\text{UCSD}}(h) &= \frac{1}{n}\sum_{i=1}^{n} L_{\text{UCSD}}(h, y_i) \\ &= \frac{1}{n}\sum_{i=1}^{n} \left[1 - e^{-(h-y_i)^2/\sigma^2}\right] \end{aligned}$$

Let's take a step back for a moment to visualize this function. In order to plot $R_{\text{UCSD}}(h)$, we first have to pick a value for the scale parameter, σ. Let's start with $\sigma = 3000$:

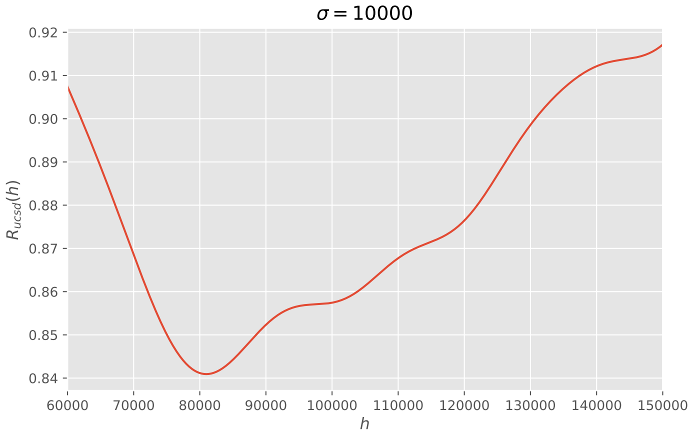


We see that the function is somewhat wiggly. This is because $R_{\text{UCSD}}(h)$ is the average of a bunch of loss functions, L_{UCSD}, which look like upside-down bell curves centered around the data points. The width of each of these curves is controlled by σ. When σ is small, the width is too, and so the sum is rather jagged.

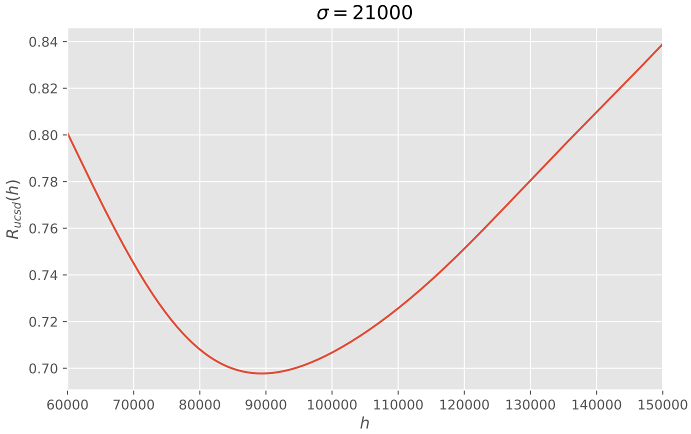
Now let's plot the risk when the scale parameter is set to $4,500$.



We notice that as we increase σ, the function becomes smoother. At $\sigma = 10000$ it still has a few wrinkles:



But at $\sigma = 21000$ it is almost entirely smoothed out.



Our goal is to find the minimizer of this function. It is differentiable, so we'll try our familiar strategy of taking a derivative, setting it to zero, and solving. You can verify that the derivative is:

$$\frac{dR_{\text{UCSD}}}{dh}(h) = \frac{2}{n\sigma^2}\sum_{i=1}^{n}(h - y_i) \cdot e^{-(h-y_i)^2/\sigma^2}$$

Now we just need to set this to zero and solve for h. We have:

$$\frac{dR_{\text{UCSD}}}{dh}(h) = 0 \implies \frac{2}{n\sigma^2}\sum_{i=1}^{n}(h - y_i) \cdot e^{-(h-y_i)^2/\sigma^2} = 0$$

Multiplying both sides by $2/(n\sigma^2)$:

$$\implies \sum_{i=1}^{n}(h - y_i) \cdot e^{-(h-y_i)^2/\sigma^2} = 0$$

Now we split the summand in an attempt to isolate h on one side of the equation:

$$\implies \sum_{i=1}^{n} h \cdot e^{-(h-y_i)^2/\sigma^2} - \sum_{i=1}^{n} y_i \cdot e^{-(h-y_i)^2/\sigma^2} = 0$$

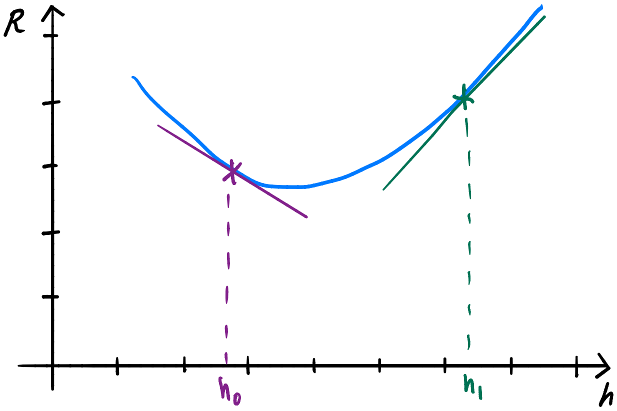
It is here that we get stuck. You can try several things, but you'll never be able to get an equation with just h on one side; the problem is that h appears in the exponential, and it is hard to remove it.

This is a case where we have a function R_{UCSD} which is differentiable, but whose derivative is complicated enough that we cannot solve for the place where it is zero. We'll need to use a different approach: **gradient descent**.

1.3 Gradient Descent

In the last section we found a risk function R_{UCSD} which is differentiable, but whose derivative is complicated enough that we couldn't directly solve for its minimizer. In response, we'll develop a strategy for minimizing risk functions that plays a large role in modern machine learning: **gradient descent**.

Recall that the derivative of a function R at a point h tells us the slope of R at that point. Consider for instance the function plotted in the figure below. The derivative of this function at the point h_0 is negative, while the derivative at h_1 is positive.



Suppose we were standing on the blue line at h_0 and wearing a blindfold. We wouldn't be able to see what the blue line does anywhere else; in particular, we wouldn't be able to see where the minimizer is. But we could feel around with our toe and realize that the line decreases from left to right. That is, the slope is negative. And with only this information at our disposal, the best choice would be to move to the right. Likewise, if we were at the point h_1, we would be able to tell that the slope is positive, and that we should move to the left.

This suggests an iterative procedure for finding the minimizer. We start at some point, h_0; perhaps chosen randomly. We then compute the derivative at h. If it is positive, we walk to the left (decrease h). If it is positive, we walk to the right (increase h).

How big of a step do we take? If the slope at our current location is close to zero, we might be close to a local minimum, so we shouldn't take too large of a step. It therefore makes sense to take a step of size $|dR/dh|$; the magnitude of the slope. More precisely, if we are standing at a point h_0, then our next position h_1 should be:

$$h_1 = h_0 - \frac{dR}{dh}(h_0)$$

This is called the **update rule**. Check that when the slope is negative, this rule will *increase* our prediction, h. Likewise, when the slope is positive, this rule will *decrease* our prediction.

Once we've made our step, we are (hopefully) closer to the minimizer. Naturally, we repeat the process: compute the slope at our new position and decide on which direction to move, and how far. The hope is that we'll eventually get close to a local minimum. If the function is indeed differentiable, then the slope near the local minimum is close to zero, and our step sizes will be proportionately small. Our steps will eventually become so small that we are barely moving; at this point, we will say that the process has **converged** to an answer.

The procedure we have just described is **gradient descent**. More concisely, gradient descent has the following steps:

- Pick α to be a positive number. It is the **learning rate**.
- Pick a starting prediction, h_0.
- On step i, perform update $h_i = h_{i-1} - \alpha \cdot \frac{dR}{dh}(h_{i-1})$
- Repeat until convergence (when h doesn't change much).

There is only one thing new here: α. It is a parameter that allows us to tweak how large our steps are. Choosing α to be large can get us to the minimizer faster, but choosing it to be too large can cause us to never reach the minimizer at all, as we'll soon see.