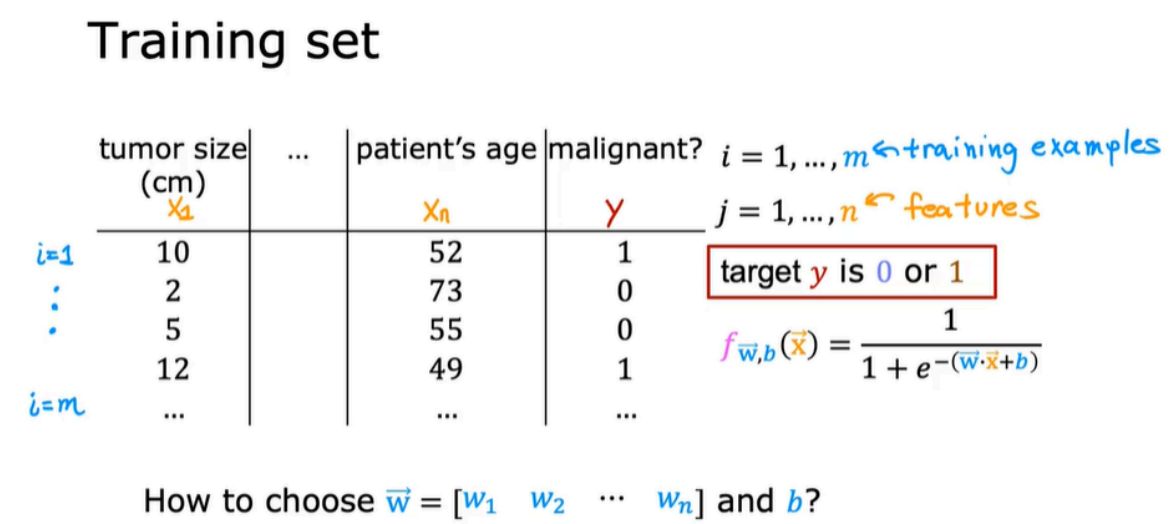
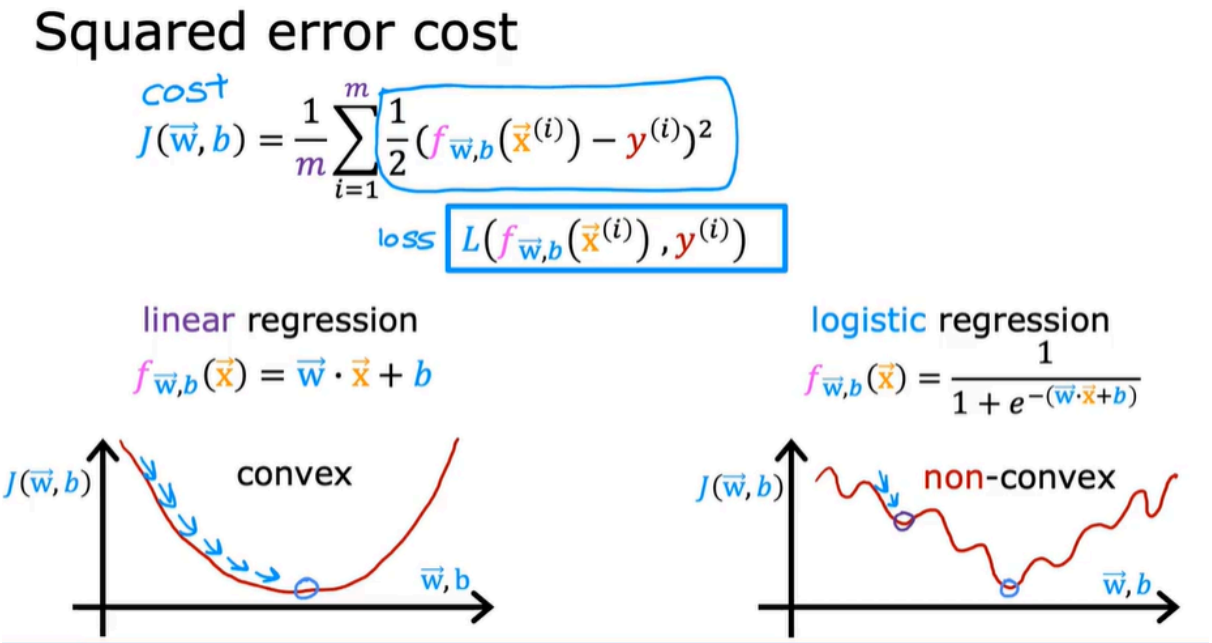
Cost function for logistic regression

Cost function for logistic regression

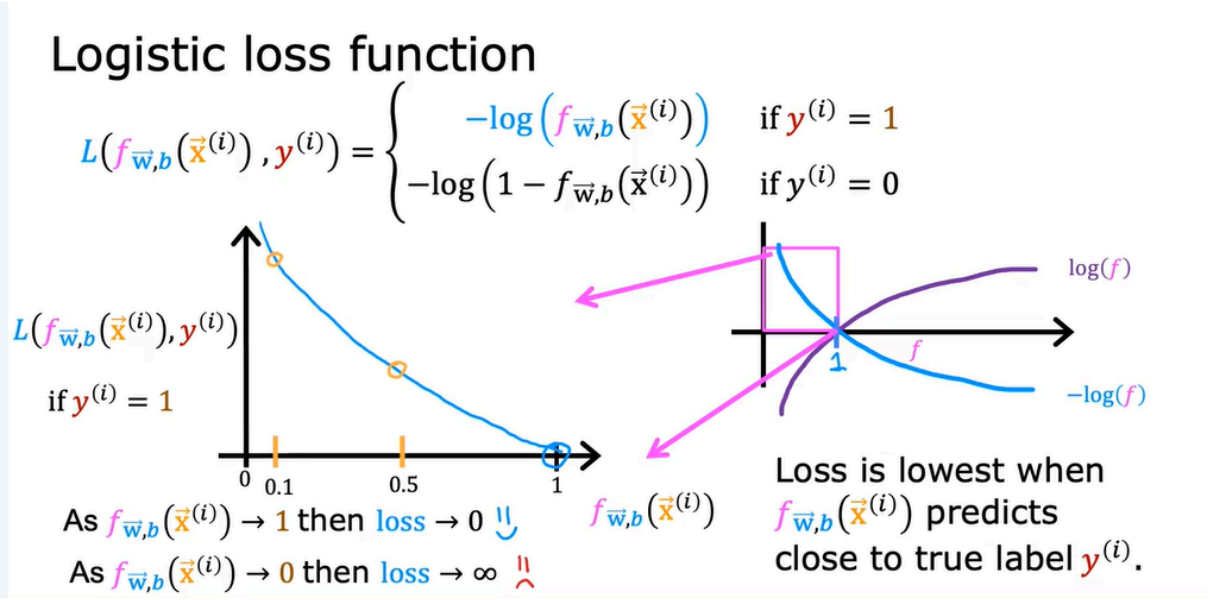


- Giving this Training set, how to choose $\vec{w}$ and b



- You might remember that in the case of linear regression, where f(x) is the linear function, $\vec{w} \cdot \vec{x} + b$. The cost function looks like this (the image in the left), is a convex function or a bowl shape or hammer shape. Gradient descent will look like this, where you take one step, one step, and so on to converge at the global minimum.
- Now you could try to use the same cost function for logistic regression. But it turns out that if I were to write $f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-(\vec{w} \cdot \vec{x}+b)}}$ and plot the cost function using this value of f of x, then the cost will look like this (The image in the right). This becomes what's called a non-convex cost function is not convex. What this means is that if you were to try to use gradient descent. There are lots of local minima that you can get sucking.
- It turns out that for logistic regression, this squared error cost function is not a good choice. Instead, there will be a different cost function that can make the cost function convex again. The gradient descent can be guaranteed to converge to the global minimum. The only thing I've changed is that I put the one half inside the summation instead of outside the summation. This will make the math you see later on this slide a little bit simpler. In order to build a new cost function, one that we'll use for logistic regression. I'm going to change a little bit the definition of the cost function $J(\vec{w}, b)$.
- In particular, if you look inside this summation, let's call this term inside the loss on a single training example. I'm going to denote the loss via this capital L and as a function of the prediction of the learning algorithm, f(x) as well as of the true label y. The loss given the predictor f of x and the true label y is equal in this case to 1.5 of the squared difference. We'll see shortly that by choosing a different form for this loss function, will be able to keep the overall cost function, which is 1 over n times the sum of these loss functions to be a convex function.

Logistic loss function

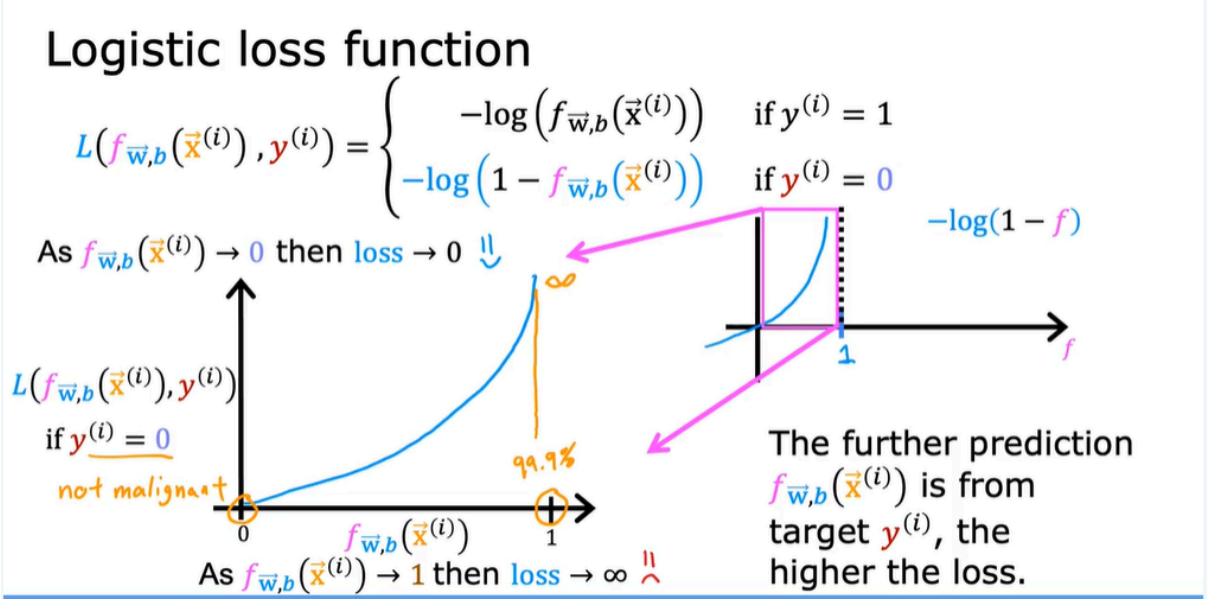


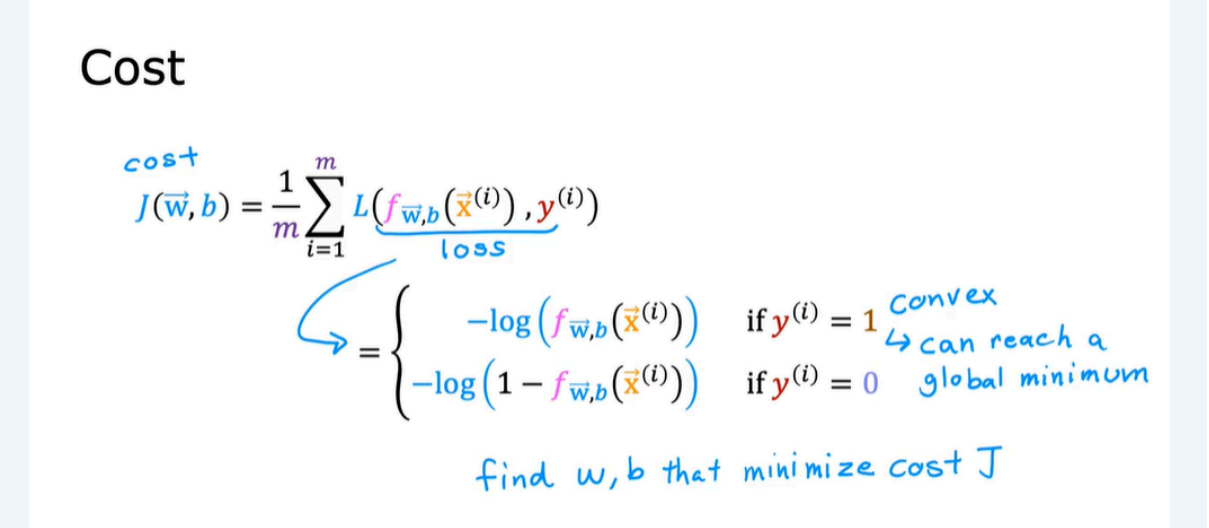
Now, the loss function inputs f of x and the true label y and tells us how well we're doing on that example. I'm going to just write down here at the definition of the loss function we'll use for logistic regression.

- If $y^{(i)}$ = 1, then the loss is -log(f(x)) and if $y^{(i)}$ = 0, then the loss is -log(1 - f(x)). Let's take a look at why this loss function hopefully makes sense. Let's first consider the case of y equals 1 and plot what this function looks like to gain some intuition about what this loss function is doing.
    - Remember, the loss function measures how well you're doing on one training example and is by summing up the losses on all of the training examples that you then get, the cost function, which measures how well you're doing on the entire training set.
- If you plot log(f), it looks like this curve here (the purple line in the right), where f here is on the horizontal axis. A plot of a -log(f) looks like this, where we just flip the curve along the horizontal axis. Notice that it intersects the horizontal axis at f equals 1 and continues downward from there.
- Now, f is the output of logistic regression. Thus, f is always between zero and one because the output of logistic regression is always between zero and one. The only part of the function that's relevant is therefore this part over here, corresponding to f between 0 and 1. Let's zoom in and take a closer look at this part of the graph. If the algorithm predicts a probability close to 1 and the true label is 1, then the loss is very small. It's pretty much 0 because you're very close to the right answer.

- Now continue with the example of the true label y being 1, say everything is a malignant tumor. If the algorithm predicts 0.5, then the loss is at this point here, which is a bit higher but not that high.
- Whereas in contrast, if the algorithm were to have outputs at 0.1 if it thinks that there is only a 10 percent chance of the tumor being malignant but y really is 1. If really is malignant, then the loss is this much higher value over here. When y is equal to 1, the loss function incentivizes or nurtures, or helps push the algorithm to make more accurate predictions because the loss is lowest, when it predicts values close to 1.

On this slide, let's look at the second part of the loss function corresponding to when y is equal to 0





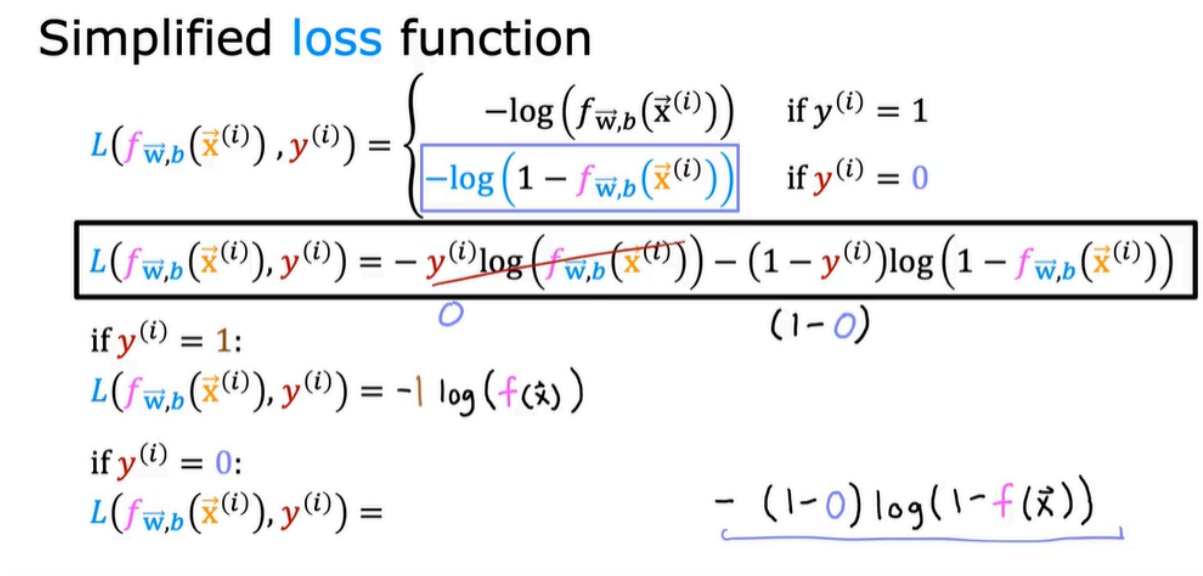
You saw why the squared error cost function doesn't work well for logistic regression. We also defined the loss for a single training example and came up with a new definition for the loss function for logistic regression.

It turns out that with this choice of loss function, the overall cost function will be convex and thus you can reliably use gradient descent to take you to the global minimum. Proving that this function is convex, it's beyond the scope of this cost.

You may remember that the cost function is a function of the entire training set and is, therefore, the average or 1/m times the sum of the loss function on the individual training examples. The cost on a certain set of parameters, w and b, is equal to 1/m times the sum of all the training examples of the loss on the training examples.

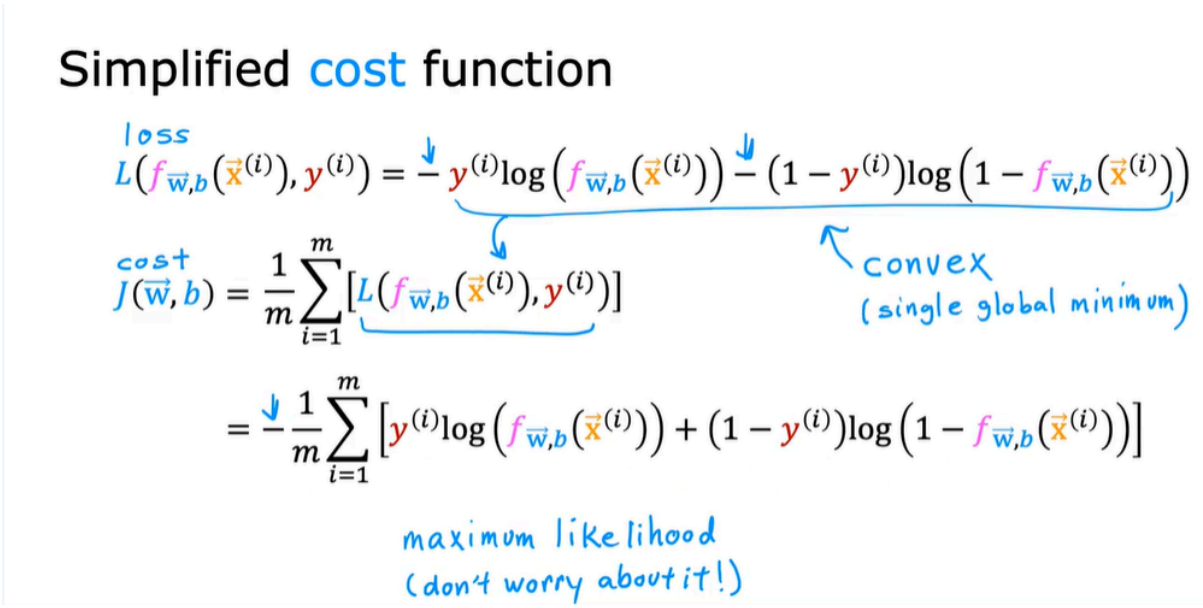
If you can find the value of the parameters, w and b, that minimizes this, then you'd have a pretty good set of values for the parameters w and b for logistic regression.

Simplified Cost Function for Logistic Regression



Cost Function:

$$L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}) = -y^{(i)} \log(f_{\vec{w},b}(\vec{x}^{(i)})) - (1 - y^{(i)}) \log(1 - f_{\vec{w},b}(\vec{x}^{(i)}))$$



You might be wondering, why do we choose this particular function when there could be tons of other costs functions we could have chosen?

Although we won't have time to go into great detail on this in this class, I'd just like to mention that this particular cost function is derived from statistics using a statistical principle called maximum likelihood estimation, which is an idea from statistics on how to efficiently find parameters for different models.

This cost function has the nice property that it is convex. But don't worry about learning the details of maximum likelihood. It's just a deeper rationale and justification behind this particular cost function.