Understanding the Loss function

In performing Error Analysis (post-training) out of sample, we *identified* examples where our model failed to generalize.

What can we do to make the model better ? How can we influence the models' choice of Θ to lead to a better fit ?

In the event that the Performance Metric (evaluated out of sample) and the Loss Function (evaluated in sample) differ

- We must see how we can influence the Loss function
- In the hope that better in sample performance leads to better out of sample performance

Now is a good time to recall the distinction between Accuracy (Performance Metric) and Cross Entropy (Loss function) for classification

Recall the mapping of probability to prediction

$$\hat{y}^{(\mathbf{i})} = \begin{cases} \text{Negative} & \text{if } \hat{p}^{(\mathbf{i})} < 0.5 \\ \text{Positive} & \text{if } \hat{p}^{(\mathbf{i})} \geq 0.5 \end{cases}$$

where, for Logistic Regression, probability $\hat{p}^{(\mathbf{i})}$ is a function of $\mathbf{x}^{(\mathbf{i})}$ and parameters Θ.

$$\hat{p}^{(\mathbf{i})} = \sigma(\Theta^T \cdot \mathbf{x}^{(\mathbf{i})})$$

- Accuracy (for example i) won't *necessarily* vary with Θ unless $\hat{p}^{(\mathbf{i})}$ crosses the threshold of 0.5
- But $\hat{p}^{(\mathbf{i})}$ will vary with Θ

Thus, a Θ' which pushes $\hat{p}^{(\mathbf{i})}$ closer to the correct probability (0 or 1) may be preferred to a Θ that leaves $\hat{p}^{(\mathbf{i})}$ farther away.

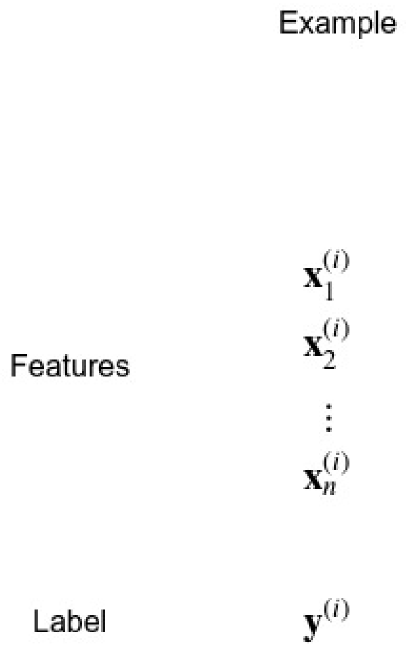
In this section

- We will be using *training examples* (in-sample) rather than out of sample data.
- In an attempt to reduce the Loss function
- Under the assumption that better in sample performance will lead to better out of sample performance

Recall that

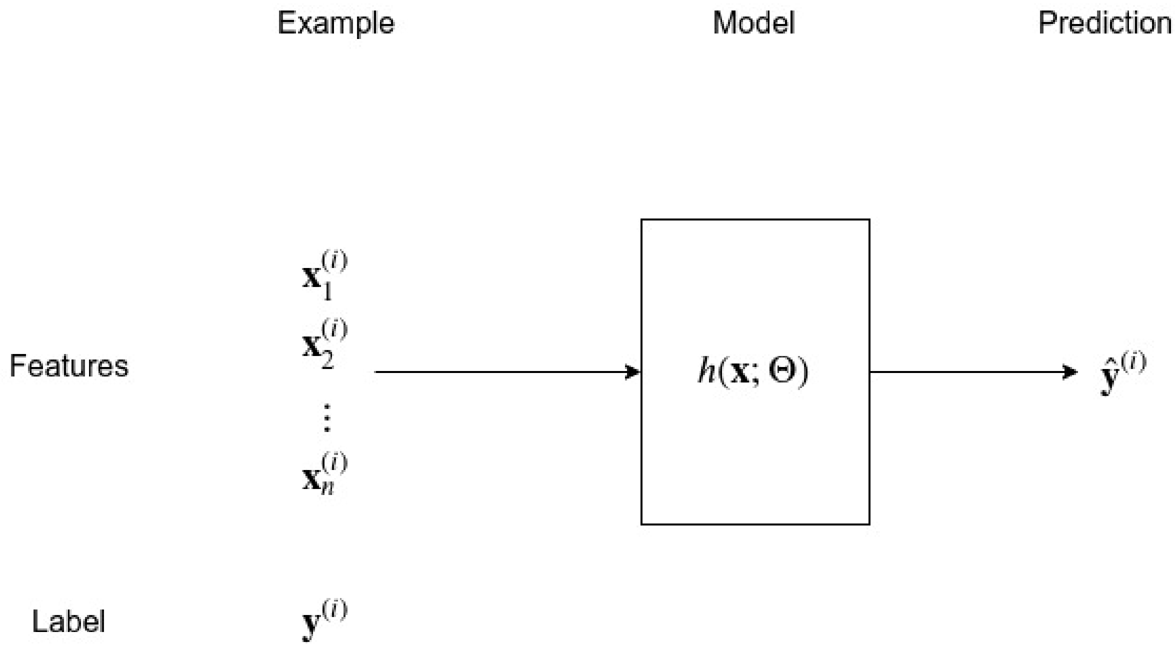
- the model is a function of parameters Θ
- Θ is found by minimizing Average Loss $\mathcal{L}_\Theta$
- The Average Loss is the average of the per-examples losses $\mathcal{L}_\Theta^{(\mathbf{i})}, i = 1, \ldots, m$

Training Example



	Example
Features	$\mathbf{x}_1^{(i)}$ $\mathbf{x}_2^{(i)}$ $\vdots$ $\mathbf{x}_n^{(i)}$
Label	$\mathbf{y}^{(i)}$

Training



Example

Model

Prediction

Features

$\mathbf{x}_1^{(i)}$

$\mathbf{x}_2^{(i)}$

$\vdots$

$\mathbf{x}_n^{(i)}$

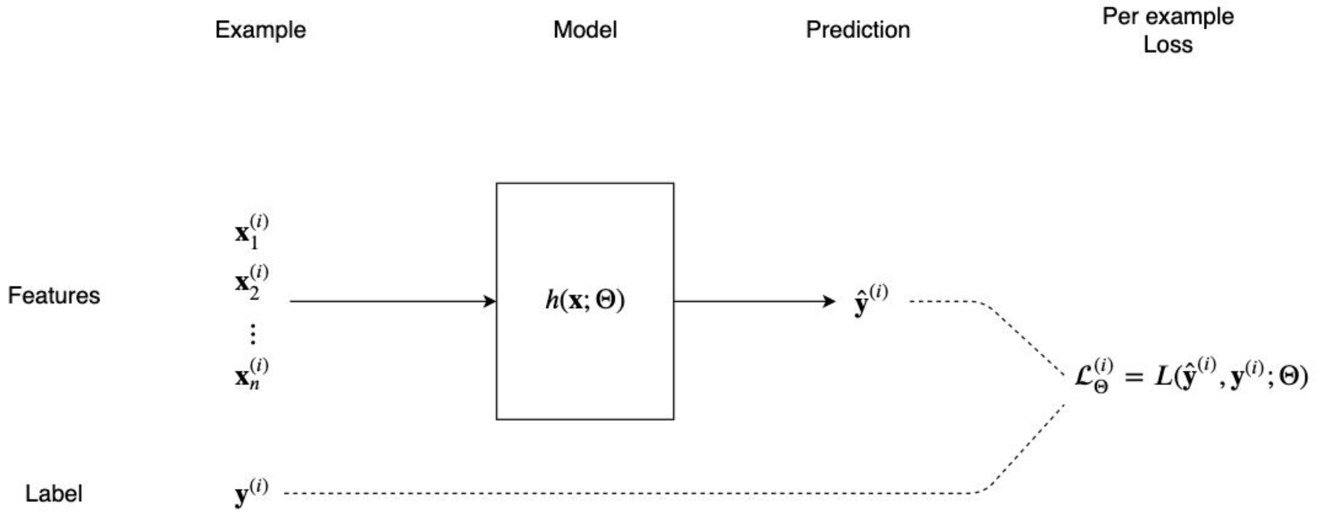
$h(\mathbf{x}; \Theta)$

$\hat{\mathbf{y}}^{(i)}$

Label

$\mathbf{y}^{(i)}$

Training Example



Training Example				
$\mathbf{x}^{(1)}$	$\mathbf{y}^{(1)}$	$\hat{\mathbf{y}}^{(1)}$	$\mathcal{L}_{\Theta}^{(1)}$	
$\mathbf{x}^{(2)}$	$\mathbf{y}^{(2)}$	$\hat{\mathbf{y}}^{(2)}$	$\mathcal{L}_{\Theta}^{(2)}$	
	$\vdots$			
$\mathbf{x}^{(i)}$	$\mathbf{y}^{(i)}$	$\hat{\mathbf{y}}^{(i)}$	$\mathcal{L}_{\Theta}^{(i)}$	
	$\vdots$			
$\mathbf{x}^{(m)}$	$\mathbf{y}^{(m)}$	$\hat{\mathbf{y}}^{(m)}$	$\mathcal{L}_{\Theta}^{(m)}$	
			$\mathcal{L}_{\Theta}$	Total Loss

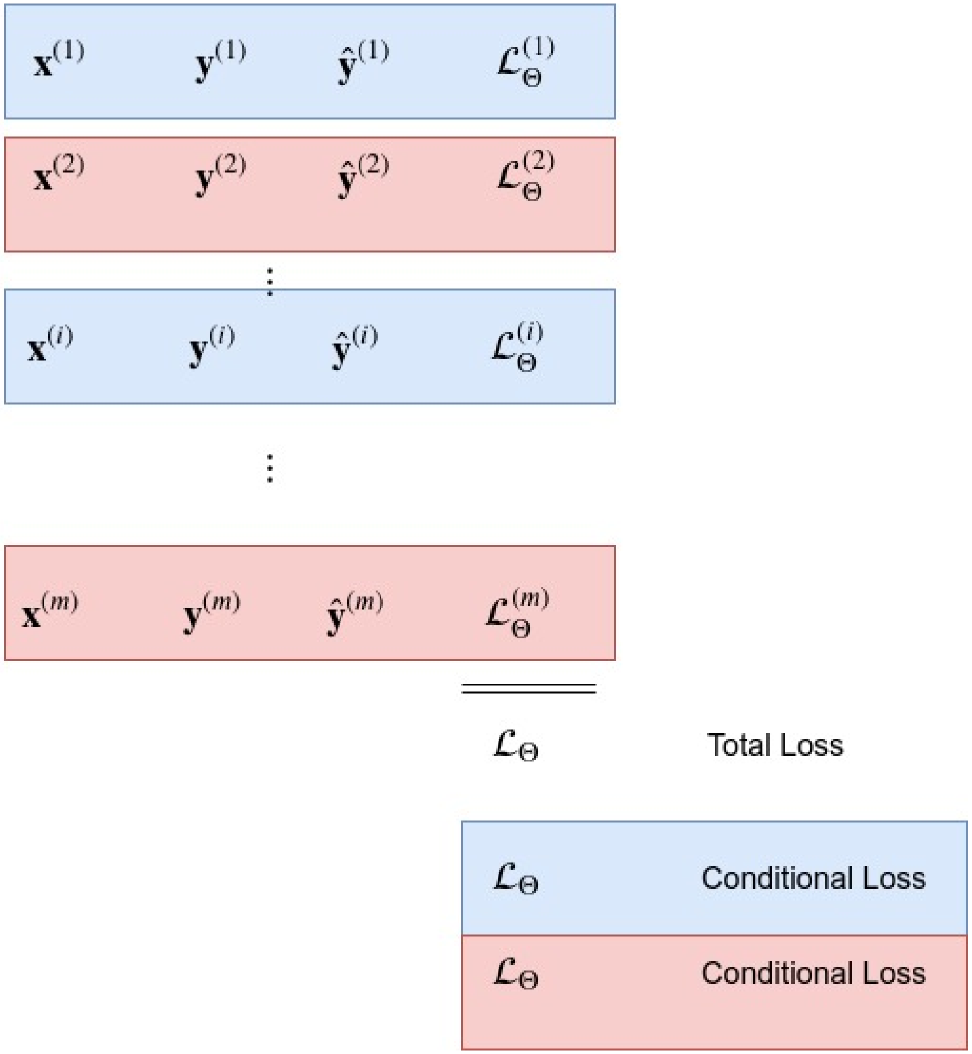
Conditional loss

The key to improving the model is understanding who each per example loss contributes to the optimizer choosing Θ.

One way to try to improve the model is to look at the per-example losses in Training

- similar to the way we looked at Errors out of sample
- colors represented groups similar examples

Loss analysis: conditional loss



$\mathbf{x}^{(1)}$ $\mathbf{y}^{(1)}$ $\hat{\mathbf{y}}^{(1)}$ $\mathcal{L}_\Theta^{(1)}$

$\mathbf{x}^{(2)}$ $\mathbf{y}^{(2)}$ $\hat{\mathbf{y}}^{(2)}$ $\mathcal{L}_\Theta^{(2)}$

$\vdots$

$\mathbf{x}^{(i)}$ $\mathbf{y}^{(i)}$ $\hat{\mathbf{y}}^{(i)}$ $\mathcal{L}_\Theta^{(i)}$

$\vdots$

$\mathbf{x}^{(m)}$ $\mathbf{y}^{(m)}$ $\hat{\mathbf{y}}^{(m)}$ $\mathcal{L}_\Theta^{(m)}$

$\mathcal{L}_\Theta$ Total Loss

🟦 $\mathcal{L}_\Theta$ Conditional Loss

🟥 $\mathcal{L}_\Theta$ Conditional Loss

What can we do to reduce loss ?

Understanding the per example loss can help you "push" the optimizer toward find a "better" Θ.

We will outline some simple strategies via examples that identify a probelm and propose a solution.

Increase number of ”problem” training example

In our MNNIST digit classification error analysis, we identified a certain sub-class of the digit "8" that was mis-classified

- at least one of the "holes" in the 8 was very small
- the digit was slanted in "opposite" direction

Recall

$$\mathcal{L}_\Theta = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}_\Theta^{(\mathbf{i})}$$

So problem example i's contribution to Average Loss is $\frac{1}{m} \mathcal{L}_\Theta^{(\mathbf{i})}$.

If the number of problem training examples of a particular type is small, the sum of the per example losses due to the problem examples may not have enough of an impact on $\mathcal{L}$ to affect the solution Θ.

One strategy for pushing the model to better fit the problem examples is to increase their number !

- so total weight of this particular class of problems has greater impact on $\mathcal{L}$

If you can find (or synthesize) similar types of problem examples, adding them to the training set forces the optimizer to better accomodate these examples.

We will introduce *Data Augmentation* is a later module.

Decrease the influence of a ”problem” example

Sometimes the problem is not having too few "problem" examples, but is having a few "problems" that are so off-scale that they unduly influence Θ.

- That is: $\mathcal{L}_\Theta^{(\mathbf{i})}$ is so large that it dominates $\mathcal{L}$ and forces Θ to accomodate

Influential points

Some models may be quite sensitive to just a few observations.

This is particularly true for Linear Regression.

Our discussion is somewhat specialized to Linear Regression but you may come to see a similar phenomenon in other models.

Loosely speaking, an observation is **influential** if

- the parameter estimate Θ changes greatly depending on whether the observation is included/excluded

Feature values on the extreme ends of the range have greater potential for being influential.

This is one argument for constraining the range of the feature (MinMax, Standardization).

The **leverage** of an observation is related to the value of a feature in relation to the mean (across observations) of the feature

- extreme values of the feature have higher leverage

It is not always the case, but high leverage sometimes makes the point influential

Influence from leverage and distance (http://onlinestatbook.com/2/regression/influential.html)

> *An observation's influence is a function of two factors: (1) how much the observation's value on the predictor variable differs from the mean of the predictor variable and (2) the difference between the predicted score for the observation and its actual score. The former factor is called the observation's leverage. The latter factor is called the observation's distance.*

Calculation of Leverage (h) of example i, feature j

formula (https://learnche.org/pid/least-squares-modelling/outliers-discrepancy-leverage-and-influence-of-the-observations#leverage)

$$\begin{aligned} h_j^{(\mathbf{i})} &= \frac{1}{n} + \frac{(\mathbf{x}_j^{(\mathbf{i})} - \bar{\mathbf{x}_j})^2}{\sum_i (\mathbf{x}_j^{(\mathbf{i})} - \bar{\mathbf{x}_j})^2} \\ &= \frac{1 + \left(\frac{\mathbf{x}_j^{(\mathbf{i})} - \bar{\mathbf{x}_j}}{\sigma_{\mathbf{x}_j}} \right)^2}{n} \end{aligned}$$

You can see that the leverage of $\mathbf{x}_j^{(\mathbf{i})}$ depends on the (standardized) distance of $x_j^{(\mathbf{i})}$ from the mean (over all i) of $\mathbf{x}_i$.

Here's an interactive tool to get a feel for influential points.

It allows you to change the value of a single data point and see how the Linear Regression is affected.

Observe how the slope changes (displayed in the title)

- The `x_l` slider chooses the index of the data point to change
- The `y_l` slider chooses how much the data point changes
    - i.e., will change $\mathbf{x}^{(\mathbf{i})}$ when `x_l = i`
- 10 data points

In [4]:

```
# Generate some points
(x_ip,y_ip) = iph.gen_data(10)

# Fit a line to the points; get a function to update the fit and the plot
fit_update = iph.plot_init()
```

In [5]:

```
iph.plot_interact(fit_update)
```

Play around with the example

- choose a point to move using the top slidier
- choose how much to move the chosen point with the bottom slider
- see the effect of the change on the Slope (in the title)

Observe

- changing a point in the middle has little effect on the slope
- changing a point closer to either extreme can have a big effect on the slope

This illustrates the effect of a single example on Θ

Knowing how influential the point is on Θ mave cause you to reduce its influence

- drop the example
    - possible error, outlier
- clip the value (bound the range)

In [8]:

```
print("Done")
```

```
Done
```