# Understanding the Loss function

In performing Error Analysis out of sample, we *identified* **test** examples where our model failed to generalize.

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

Recall the mapping of probability to prediction

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

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

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

As we observed [before (Classification_Loss_Function.ipynb#Classification:-Loss-function)](Classification_Loss_Function.ipynb#Classification:-Loss-function),

- The prediction for example $i$ changes only when probability $\hat{p}^{(\mathbf{i})}$ crosses the threshold.
- Thus: the Performance Metric of Accuracy *won't vary continuously with changes in* $\Theta$
- The **Loss Function** (Cross Entropy) **will** vary continuously with changes in $\Theta$

Therefore, in order to influence prediction through parameters $\Theta$

- We need to focus on the Loss Function
- Which uses **in sample** (Training) examples, rather than out of sample (Test) examples
- In the hope that better in sample performance (lower Loss) leads to better out of sample performance (higher Accuracy)

A set of parameter values $\Theta$ which pushes $\hat{p}^{(\mathbf{i})}$ in the right direction

- Increasing $\hat{p}^{(\mathbf{i})}$ for Positive examples $i$
- Increasing $(1 - \hat{p}^{(\mathbf{i})})$ for Negative examples $i$

is preferred to the set of parameter values $\Theta'$ which pushes the predicted probability $\hat{p}^{(\mathbf{i})}$ in the wrong direction.

In Loss Analysis, we will be applying similar techniques as in Error Analysis

- But using training examples rather than test examples
- Trying to reduce the Loss Function rather than increase the Performance Metric

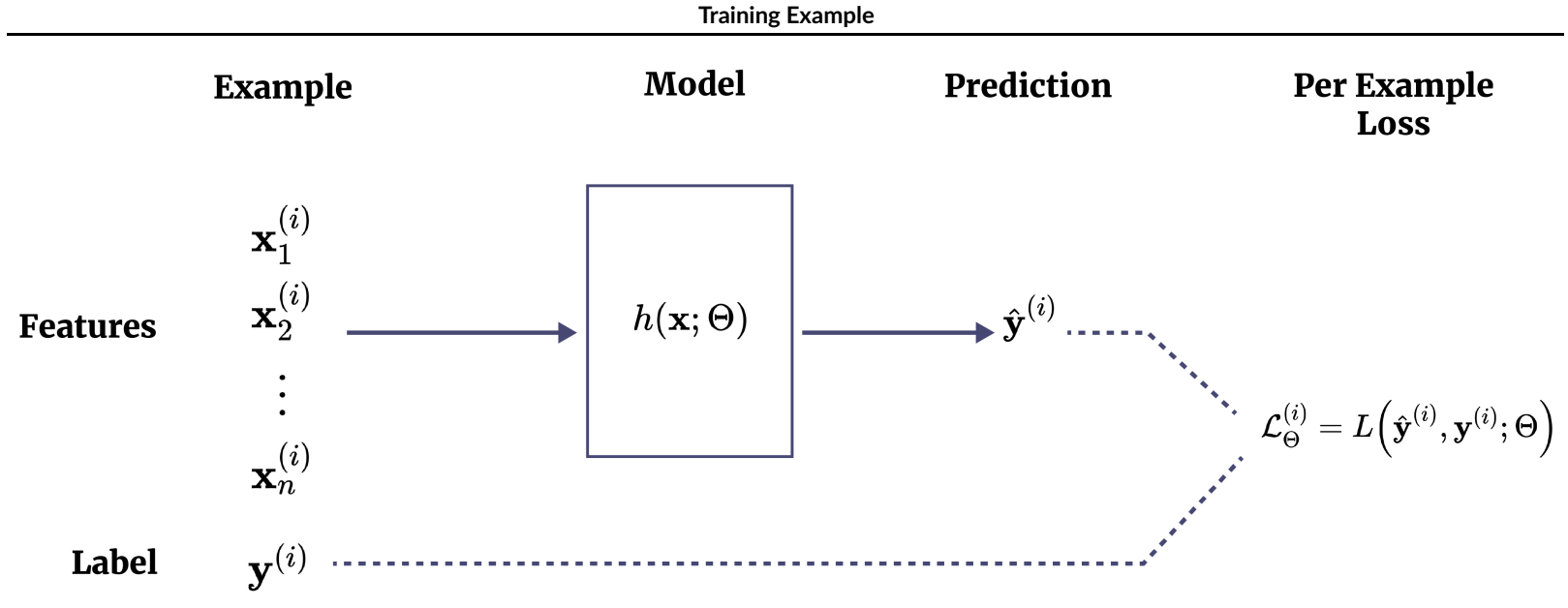
under the assumption that better in sample performance will lead to better out of sample performance

Recall the basics of minimizing Loss Functions

- Predictions $h(\mathbf{x}; \Theta)$ are a function of both inputs and parameters $\Theta$
- A given $\Theta$ induces a per-example loss $\mathcal{L}_{\Theta}^{(\mathbf{i})}$
- Average Loss is the average of the per-examples losses $\mathcal{L}_{\Theta}^{(\mathbf{i})}, i = 1, \ldots, m$
- We seek the optimal $\Theta^*$:

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \mathcal{L}_{\Theta}$$

In pictures:


Training Example
Example
Model
Prediction
Per Example Loss
Features
$\mathbf{x}_1^{(i)}$
$\mathbf{x}_2^{(i)}$
$\vdots$
$\mathbf{x}_n^{(i)}$
$h(\mathbf{x};\Theta)$
$\hat{\mathbf{y}}^{(i)}$
$\mathcal{L}_\Theta^{(i)} = L\left(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)}; \Theta\right)$
Label
$\mathbf{y}^{(i)}$

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

- 

# Conditional loss

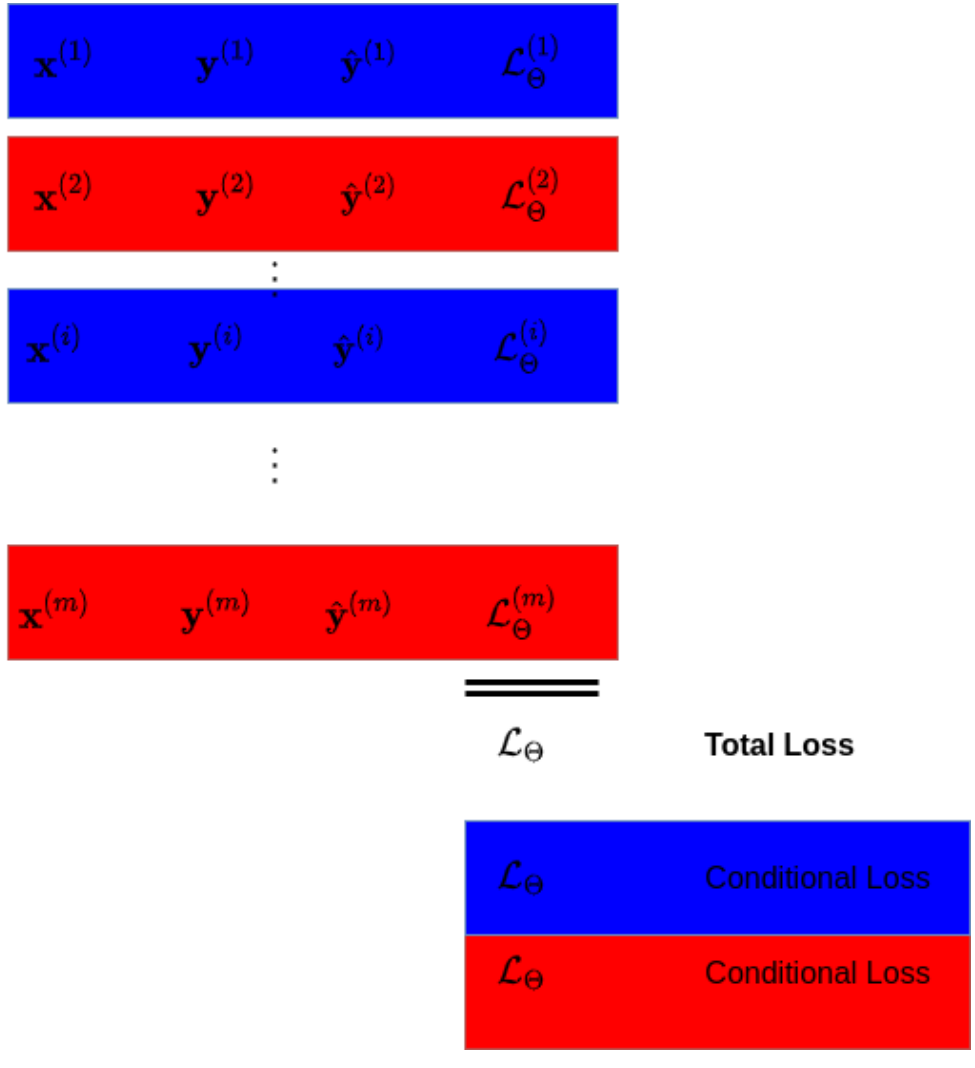
In Error Analysis we partition test examples into groups with some common property, such as

- Commonality of result: TP, FN, TN, FP
- Commonality of features in order to compute a *conditional* out of sample metric.

In Loss Analysis we partition training examples into groups to in order to compute a *conditional* in sample metric.

The following picture uses colors to identify which group a training example belongs to:

## Loss analysis: conditional loss



The real advantage of performing Conditional analysis in sample

- In sample examples (training/validation) can be re-used, unlike Test examples
- Added features based on in sample analysis is likely to affect the Loss
    - Unknown whether it will affect Performance Metric (when it is different than Loss, e.g., Accuracy)

## What can we do to reduce loss ?

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

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

# Increase number of ”problem” training example

For MNIST digit classification

- We hypothesize a commonality that causes images of the digit 8 to be mis-classified
    - 8's that are slanted in the "opposite" direction of normal
    - We will refer to this as the *problematic* class

One reason our classifier may fail on this sub-class of 8's

- There are many fewer of them than the more prevalent images of 8's

Mathematically, the Average Loss is equally weighted

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

but the cumulative weight of the problematic class (mis-shaped 8's) is very small.

So even if all examples in the problematic class were mis-classified

- The impact on Average Loss may be sufficiently small.
- That $\Theta$ doesn't get updated in the direction that will improve these examples
    - Especially if we end optimization before absolute convergence occurs, as is common

One strategy for pushing the model to better fit the problematic examples is

- Increase their cumulative weight in the Loss
- By increasing their number !

The strategy known as *Data Augmentation* adds examples to the Training examples

- Here we try to find/synthesize more instances of the problematic type

# Influential points

We have described the case where the issue is mis-classification of an important but small sub-class.

- Which results in a small cumulative contribution to the Loss

Sometimes the problem is a small sub-class with an *out-sized* contribution to the Loss

- Having a few problem examples
- Whose contribution to Loss is so large
- That it pushes $\Theta$ in the wrong direction for the more numerous non-problem examples

That is: $\mathcal{L}_{\Theta}^{(\mathbf{i})}$ is so large (for some example $i$) that

- $\Theta$ is changed to reduce $\mathcal{L}_{\Theta}^{(\mathbf{i})}$
- Resulting in an increase in $\mathcal{L}_{\Theta}^{(i')}$ for each non-problematic example $i'$

The phenomenon we just described is sometimes called *Influential Points*.

These have been particularly well-studied in the context of Linear Regression.

We will use Linear Regression as an illustration.

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

- the parameter estimate $\Theta$ changes greatly depending on whether the example 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).

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

It allows you to change the value of a single data point and see the effect on the fitted line.

Observe how the slope changes (displayed in the title)

- 10 labelled examples $\{[\mathbf{x^{(i)}}, \mathbf{y^{(i)}}] \,|\, 0 \leq i < 10\}$
- The top slider chooses the index $i \in \{0 \ldots 9\}$ of one data point to change
- The bottom slider is the new value $\mathbf{y^{(i)}}$ for the point at the chosen index $i$

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 tool

- Choose an index near the middle (set top slider to a value of around 5)
    - Move the bottom slider, observing the effect on the fitted line (slope displayed in the title)
- Choose an index at either extreme (index 0 or 9)
    - Move the bottom slider, observing the effect on the fitted line (slope displayed in the title)

Observe

- Changing $\mathbf{y^{(i)}}$ for $i$ near the middle of the range has little effect on the fit
- Changing $y^{(\mathbf{i})}$ for $i$ near either end ($0$ or $9$) has a large effect on the fit

The solution is to somehow reduce example $i$'s contribution to Average Loss

- Removing the example: possible data error or outlier
- Down-weighting
- Clipping the values of the features/target to some upper bound

# Further background

Consider feature $j$.

The **leverage** of example $i$ is related to

- How far $\mathbf{x}_j^{(\mathbf{i})}$ is from $\bar{\mathbf{x}}$, the average of $\mathbf{x}_j$ across all examples

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

Reference: 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 $\mathbf{x}_j^{(\mathbf{i})}$ from the mean (over all $i$) of $\mathbf{x}_i$.

In [6]:
```
print("Done")
```

```
Done
```