## **Classification: Loss function**

It would be natural to expect the Average Loss to be Accuracy (fraction of correct predictions).

On a per example basis, the corresponding loss  $\mathcal{L}^{(i)}$  would be either 1 or 0, depending on correctness.

This is not the case.

Recall the mapping of probability to prediction

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

The prediction for example i changes only when probability  $\hat{p}^{(i)}$  crosses the threshold. Suppose the class for example i is Positive:  $\mathbf{y^{(i)}} = \text{Positive}$ .

• Is our model "better" when

$$\hat{p}^{(\mathbf{i})} pprox 1 \qquad ext{than when} \qquad \hat{p}^{(\mathbf{i})} = 0.5 \ \hat{p}^{(\mathbf{i})} = (.5 - \epsilon) \qquad ext{than when} \qquad \hat{p}^{(\mathbf{i})} pprox 0$$

- The per-example Accuracy is the same in both comparisons
- But a model with probability  $\hat{p}^{(i)}$  closer to 1 for a Positive example i would seem to better

There is no *degree* or magnitude of inaccuracy

- Two models may have the same Accuracy even though the probabilities of one may be closer to perfect than the other
- In our search for the best  $\Theta$ , Accuracy won't be a guide

In mathematical terms: we want our Loss function be be continuous and differentiable.

Accuracy (and the per-example analog) satisfies neither.

We will introduce Binary Cross Entropy loss to overcome this difficulty.

Think of Binary Cross Entropy as a continuous analog of Accuracy.

## **Binary Cross Entropy**

Let's encode the Positive labels  $\mathbf{y^{(i)}}$  with the number 1 and Negative labels with the number 0. The loss for example i will be defined as  $\$  \loss^\ip\_\Theta = \begin{cases}

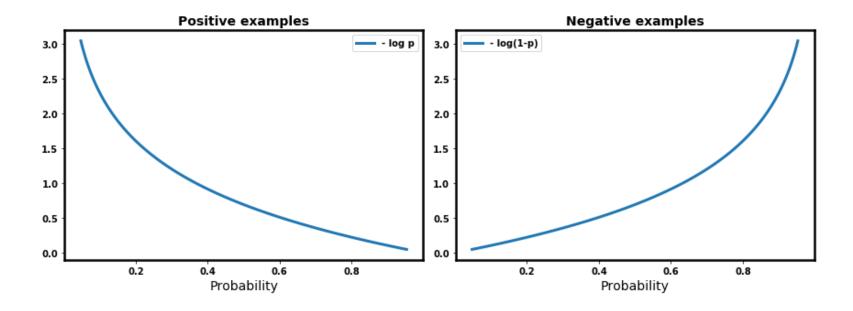
- \log(\hat{p}) & \textrm{if } & \y^\ip = 1 \
- \log(1-\hat{p}) & \textrm{if } & \y^\ip = 0 \ \end{cases} \$\$

Note the negative signs:

The term being negated is a Utility (which we want to maximize)

A plot will give us some intuition.

In [4]: svmh.plot\_log\_p(x\_axis="Probability")



- For Positive examples: the loss approaches 0 as the predicted probability approaches the correct value (1).
- For Negative examples: the loss approaches () as the predicted probability approaches the correct value (()).

In a Deep Dive (after the introduction of a bit of math) we will gain a greater appreciation it's meaning.

For now: be content that Binary Cross Entropy seems to have the right slope and asymptotic behavior.

Because only one of  $\mathbf{y^{(i)}}$  and  $(1-\mathbf{y^{(i)}})$  is non zero, we can re-write the two-case statement into a single expression

$$\mathcal{L}_{\Theta}^{(\mathbf{i})} = -\left(\mathbf{y^{(i)}} * \log(\hat{p}^{(\mathbf{i})}) + (1 - \mathbf{y^{(i)}}) * \log(1 - \hat{p}^{(\mathbf{i})})\right)$$

This expression is referred to as *Binary Cross Entropy*; it and the multi-class version will become quite familiar going forward.

The Loss for the entire training set is simply the average (across examples) of the Loss for the example

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

## Cost function for Multinomial Classification: Cross Entropy

A Multinomial Classifier (when categories/classes ||C||>2) can be created from multiple Binary Classifiers

- Create a separate Binary Classifier for each  $c \in C$
- ullet The classifier for category c attempts to classify
  - Each example with target category of c as Positive
  - All other examples as Negative
- ullet Combine the ||C|| classifiers to produce a vector  $\hat{p}$  of length ||C||
  - ullet normalize across  $c \in C$  to sum to 1
  - ullet  $\hat{p}_c$  denotes the normalized value for category c
    - Notation abuse: subscripts should be integers, not categories

Both the target  ${f y}$  and the prediction  $\hat p$  are represented as vectors of length ||C||

- We write  $\mathbf{y}_c, \hat{p}_c$  to denote the element of the vector corresponding to category c
- Each vector can be interpreted as a probability distribution, e.g.

$$orall c \in C: \mathbf{y}_c \geq 0$$

$$\sum_{c \in C} \mathbf{y}_c = 1$$

- y was created with One Hot Encoding (OHE), so properties satisfied by construction
- $\hat{p}_c$  satisfies the properties by virtue of the normalization of the predictions of the ||C|| binary classifiers

With  $\mathbf{y},\hat{p}$  encoded as a vectors, per example Binary Cross Entropy can be generalized to  $||C|| \geq 2$  categories:

$$\mathcal{L}_{\Theta}^{(\mathbf{i})} = -\sum_{c=1}^{||C||} \left(\mathbf{y}_c^{(\mathbf{i})} * \log(\hat{\mathbf{p}}_c^{(\mathbf{i})})
ight)$$

This is the multinomial analog of Binary Cross Entropy and is called **Cross Entropy**.

Cross Entropy can be interpreted as a measure of the "distance" between distributions  ${f y}$  and  $\hat p$ 

- Minimized when they are identical
- We will use Cross Entropy in the future both as a Loss function and a way of comparing probability distributions

```
In [5]: print("Done")
```

Done