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 $\mathcal{L}^{(\mathbf{i})}$ would be either 1 or 0, depending on correctness.

This is not the case.

Consider the prediction

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

The prediction (and hence, 1 or 0 per-example loss) only changes when $\hat{p}^{(\mathbf{i})}$ crosses the threshold.

This means that unless we predict $\hat{p}^{(\mathbf{i})} = 1$ for a Positive example, we are *inaccurate*

- There is no *degree* of inaccuracy
- We are equally incorrect whether $\hat{p}^{(\mathbf{i})} = (1 - .0001)$ or $\hat{p}^{(\mathbf{i})} = .0001$
- This makes it hard for the optimizer to update estimates of $\hat{p}^{(\mathbf{i})}$ as it gets no feedback that the estimate has improved

I 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

The loss for example i will be defined as $$ \loss^\ip_\Theta = \begin{cases}

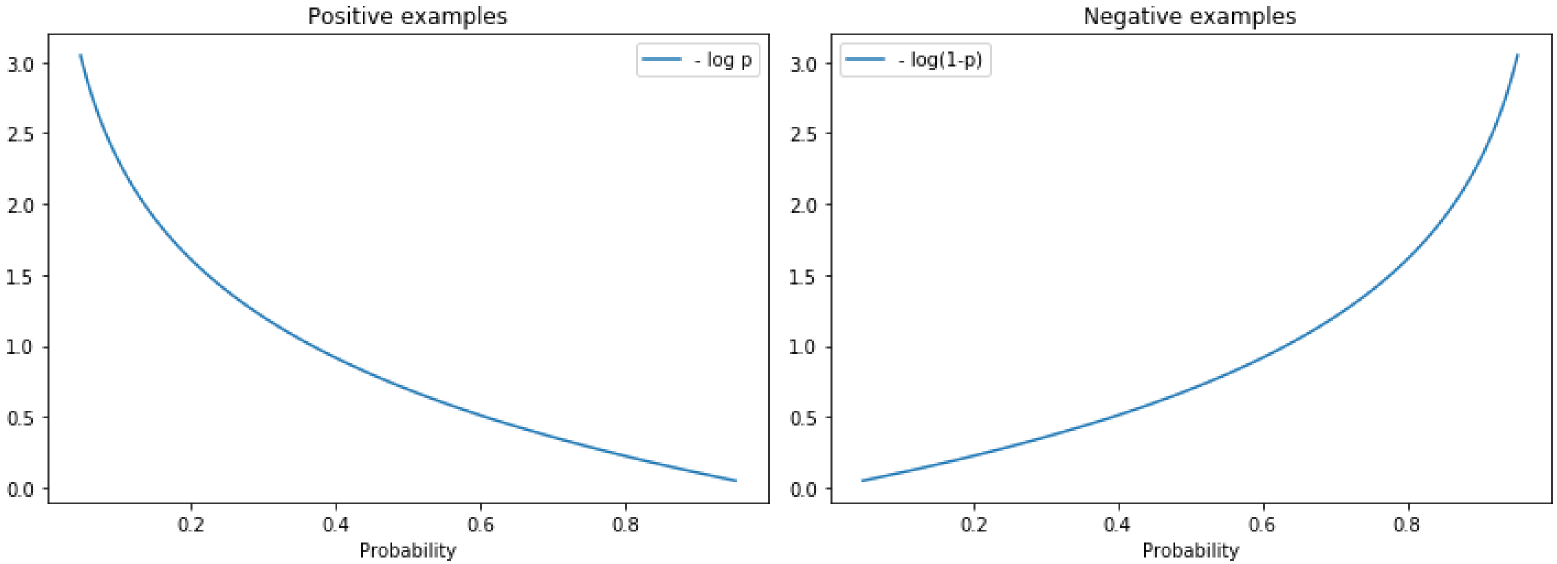
- \log(\hat{p}) & \textrm{if } & \y^\ip = 1 \
- \log(1-\hat{p}) & \textrm{if } & \y^\ip = 0 \ \end{cases} $$

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 Negtive examples: the loss approaches 0 as the predicted probability approaches the correct value (0).

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.

Let's encode the Positive labels $\mathbf{y^{(i)}}$ with the number 1 and Negative labels with the number 0.

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

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

Cost function for Multinomial Logistic Classification: Cross Entropy

Let us represent target $\mathbf{y^{(i)}}$

- as a *vector* of 1's and 0's of length $||C||$
- with exactly 1 non-zero element
- if example i's target is the j^{th} element of C
    - $\mathbf{y}_j^{\mathbf{(i)}} = 1$
- the sum of the elements of the representation is 1.

This representing of 1 out of $||C||$ is called *One Hot Encoding (OHE)*.

With the target now encoded as a vector $\mathbf{y}$ we can write the cost function per example as

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

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

This is called the **Cross Entropy** Cost Function.

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

```
Done
```