

# Support Vector Classifier: Loss function

In concept, the SVC is quite similar to the Logistic Regression model.

The main difference between the two is the Loss function

- Cross Entropy for Logistic Regression
- Hinge Loss for the Support Vector Classifier

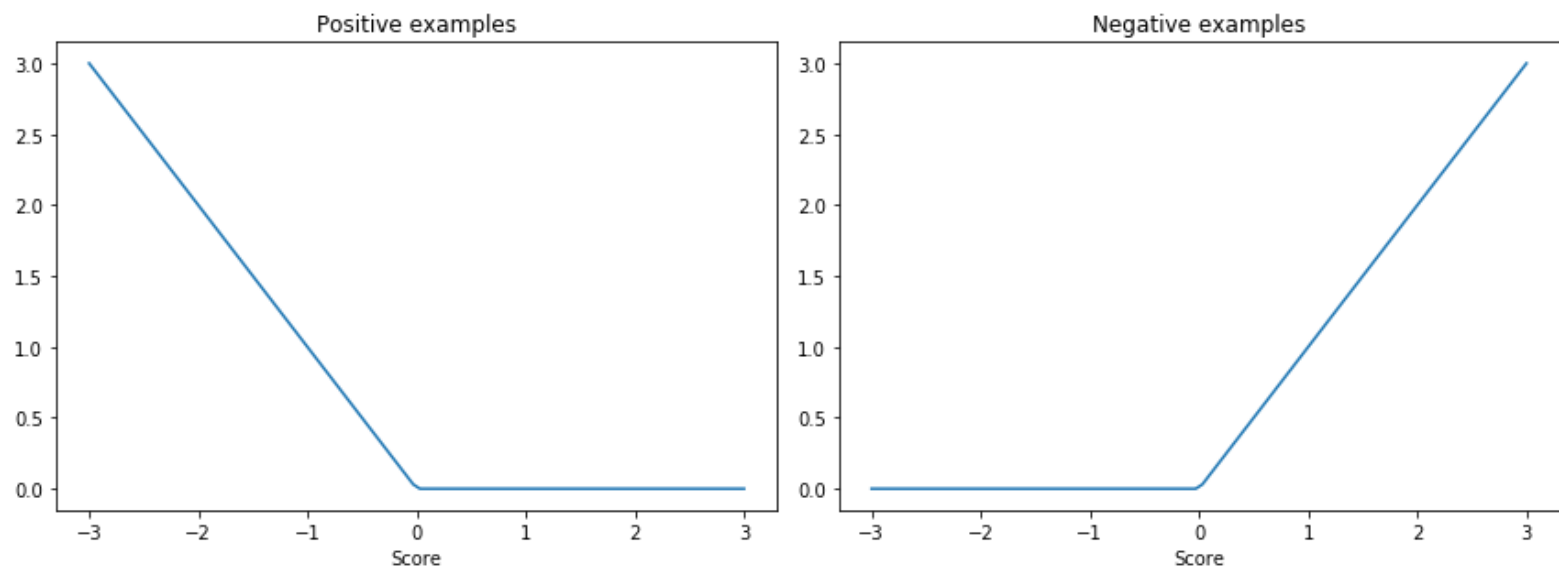
It is the Hinge Loss that makes this model quite interesting.

## Hinge Loss function

The Hinge Loss function is best described by a plot.

Here are the two sides of the per-example Hinge Loss

```
In [4]: svmh.plot_hinges()
```



That is: it is a function of a "score"  $\hat{s}$

- for Positive examples: the loss is  $\max(0, -\hat{s})$
- for Negative examples: the loss is  $\max(0, \hat{s})$

The plot resembles a *hinge*.

# SVC Loss versus Binary Cross Entropy

For Binary Logistic Regression

- we computed a score  $s$  as a linear function of the features
- we converted the linear score into a probability via the logistic function

$$\hat{p}^{(i)} = \sigma(s(\hat{\mathbf{x}}^{(i)}))$$

By encoding the Positive labels  $\mathbf{y}^{(i)}$  with the number 1 and Negative labels with the number 0

- we were able to combine the two sides (Positive, Negative) of the per-example loss into a single equation

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

This is the equation for per-example Binary Cross Entropy Loss.

For the Binary SVC:

- we compute a score as linear function of the features
- we use Hinge Loss instead of Log Loss

By analogy with Cross Entropy, we can combine the two sides (Positive, Negative) of the per-example loss into a single equation

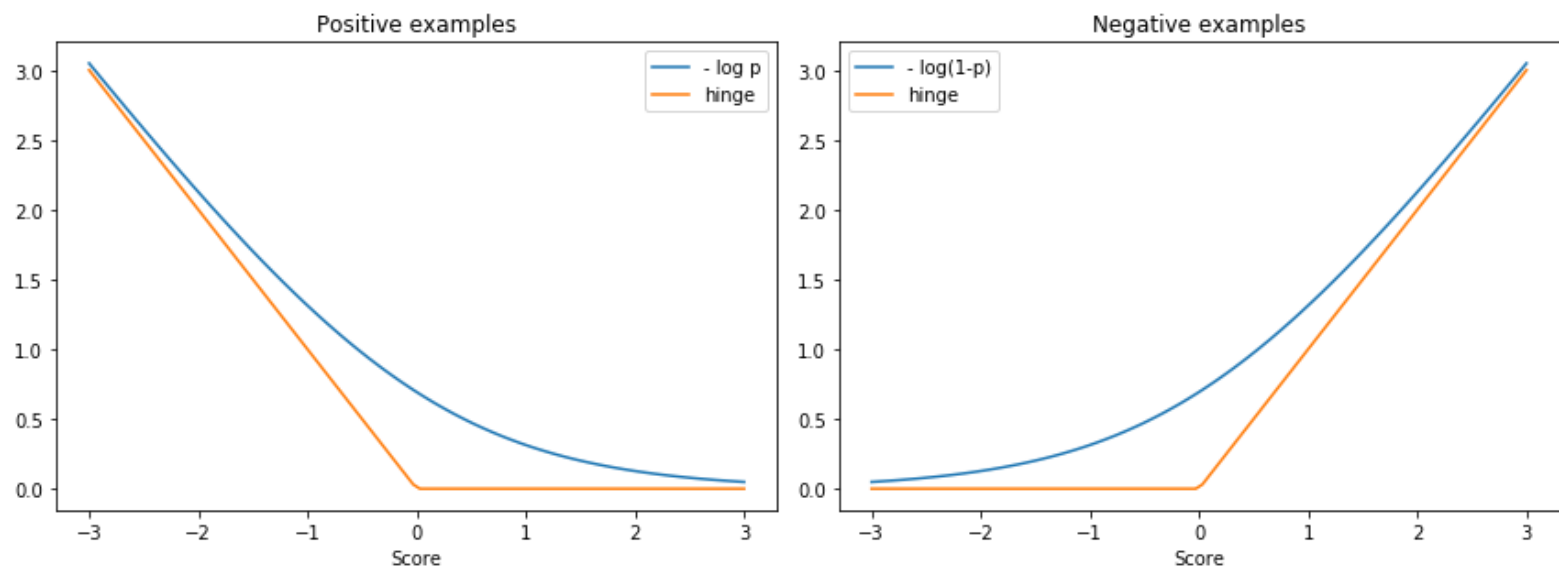
$$\mathcal{L}^{(i)} = \left( \mathbf{y}^{(i)} \max(0, -s(\hat{\mathbf{x}})) + (1 - \mathbf{y}^{(i)}) \max(0, s(\hat{\mathbf{x}})) \right)$$

You can see the similarity with Cross Entropy.

The similarity between the SVM Classification Loss and Cross Entropy becomes more apparant if we plot them together

- Note: the horizontal scale for the Cross Entropy plots are  $\hat{p}$  rather than  $\hat{s}$

```
In [5]: svmh.plot_log_p(x_axis="Score", hinge_pt=0)
```





For SVC loss

- We can eliminate the asymmetry in the two sides
- With a slightly different encoding of Positive/Negative
- Into integers +1 and -1 (rather than +1 and 0)

To make this unusual encoding clear, we will place a "dot" over  $\mathbf{y}$

$$\dot{\mathbf{y}}^{(i)} = \begin{cases} +1 & \text{if Positive } \mathbf{y}^{(i)} \\ -1 & \text{if Negative } \mathbf{y}^{(i)} \end{cases}$$

This allows us to simplify the per-example SVC loss to

$$\mathcal{L}^{(i)} = \max(0, -\dot{\mathbf{y}}^{(i)} * s(\hat{\mathbf{x}}))$$

This is the equation for per-example Hinge Loss, when the "hinge point" is 0.

# Hinge Loss interpretation

From the plot of Log Loss versus Hinge Loss, we can see the similarity.

The key difference is that

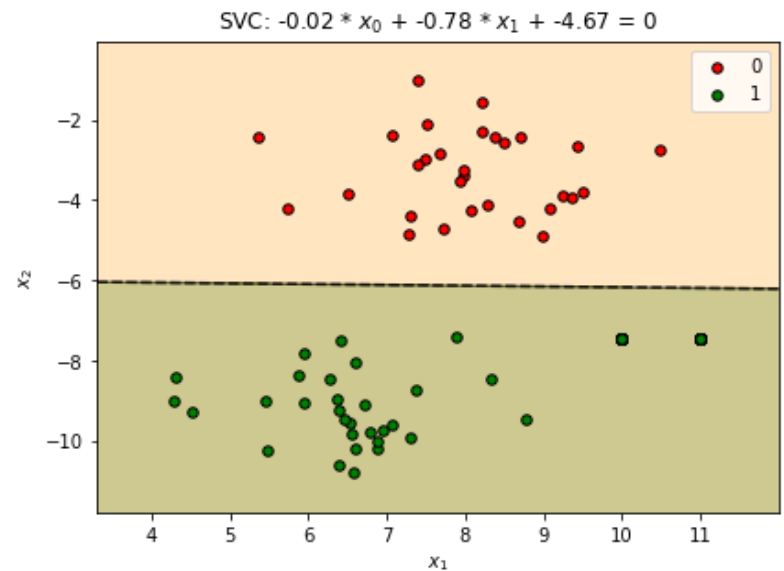
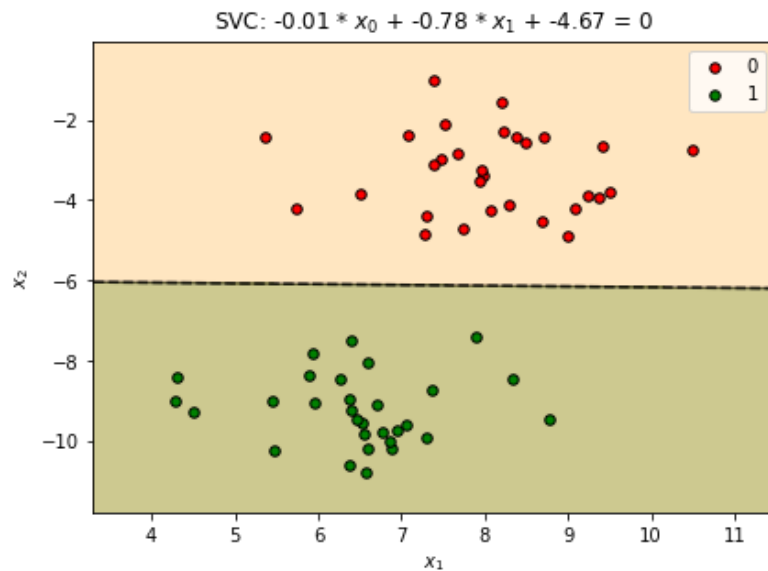
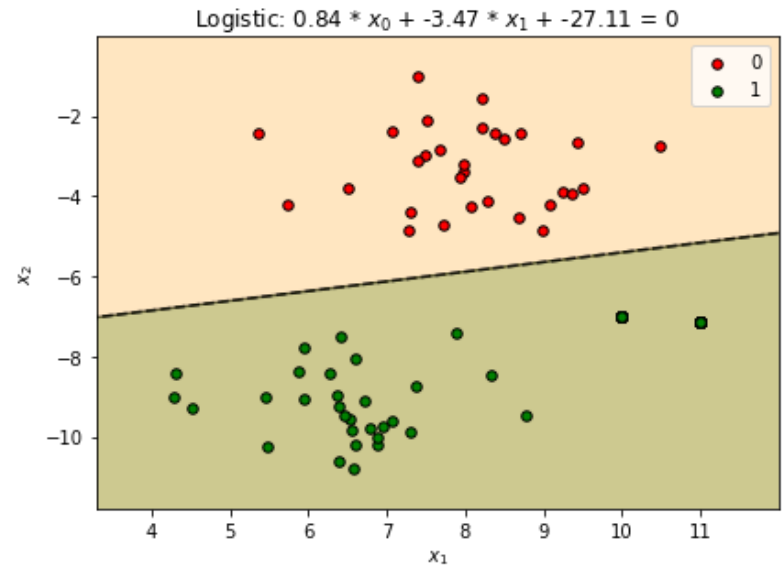
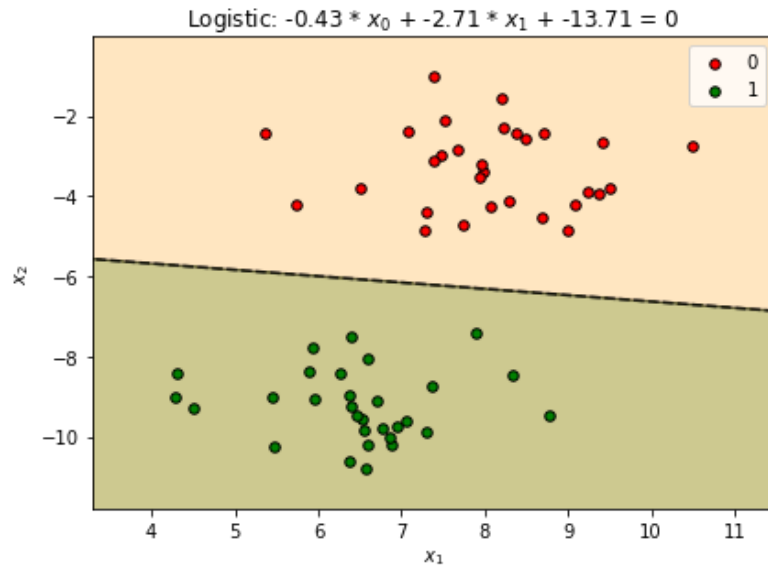
- A correctly classified example has a per-example Hinge Loss of 0
- A correctly classified example has a positive per-example Log Loss

An optimizer seeking the  $\Theta$  that minimizes Average Loss will be sensitive to non-zero per-example loss.

- Using Log Loss: once an example is correctly classified, the example contributes to Average Loss
- Using Hinge Loss: once an example is correctly classified, the example *does not* contribute to Average Loss.

Let's see this effect in practice.

```
In [6]: svm_ch = svm_helper.Charts_Helper()  
_ = svm_ch.create_data()  
fig, axs = svm_ch.create_sens()
```





The chart compares Logistic Regression to SVC on an original and augmented set of examples

- The original examples are the plots on the left
- The original examples are augmented by a cluster of examples and plotted on the right
  - the new examples are correctly classified and located just below the boundary near the right edge
  - although hard to see: there are *many* instances of each added example (all identical)



The additional examples are relatively close to the separating boundary.

- For Logistic Regression:
  - each example incurs a relatively high Log Loss  $\mathcal{L}^{(i)}$  since it is close to the boundary
  - there are a lot of such examples, each contributing a positive amount to Average Loss

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

- minimizing Average Loss when these new examples are present means moving the boundary away from them

For SVC:

- The additional examples are on the correct side of the boundary and incur zero Hinge Loss
- Hence the additional examples do not affect the fit.

The key difference

- Once the Hinge Loss for an example reaches 0
- There is no benefit (i.e., reduction of Average Loss)
- To improving the parameters to make the example be "further" from the boundary

For Cross Entropy Loss

- There is always benefit until per example loss reaches 0
- Hence, in the absence of other constraints, the optimizer will try to "improve" the fit

For a Classification task

- Cross Entropy Loss continues to try to improve the probability  $\hat{p}^{(i)}$
- Long after  $\hat{p}^{(i)}$  has crossed the prediction threshold (e.g., 0.5)
  - this might lead to overfitting (high variance)
- Hinge Loss will not try to improve prediction once we cross the threshold
  - this might lead to a fit that is "good" but not "best" (high bias)

For a Classification task

- Cross Entropy Loss continues to try to improve the probability  $\hat{p}^{(i)}$  long after  $\hat{p}^{(i)}$  has crossed the prediction threshold (e.g., 0.5)
  - this might lead to overfitting (high variance)
- Hinge Loss will not try to improve prediction once we cross the threshold
  - this might lead to a fit that is "good" but not "best" (high bias)

In [7]: `print("Done")`

Done