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

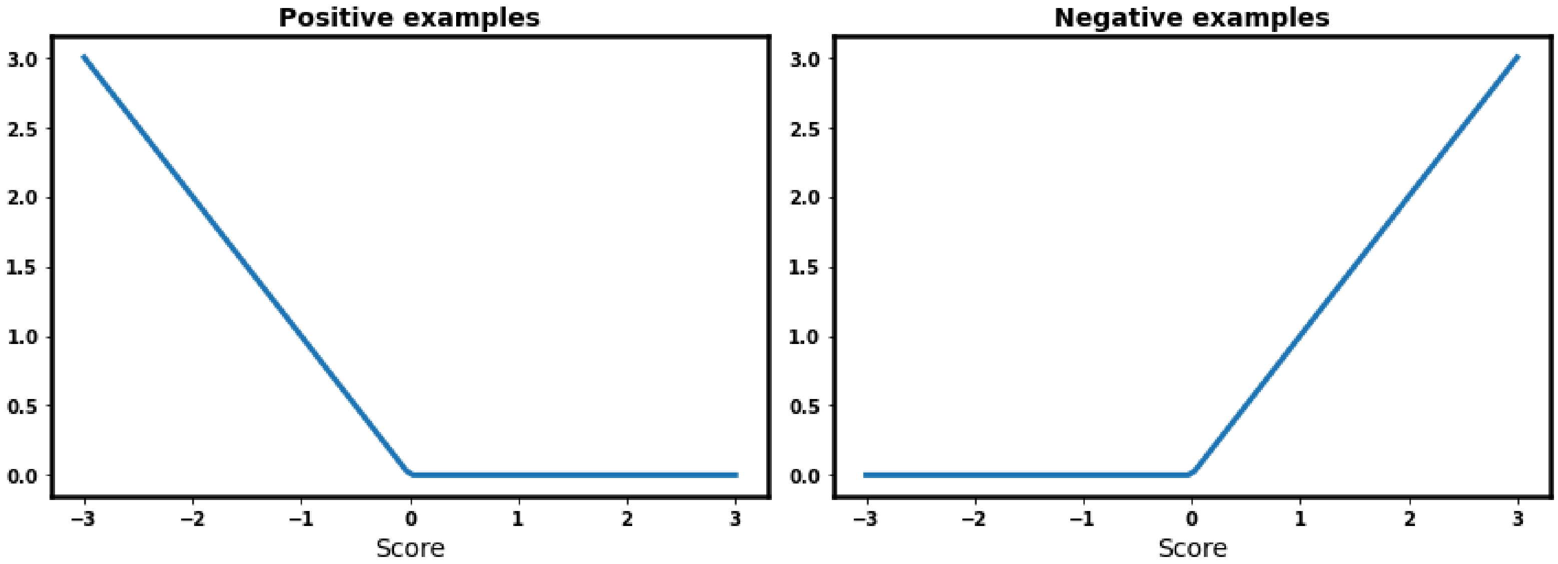
It is the Hinge Loss that makes this model quite interesting.

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*.

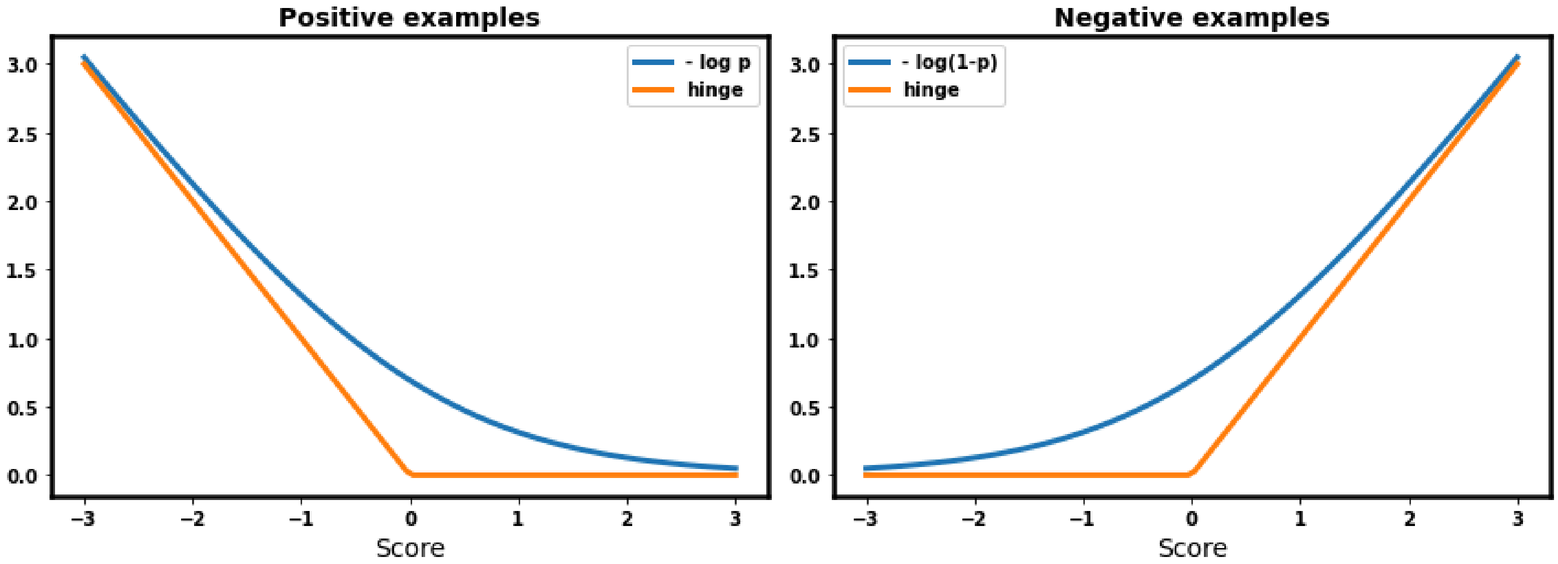
The similarity between

- the two-side Hinge Loss and
- the two-side loss of Cross Entropy (used in Logistic Regression)

becomes more apparent 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)
```



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

By encoding the Positive labels $\mathbf{y}^{(\mathbf{i})}$ 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
$$\mathcal{L}^{(\mathbf{i})} = - \left(\mathbf{y}^{(\mathbf{i})} * \log(\hat{p}^{(\mathbf{i})}) + (1 - \mathbf{y}^{(\mathbf{i})}) * \log(1 - \hat{p}^{(\mathbf{i})}) \right)$$

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

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

You can see the similarity with Cross Entropy.

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 $\mathbf{y}$

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

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

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

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

Hinge Loss interpretation

From the previous plot of Cross Entropy Loss (log p) versus Hinge Loss, we can see the similarity.

The key difference is that

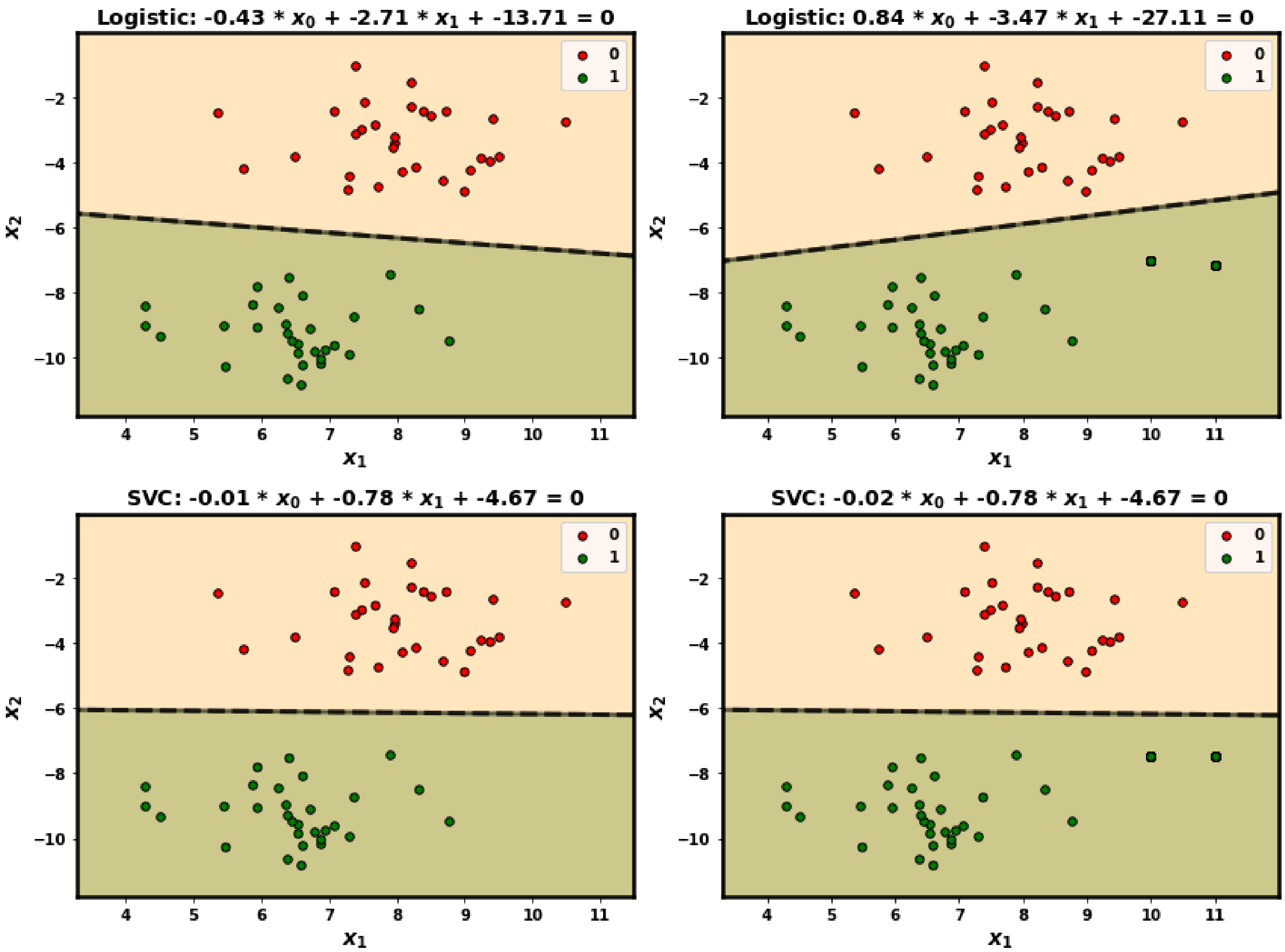
- A correctly classified example has a per-example Hinge Loss of 0
- A correctly classified example has a positive per-example Log Loss

An optimizer seeking the Θ 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()
```


Logistic: -0.43 * x_0 + -2.71 * x_1 + -13.71 = 0
Logistic: 0.84 * x_0 + -3.47 * x_1 + -27.11 = 0
SVC: -0.01 * x_0 + -0.78 * x_1 + -4.67 = 0
SVC: -0.02 * x_0 + -0.78 * x_1 + -4.67 = 0
x_1
x_2
0
1

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 $\mathcal{L}^{(\mathbf{i})}$ since it is close to the boundary
    - There are a lot of such examples, each contributing a positive amount to Average Loss

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

    - 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

- One 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 $\hat{p}^{(\mathbf{i})}$ long after $\hat{p}^{(\mathbf{i})}$ 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)

More formally: let

- $\langle \mathbf{X}, \mathbf{y} \rangle$ denote the training dataset used in the graph on the left
- $\langle \mathbf{X}', \mathbf{y}' \rangle$ denote the set of *additional* training examples added to $\langle \mathbf{X}, \mathbf{y} \rangle$ in the graph on the right

The total loss for the graph on the right is

$$\mathcal{L} = \sum_{i=1}^{\|\mathbf{X}\|} \mathcal{L}^{(\mathbf{i})}(\mathbf{X}^{(\mathbf{i})}, \mathbf{y}^{(\mathbf{i})}) + \sum_{i=1}^{\|\mathbf{X}'\|} \mathcal{L}^{(\mathbf{i})}(\mathbf{X}'^{(\mathbf{i})}, \mathbf{y}'^{(\mathbf{i})})$$

The increase in total loss resulting from the additional training examples is

$$\sum_{i=1}^{\|\mathbf{X}'\|} \mathcal{L}^{(\mathbf{i})}(\mathbf{X}'^{(\mathbf{i})}, \mathbf{y}'^{(\mathbf{i})}) \geq 0$$

Using Log Loss

- the per-example loss $\mathcal{L}^{(\mathbf{i})}$ is positive for each example in the sum above
- so the total increase accumulates the $\|\mathbf{X}'\|$ additional positive per-example losses
- potentially forcing the optimizer to shift the separating boundary to account for the increase in total loss

Using Hinge Loss

- the per-example loss $\mathcal{L}^{(\mathbf{i})}$ equals 0 (since additional example i in $\mathbf{X}'$ is correctly classified by the original separating boundary)
- hence, the total loss is unchanged
    - and so is the separating boundary

In [7]:

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

```
Done
```