loss-functions-implementation

March 4, 2024

[1]:

```python
import matplotlib.pyplot as plt
import numpy as np
```

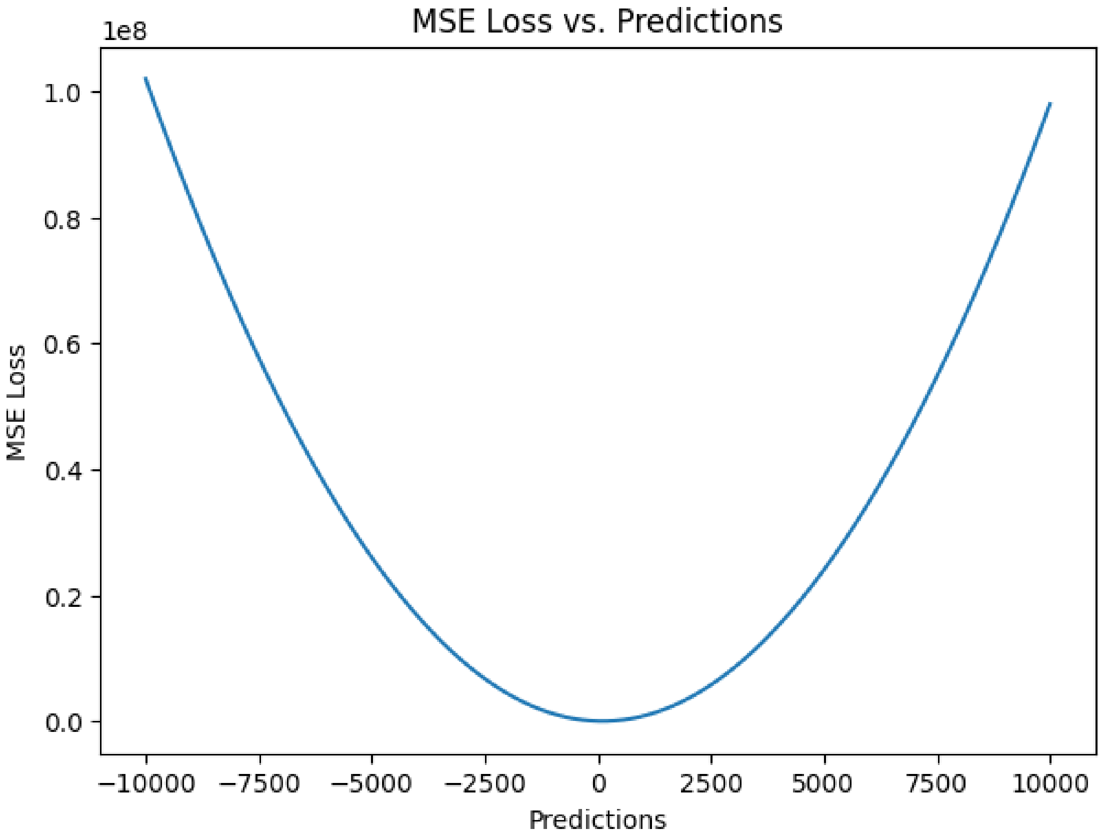
0.1 Regression losses

[2]:

```python
# Mean Squared Error (MSE) or L2 Loss
def mse_loss(true, pred):
    return np.sum((true - pred)**2)
```

[3]:

```python
target = np.repeat(100, 10000)
pred = np.arange(-10000, 10000, 2)
loss_mse = [mse_loss(target[i], pred[i]) for i in range(len(pred))]
plt.figure(figsize=(7,5))
plt.xlabel("Predictions")
plt.ylabel("MSE Loss")
plt.title("MSE Loss vs. Predictions")
plt.plot(pred, loss_mse)
plt.show()
```

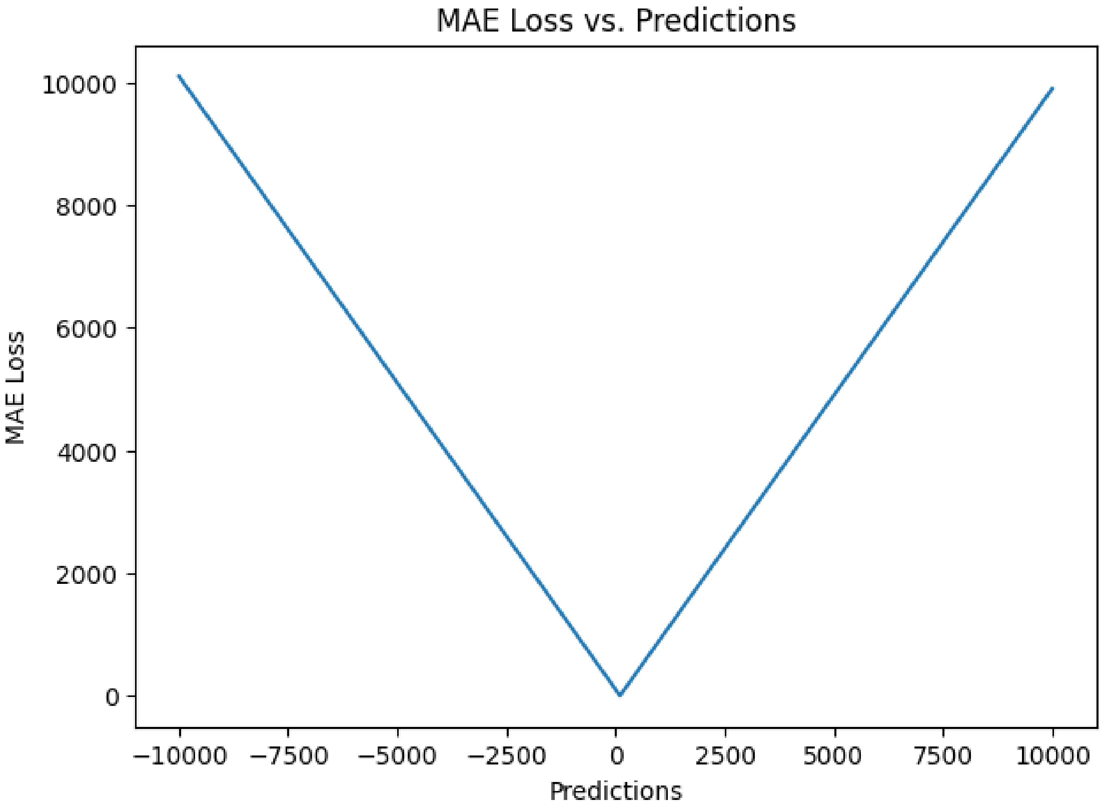


[4]:

```
# Mean Absolute Error (MAE) or L1 Loss
def mae_loss(true, pred):
    return np.sum(np.abs(true - pred))
```

[5]:

```
target = np.repeat(100, 10000)
pred = np.arange(-10000, 10000, 2)
loss_mae = [mae_loss(target[i], pred[i]) for i in range(len(pred))]
plt.figure(figsize=(7,5))
plt.xlabel("Predictions")
plt.ylabel("MAE Loss")
plt.title("MAE Loss vs. Predictions")
plt.plot(pred, loss_mae)
plt.show()
```

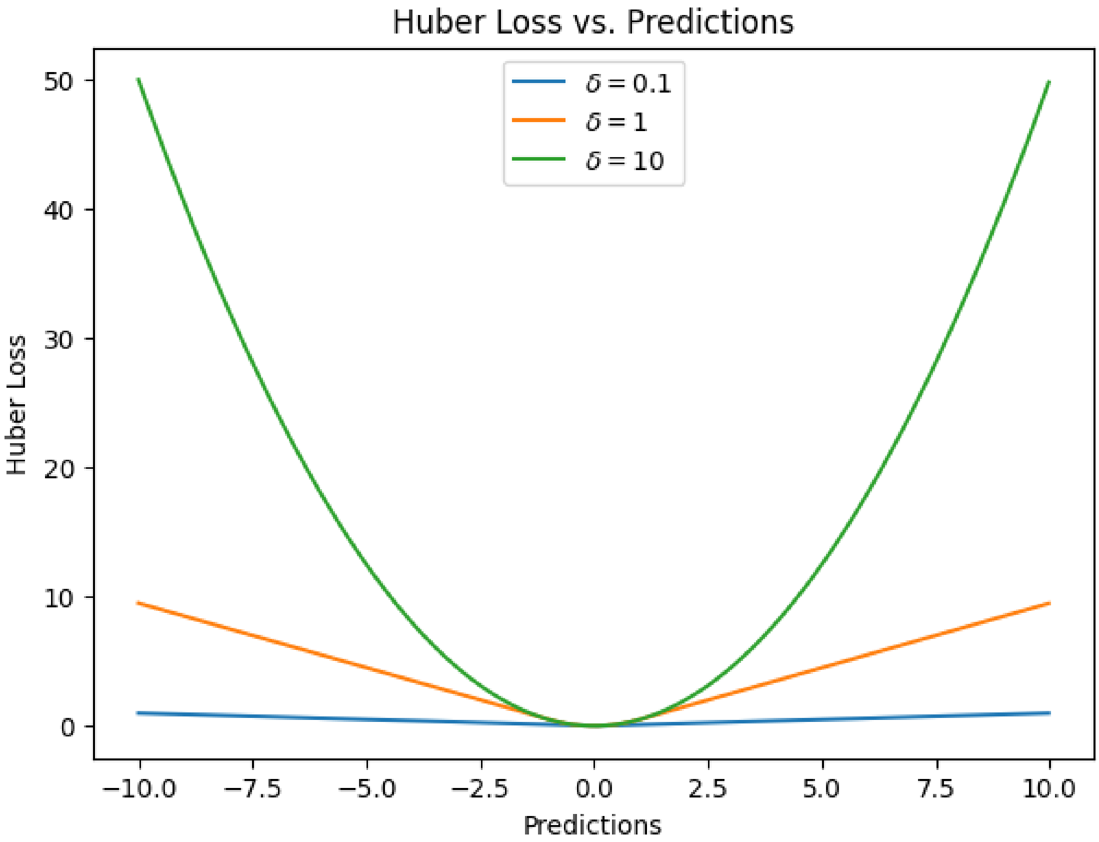


[6]:

```python
# Huber Loss or Smooth L1 Loss
def huber_loss(true, pred, delta):
    loss = np.where(np.abs(true - pred) < delta, 0.5*((true - pred)**2), delta*np.abs(true - pred) - 0.5*(delta**2))
    return np.sum(loss)
```

[7]:

```python
target = np.repeat(0, 1000)
pred = np.arange(-10, 10, 0.02)
delta = [0.1, 1, 10]
losses_huber = [[huber_loss(target[i], pred[i], q) for i in range(len(pred))] for q in delta]
plt.figure(figsize=(7,5))
for i in range(len(delta)):
    plt.plot(pred, losses_huber[i], label=f"$\delta={delta[i]}$")
plt.xlabel("Predictions")
plt.ylabel("Huber Loss")
plt.title("Huber Loss vs. Predictions")
plt.legend()
plt.show()
```

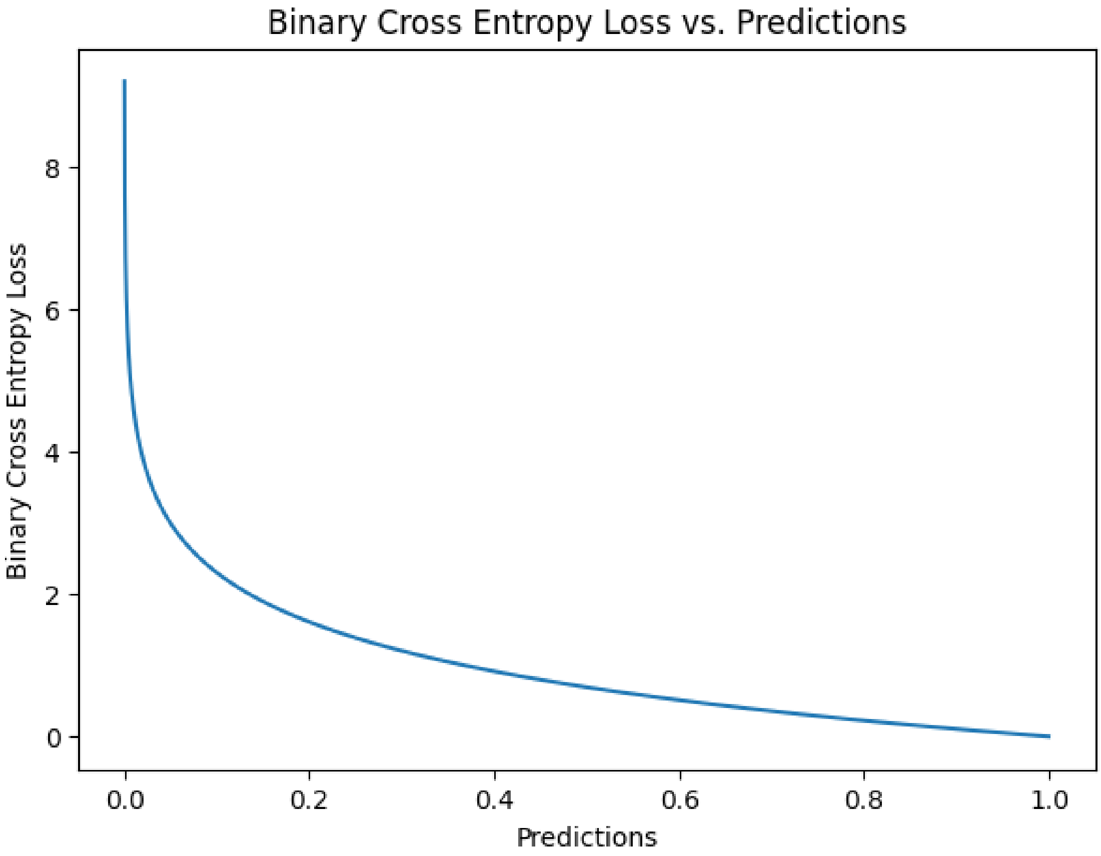


0.2 Binary Classification loss functions

[8]:
```
# Binary Cross Entropy
def bin_ce(true, pred):
    loss = np.where(true == 1, np.log(pred), np.log(1-pred))
    return -np.sum(loss)
```

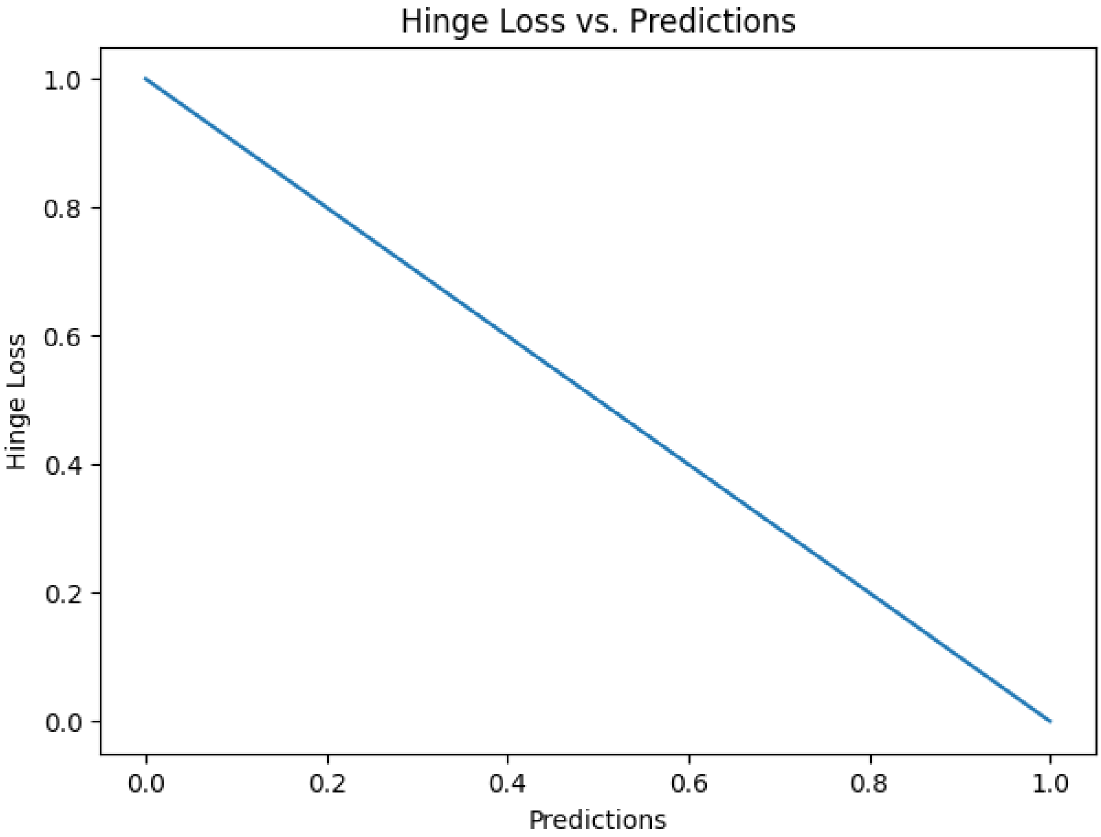
[9]:
```
target = np.repeat(1, 10000)
pred = np.arange(0, 1, 0.0001)
loss_bin_ce = [bin_ce(target[i], pred[i]) for i in range(len(pred))]
plt.figure(figsize=(7,5))
plt.xlabel("Predictions")
plt.ylabel("Binary Cross Entropy Loss")
plt.title("Binary Cross Entropy Loss vs. Predictions")
plt.plot(pred, loss_bin_ce)
plt.show()
```

```
/tmp/ipykernel_33/2028509297.py:3: RuntimeWarning: divide by zero encountered in
log
  loss = np.where(true == 1, np.log(pred), np.log(1-pred))
```



```
[10]: # Hinge Loss
      def hinge_loss(true, pred):
          loss = np.max((0, (1 - pred*true)))
          return np.sum(loss)
```

```
[11]: target = np.repeat(1, 10000)
      pred = np.arange(0, 1, 0.0001)
      loss_hinge = [hinge_loss(target[i], pred[i]) for i in range(len(pred))]
      plt.figure(figsize=(7,5))
      plt.xlabel("Predictions")
      plt.ylabel("Hinge Loss")
      plt.title("Hinge Loss vs. Predictions")
      plt.plot(pred, loss_hinge)
      plt.show()
```



0.3 Multi-class Classification loss functions

[12]:

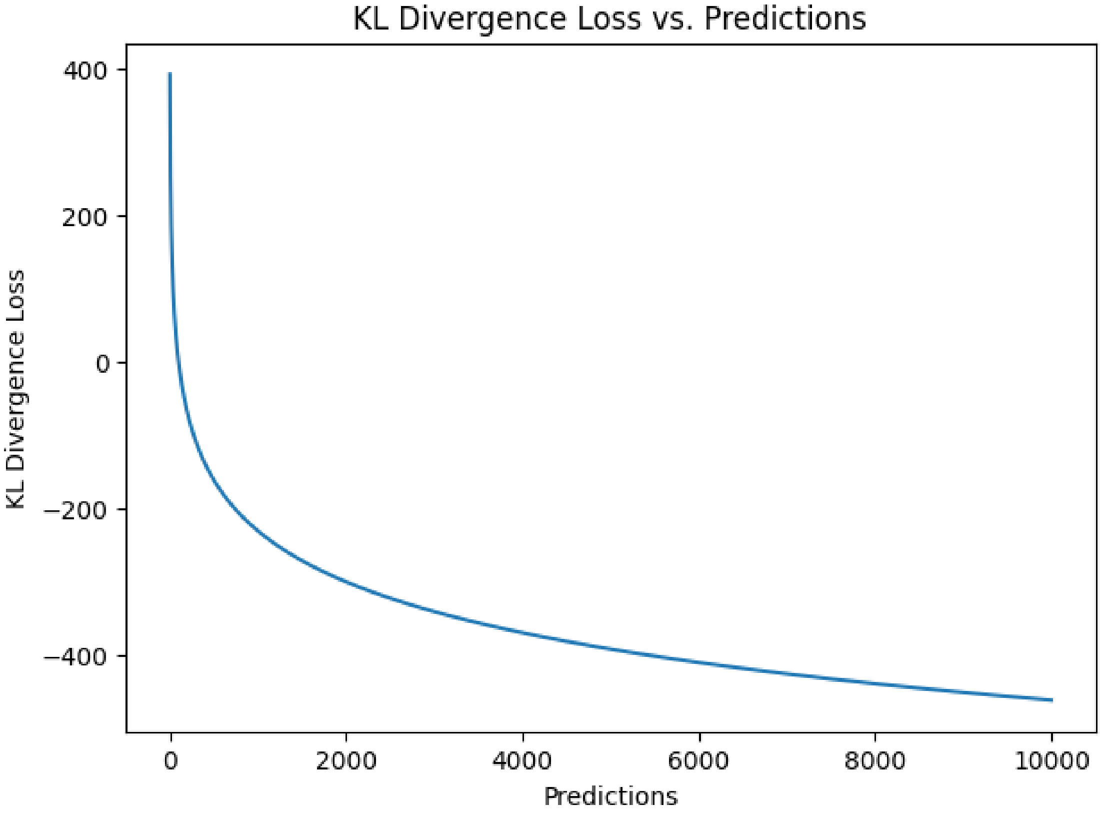
```python
# Kullback-Leibler Divergence
def kl_divergence_loss(true, pred):
    return np.sum(true * np.log(true/pred))
```

[13]:

```python
target = np.repeat(100, 10000)
pred = np.arange(-10000, 10000, 2)
kl_divergence_losses = [kl_divergence_loss(target[i], pred[i]) for i in range(len(pred))]
plt.figure(figsize=(7,5))
plt.xlabel("Predictions")
plt.ylabel("KL Divergence Loss")
plt.title("KL Divergence Loss vs. Predictions")
plt.plot(pred, kl_divergence_losses)
plt.show()
```

```
/tmp/ipykernel_33/4103857248.py:3: RuntimeWarning: invalid value encountered in
log
  return np.sum(true * np.log(true/pred))
```

```
/tmp/ipykernel_33/4103857248.py:3: RuntimeWarning: divide by zero encountered in
scalar divide
  return np.sum(true * np.log(true/pred))
```


KL Divergence Loss vs. Predictions
KL Divergence Loss
Predictions