DATA MINING

/CISC 873 - Steven Ding
/Metric/Loss

Performance Metric

- Not all errors are equally important.
- It depends on what you NEED. (Need-driven)

- Classification:
  - TP, FP, TN, FN, Accuracy, Recall, Precision, Sensitivity, Specificity, AUROC, F1, …
- Prediction (Regression)
  - MSE, MAE, MSLE, R2, …

Confusion Matrix

Real class\Predicted class	C_1	C_2
C_1	True positive (TP)	False negative (FN)
C_2	False positive (FP)	True negative (TN)

Fact\Answer	Yes	no	total
yes	6 (TP)	3 (FN)	9 (P)
no	2 (FP)	3 (TN)	5 (N)
total	8	6	14

- Accuracy: percentage of test set tuples that are correctly classified:
  - $Accuracy = \frac{TP+TN}{P+N} = \frac{6+3}{14}$
  - $Recall = \frac{TP}{P} = \frac{6}{9}$ (percentage of positives correctly answered)
  - $Precision = \frac{TP}{TP+FP} = \frac{6}{6+2}$ (chance that a positive answer is correct)

Confusion Matrix – F1

Real class\Predicted class	C_1	C_2
C_1	True positive (TP)	False negative (FN)
C_2	False positive (FP)	True negative (TN)

Fact\Answer	Yes	no	total
yes	6 (TP)	3 (FN)	9 (P)
no	2 (FP)	3 (TN)	5 (N)
total	8	6	14

- F measure: the harmonic mean of precision and recall:
  - $F = \frac{2 * Precision * Recall}{Precision + Recall}$

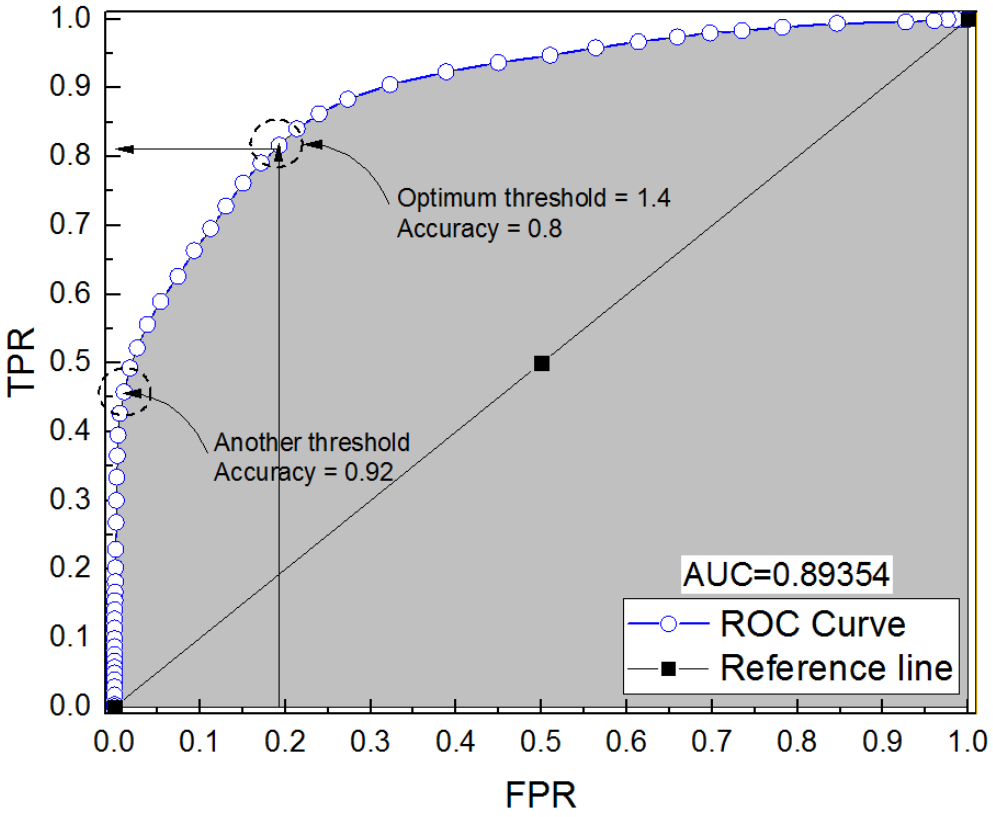
Multi-class Measures

- Micro-average
  - sum up the individual true positives, false positives, and false negatives for different classes and then apply them to get the statistics.

- Macro-average
  - take the average of the chosen metric (e.g. precision, recall etc.) of the system on different classes

AUROC

- Area under receiver operating characteristic curve



https://stats.stackexchange.com/questions/225210/accuracy-vs-area-under-the-roc-curve

Typical Training Loss

- Classification loss
  - Logistic/Binary Cross Entropy (Binary)
  - Hinge Loss (Binary)
  - KL divergence (Binary)
  - Cross Entropy (Categorical, one-hot encoded)
  - Sparse Categorical Cross entropy (Categorical)

Binary Cross Entropy (Binary)

$$-(y \log(p) + (1 - y) \log(1 - p))$$

```
# Compute cross entropy from probabilities.
bce = target * math_ops.log(output + epsilon())
bce += (1 - target) * math_ops.log(1 - output + epsilon())
return -bce
```

y_true = [[0., 1.], [0., 0.]]
y_pred = [[0.6, 0.4], [0.4, 0.6]]

bce = tf.keras.losses.BinaryCrossentropy()
bce(y_true, y_pred).numpy()

https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#cross-entropy

Hinge Loss

$$\text{loss}(\boldsymbol{w}) = \sum_{i}^{n} \max(0, 1 - y_i \boldsymbol{w}^T \boldsymbol{x})$$

```
1381    y_pred = ops.convert_to_tensor_v2(y_pred)
1382    y_true = math_ops.cast(y_true, y_pred.dtype)
1383    y_true = _maybe_convert_labels(y_true)
1384    return K.mean(math_ops.maximum(1. - y_true * y_pred, 0.), axis=-1)
1385
```

y_true = [[0., 1.], [0., 0.]]
y_pred = [[0.6, 0.4], [0.4, 0.6]]

h = tf.keras.losses.Hinge()
h(y_true, y_pred).numpy()

KL divergence

- Kullback-Leibler divergence loss
- Measure how one probability distribution differs from a second
- Asymmetric

loss = y * log(y / p)

```
y_pred = ops.convert_to_tensor_v2(y_pred)
y_true = math_ops.cast(y_true, y_pred.dtype)
y_true = K.clip(y_true, K.epsilon(), 1)
y_pred = K.clip(y_pred, K.epsilon(), 1)
return math_ops.reduce_sum(y_true * math_ops.log(y_true / y_pred), axis=-1)
```

Categorical Target

- Softmax Function:
  - logits: output from the dense layer before prediction

pred = tf.exp(logits) / tf.reduce_sum(tf.exp(logits))

logits = [1, 2, 3, 4, 5, 6]

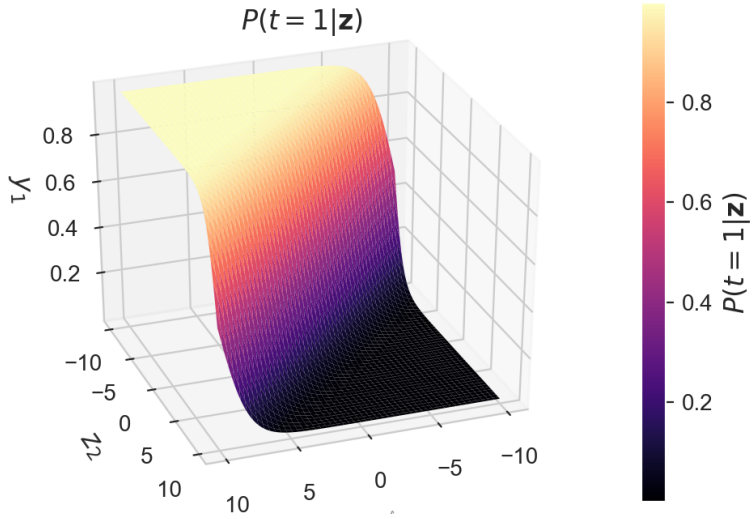
pred = [
0.0042697826282382,
0.011606461368501186,
0.0315496325492858 9,
0.0857607871294021 6,
0.2331220060586929 3,
0.6336913108825684]

https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#cross-entropy

Cross Entropy (Categorical)

pred = [	y = [
0.0042697782628238 2,	0,
0.011606461368501186,	0,
0.031549632549285 89,	0,
0.085760787129402 16,	0,
0.233122006058692 93,	1,
0.633691310882568 4]	0]

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i)$$



https://peterroelants.github.io/posts/cross-entropy-softmax/

Cross Entropy (TF)

View aliases

```
tf.keras.losses.categorical_crossentropy(
    y_true, y_pred, from_logits=False, label_smoothing=0
)
```

Standalone usage:

```
>>> y_true = [[0, 1, 0], [0, 0, 1]]
>>> y_pred = [[0.05, 0.95, 0], [0.1, 0.8, 0.1]]
>>> loss = tf.keras.losses.categorical_crossentropy(y_true, y_pred)
>>> assert loss.shape == (2,)
>>> loss.numpy()
array([0.0513, 2.303], dtype=float32)
```

https://peterroelants.github.io/posts/cross-entropy-softmax/

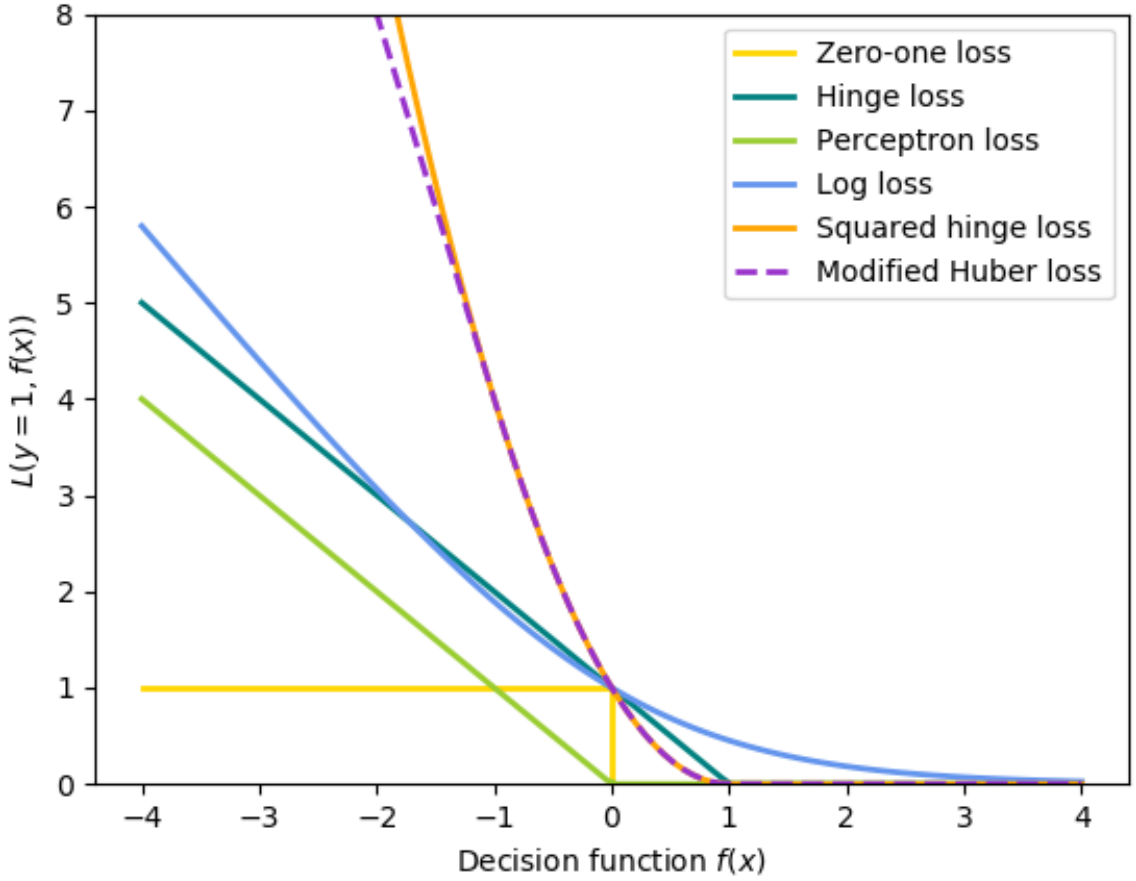
Cross Entropy (TF)

```
tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=False, reduction=losses_utils.ReductionV2.AUTO,
    name='sparse_categorical_crossentropy'
)
```

```
>>> y_true = [1, 2]
>>> y_pred = [[0.05, 0.95, 0], [0.1, 0.8, 0.1]]
>>> # Using 'auto'/'sum_over_batch_size' reduction type.
>>> scce = tf.keras.losses.SparseCategoricalCrossentropy()
>>> scce(y_true, y_pred).numpy()
1.177
```

https://peterroelants.github.io/posts/cross-entropy-softmax/

Summary



https://scikit-learn.org/stable/auto_examples/linear_model/plot_sgd_loss_functions.html