



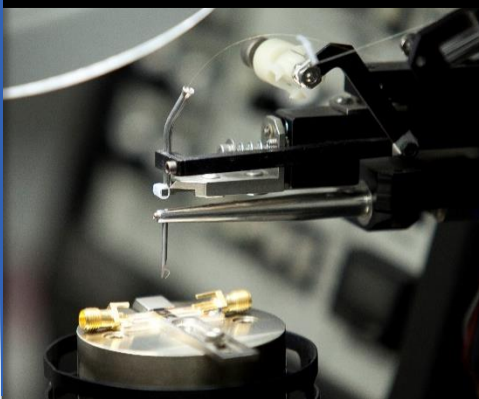
Centro Brasileiro de Pesquisas Físicas



Redes Neurais profundas e aplicações Deep Learning

Clécio Roque De Bom – debom@cbpf.br

clearnightsrthebest.com



EXEMPLO 3

RECONHECIMENTO DE CARACTERES MANUSCRITOS

REDE NEURAL CNN

**CONVOLUTION NEURAL
NETWORK**



The simplest example I know

```
from keras.datasets import mnist
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from keras.models import Sequential

model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(1000, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
```

The simplest example I know

```
batch_size = 128
num_classes = 10
epochs = 10

# input image dimensions
img_x, img_y = 28, 28

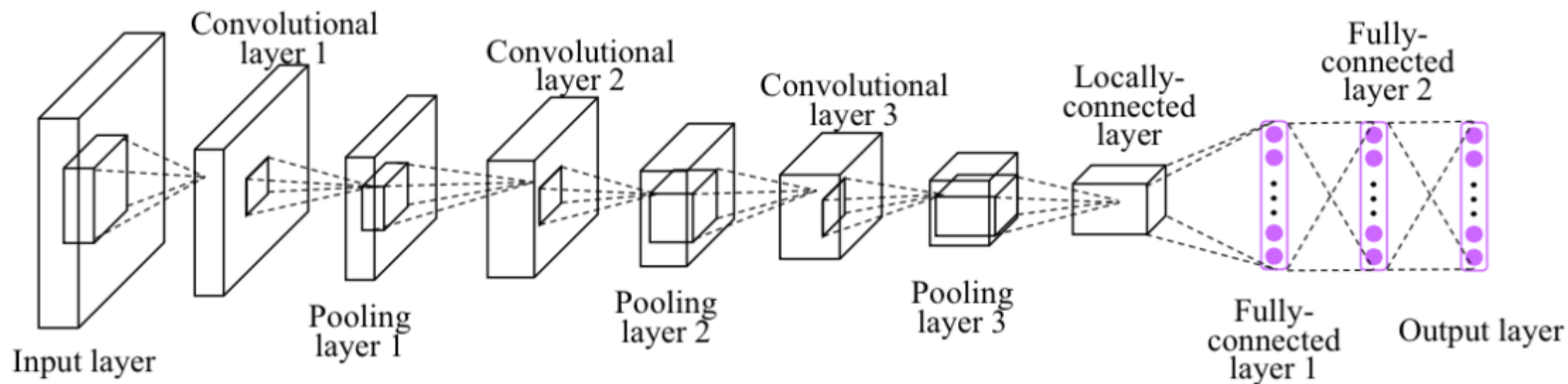
# load the MNIST data set, which already splits into train and test sets
for us
(x_train, y_train), (x_test, y_test) = mnist.load_data()

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test),
          callbacks=[history])
score = model.evaluate(x_test, y_test, verbose=0)
```



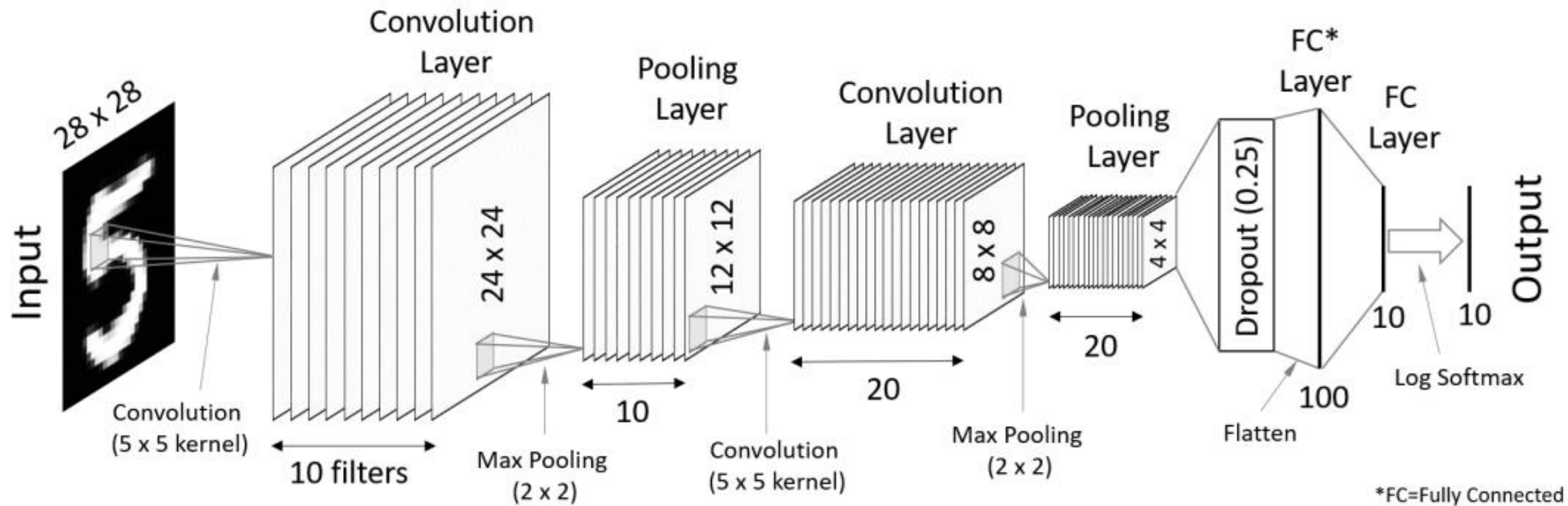
CNN

Overview

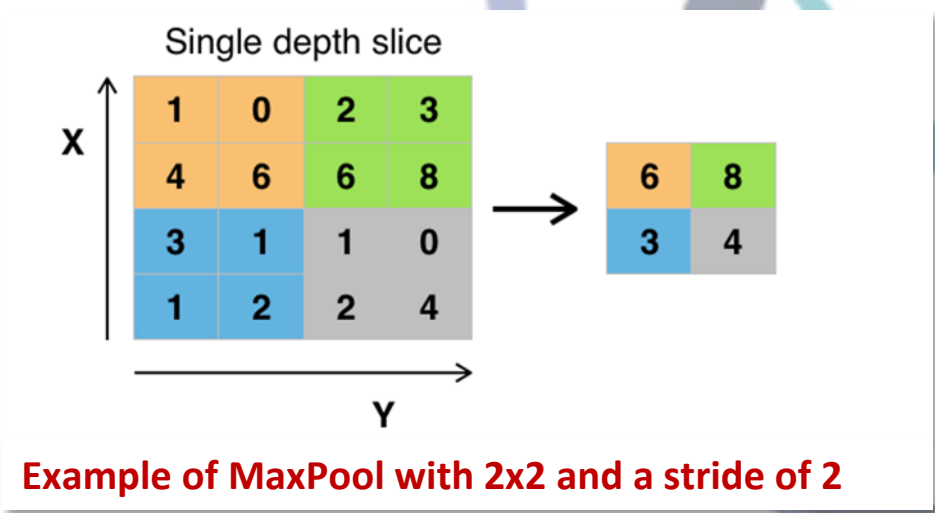
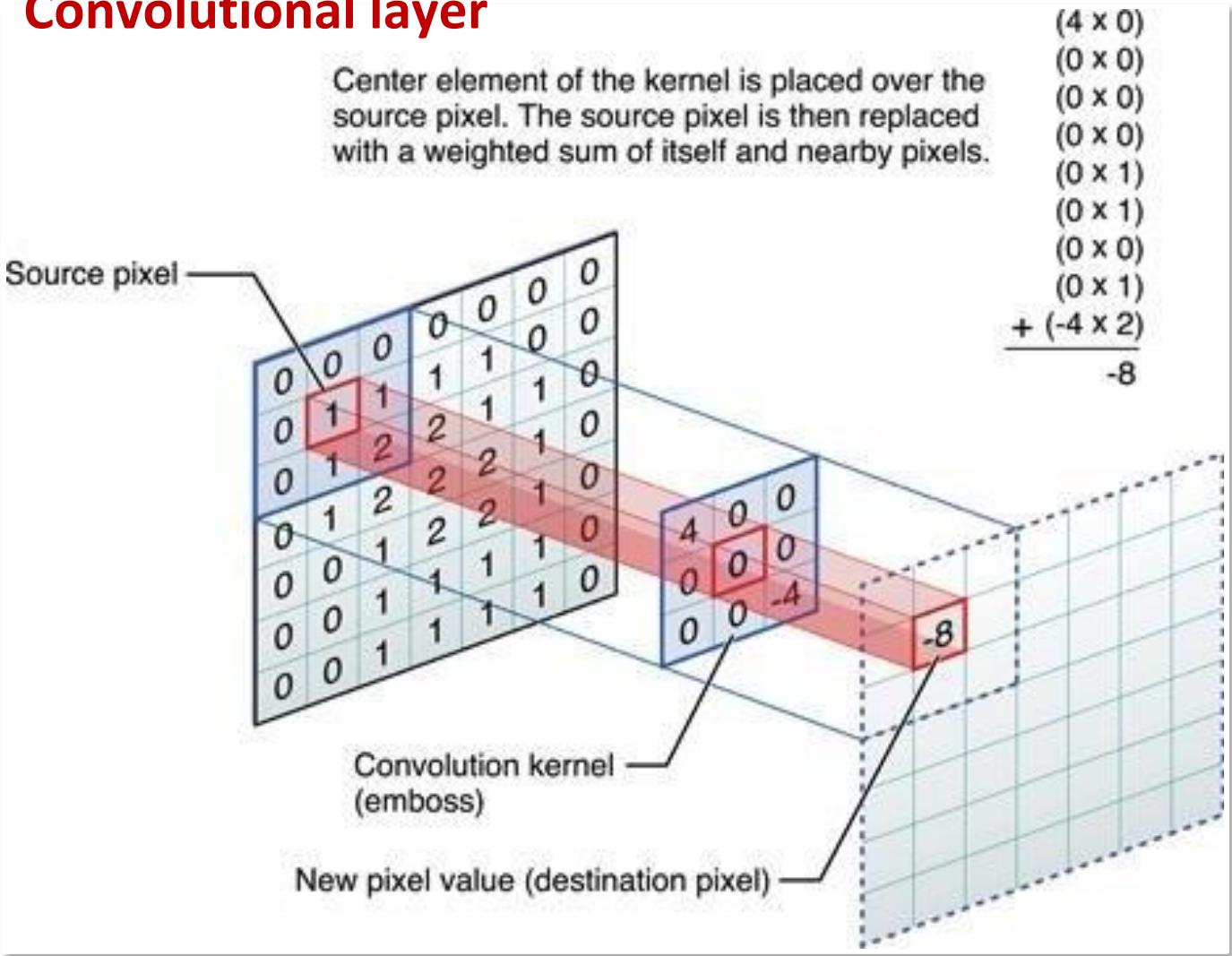


Goal

The Goal of this notebook is to present a simple example of cnn running with keras (Tensorflow backend) in the colab interface. We also present some example of results of a simple application of handwritten digit classification.



Convolutional layer

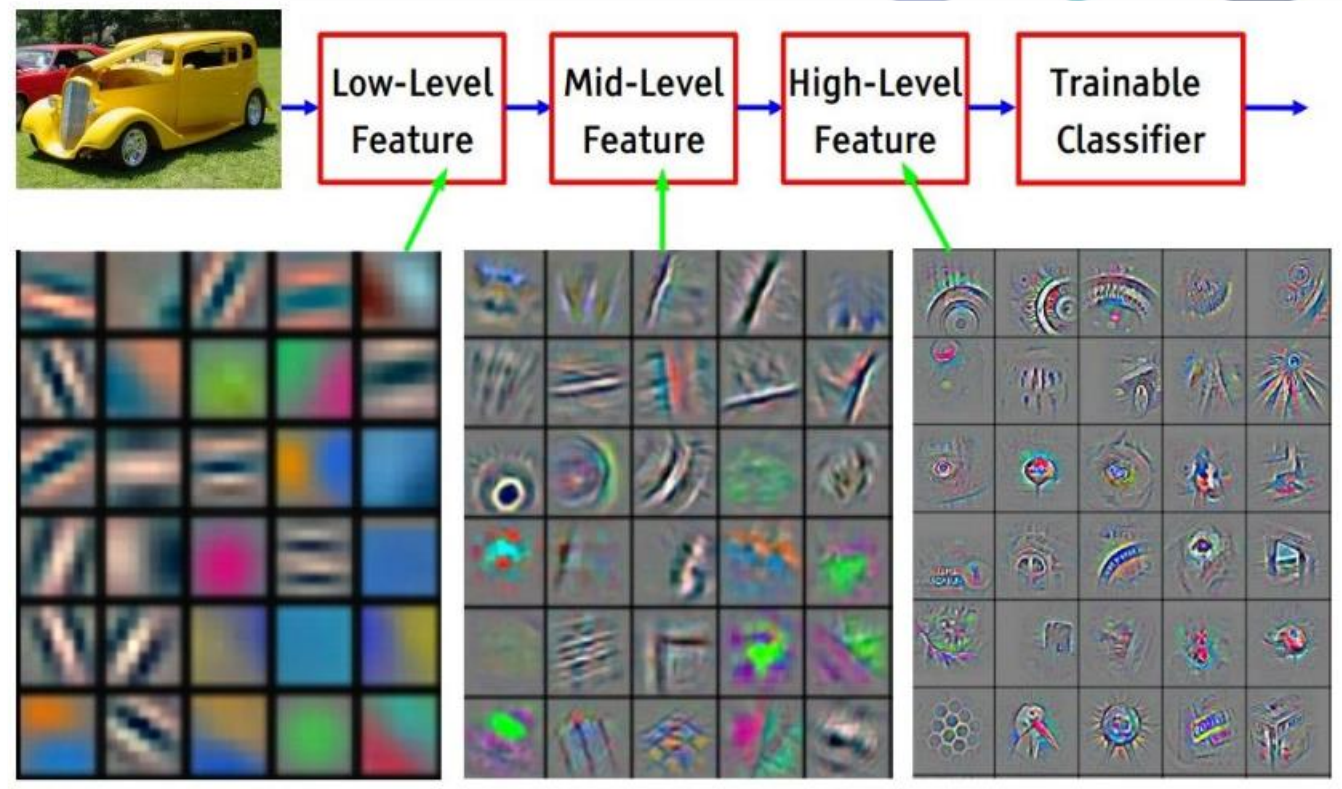


Example of MaxPool with 2x2 and a stride of 2

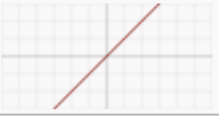

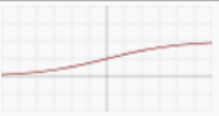
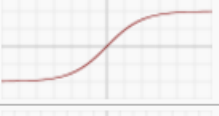
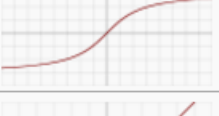


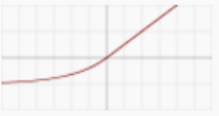

Convolutional Neural Networks

What makes CNNs so special?

- Based on mammal visual cortex
 - Extract **surrounding-dependent** high-order features.
 - Specially useful for:
 - Images
 - Time-dependent parameters
- Speech recognition*
Signal analysis



Activation Functions

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parametric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Do not take advantage of Neural Nets for non linearities estimation. Useful in regression problems since is unbounded.

Hard to train, derivative vanishes.

Easily differentiable. In the last layers can be associated with probability.

Similar results as in sigmoid activations in intermediate layers. However, is numerically faster.

Tentative to avoid the vanishing of the ReLU derivative.

Adapted from

<https://towardsdatascience.com/>

activation-functions-neural-networks-1cbd9f8d91d6

Why Sigmoid for classification?

Consider two classes, $y \in \{0, 1\}$.

The conditional probability of class $P(y|z(x))$ where $z = \omega^T h(x) + b$ the output of a set of neurons with x inputs.

Why Sigmoid for classification?

the unnormalized log probability can be written as

$$\log \hat{P}(y = 1 | z) = z \quad (\text{neurons "on"})$$

$$\log \hat{P}(y = 0 | z) = 0 \quad (\text{neurons "off"})$$

$$\hat{P}(y = 1 | z) = \exp(z)$$

$$\hat{P}(y = 0 | z) = \exp(0) = 1$$

Why Sigmoid for classification?

The Normalized version:

$$P(y = 1|z) = \frac{\exp(z)}{1+\exp(z)}$$

$$P(y = 0|z) = \frac{1}{1+\exp(z)} \quad .$$

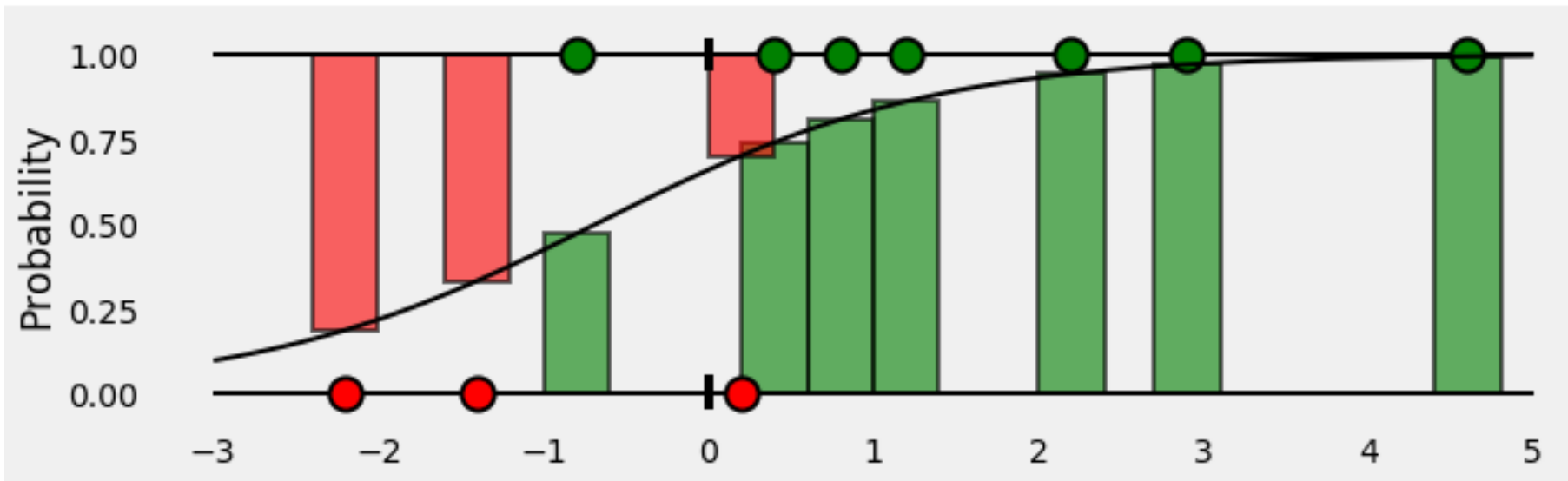
This is

$$P(y = 1|z) = \frac{\exp(z)}{1+\exp(z)} = \frac{1}{\frac{\exp(z)+1}{\exp(z)}} = \frac{1}{1+\exp(-z)} = \sigma(z)$$

$$P(y = 0|z) = \sigma(-z) \quad .$$

Some Loss intuition...

Consider the binary classification of red and greens. The True class probability of a set of z points is:

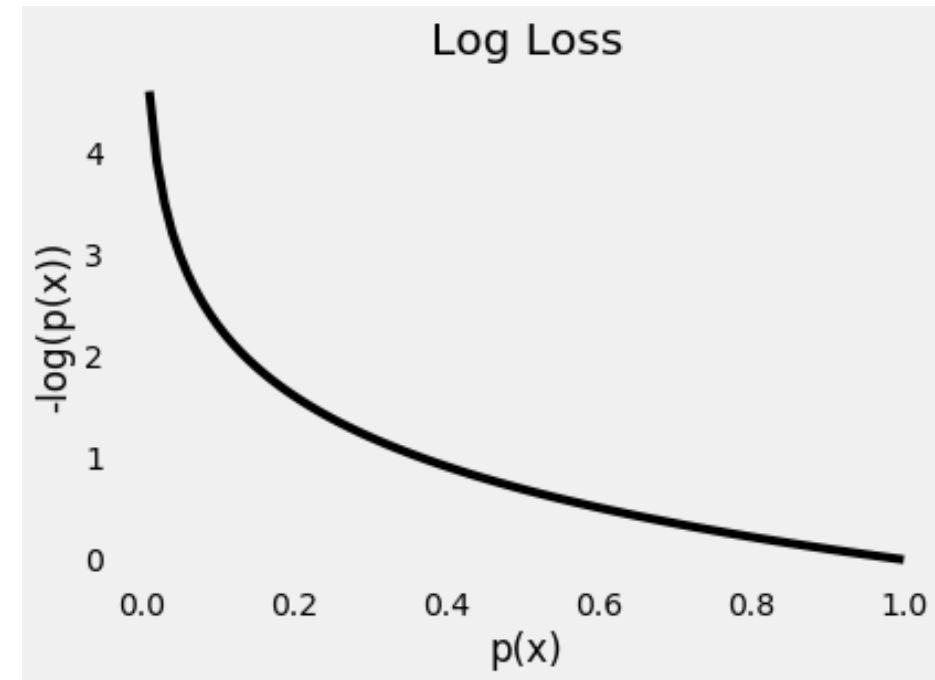
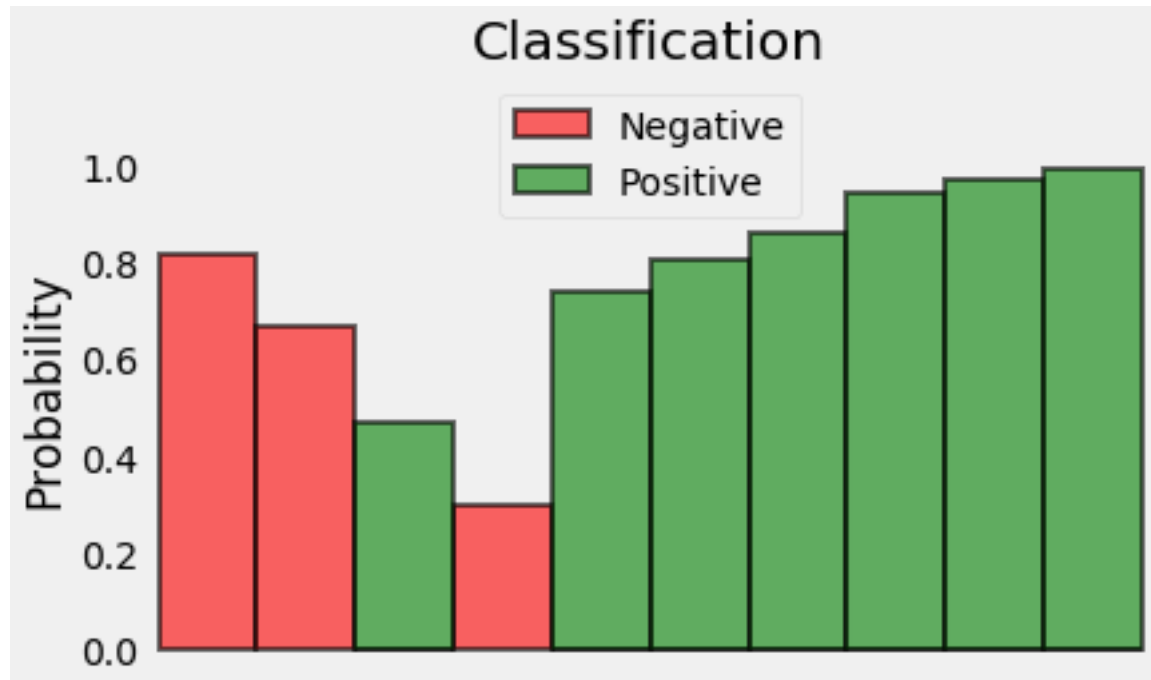


$$P(y = 0|z) = \sigma(-z)$$

$$P(y = 1|z) = \sigma(z)$$

Example adapted from : <https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a>

Some Loss intuition...

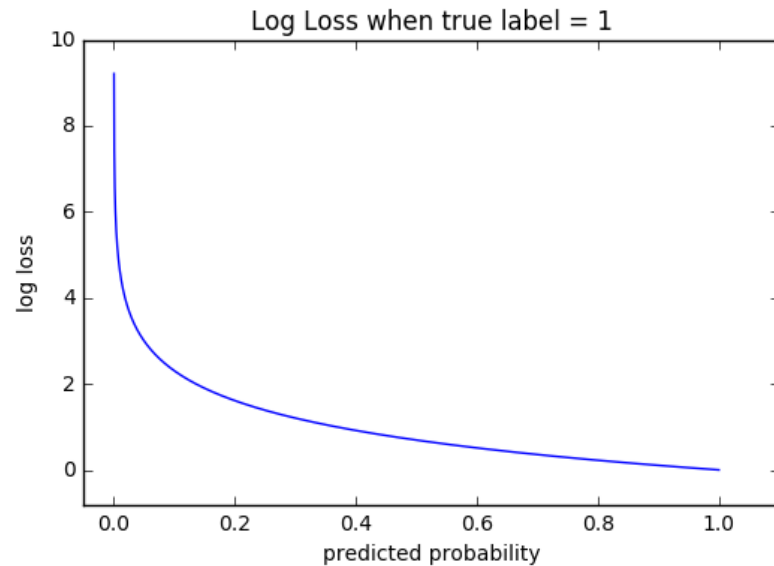


If the predicted probability of the true class gets closer to zero, the $-\log(p(x))$ increases exponentially.

So... Cross entropy Loss

Consider two classes: 1 and 0s. The predicted probability is given by

$$q_{y=1} = \hat{y} \equiv g(\mathbf{w} \cdot \mathbf{x}) = 1/(1 + e^{-\mathbf{w} \cdot \mathbf{x}}),$$



$$H(p, q) = - \sum_i p_i \log q_i = - y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

$$J(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N H(p_n, q_n) = - \frac{1}{N} \sum_{n=1}^N \left[y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n) \right],$$

How to Choose?

The Mean Squared Error loss is the default loss to use for regression problems.

It represents loss function under the inference framework of maximum likelihood.

This assumes the distribution of the target variable is Gaussian.

Change it Carefully.

$$P(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

$$\ln(P(x; \mu, \sigma)) = \ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{(x - \mu)^2}{2\sigma^2}$$

How to Choose?

The Mean Squared Error loss is the default loss to use for regression problems.

It represents loss function under the inference framework of maximum likelihood.

This assumes the distribution of the target variable is Gaussian.

Change it Carefully.

$$P(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

$$\ln(P(x; \mu, \sigma)) = \ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{(x - \mu)^2}{2\sigma^2}$$

Root Mean Squared Log Error

Regression problems in which the target value has a spread of values

When predicting a large value, you may not want to punish a model as heavily as mean squared error, that is your values are small.

Root Mean Squared Error (RMSE)

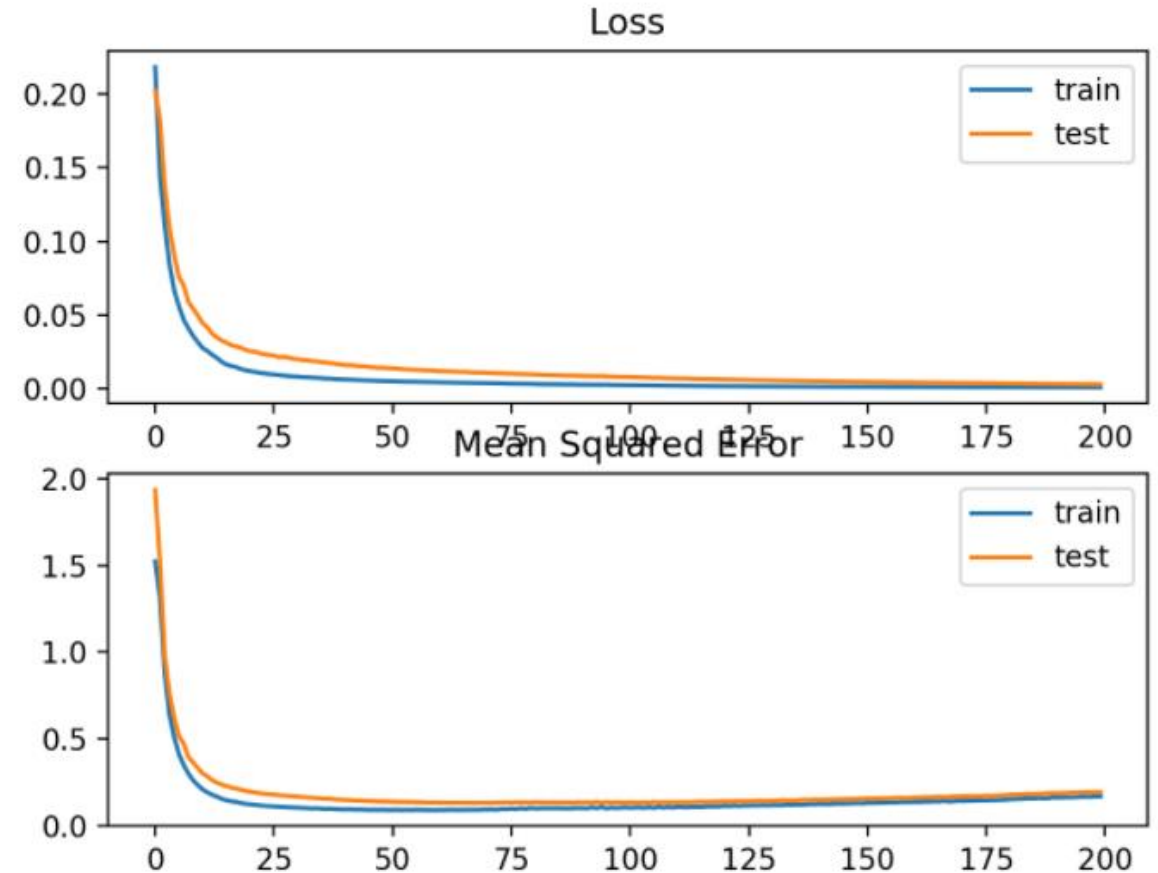
$$\sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Root Mean Squared Log Error (RMSLE)

$$\sqrt{\frac{1}{n} \sum_{i=1}^n (\log(p_i + 1) - \log(a_i + 1))^2}$$

prediction

actual

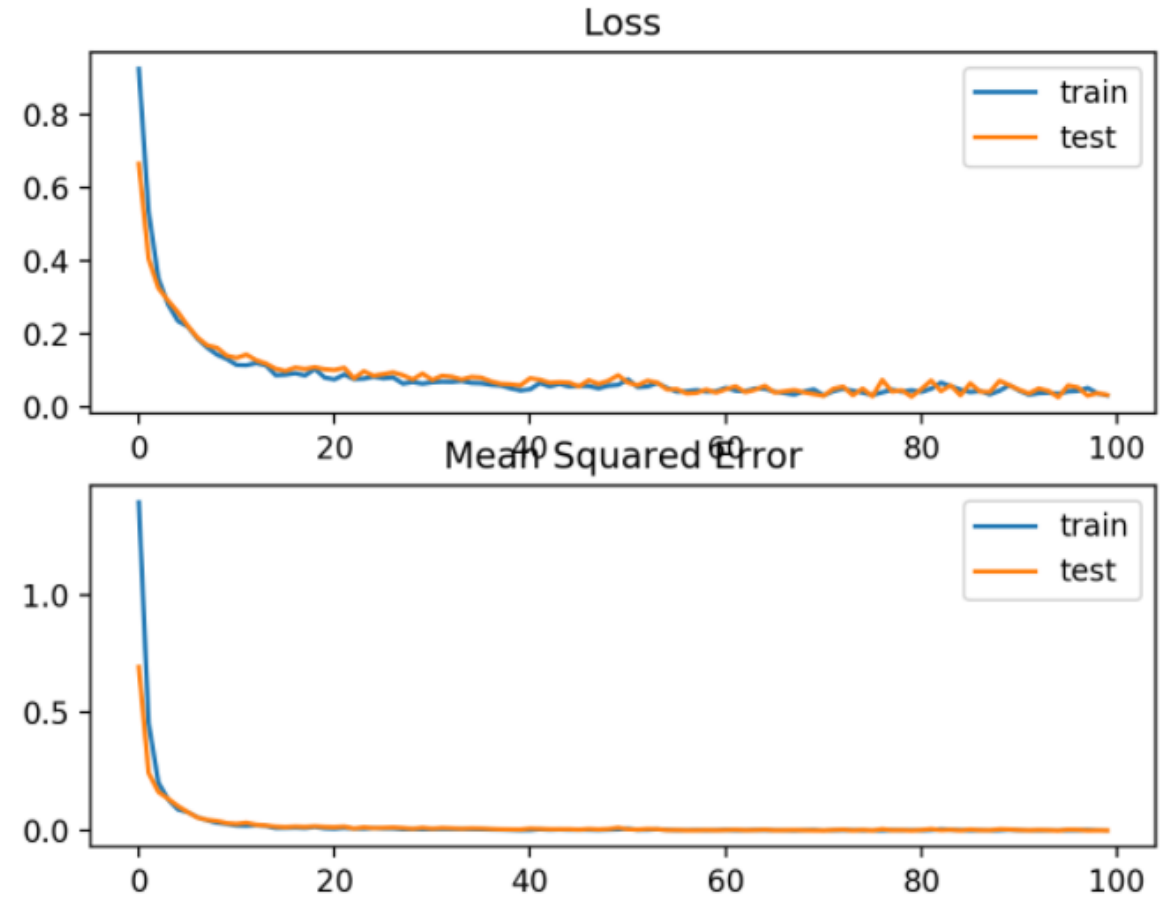


Root Absolute Squared Error

The Mean Absolute Error loss is an appropriate loss function in this case as it is more robust to outliers.

In case outliers matters!

$$\text{MAE} = \sum_{i=1}^n (Y_i - \hat{Y}_i)^2.$$



What Metrics is for ?

Common Classification Metrics

Binary Accuracy: `binary_accuracy`, `acc`

Categorical Accuracy: `categorical_accuracy`,

Common Regression Metrics

Mean Squared Error: `mean_squared_error`, MSE or mse

Mean Absolute Error: `mean_absolute_error`, MAE, mae





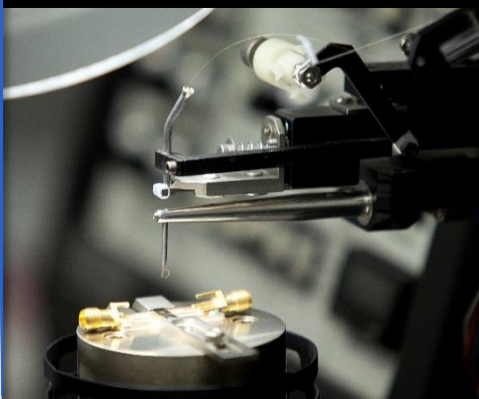
Centro Brasileiro de Pesquisas Físicas



Redes Neurais profundas e aplicações Deep Learning

Clécio Roque De Bom – debom@cbpf.br

clearnightsrthebest.com



Model Evaluation

True Positive (TP):

- Reality: A wolf threatened.
- Shepherd said: "Wolf."
- Outcome: Shepherd is a hero.

False Positive (FP):

- Reality: No wolf threatened.
- Shepherd said: "Wolf."
- Outcome: Villagers are angry at shepherd for waking them up.

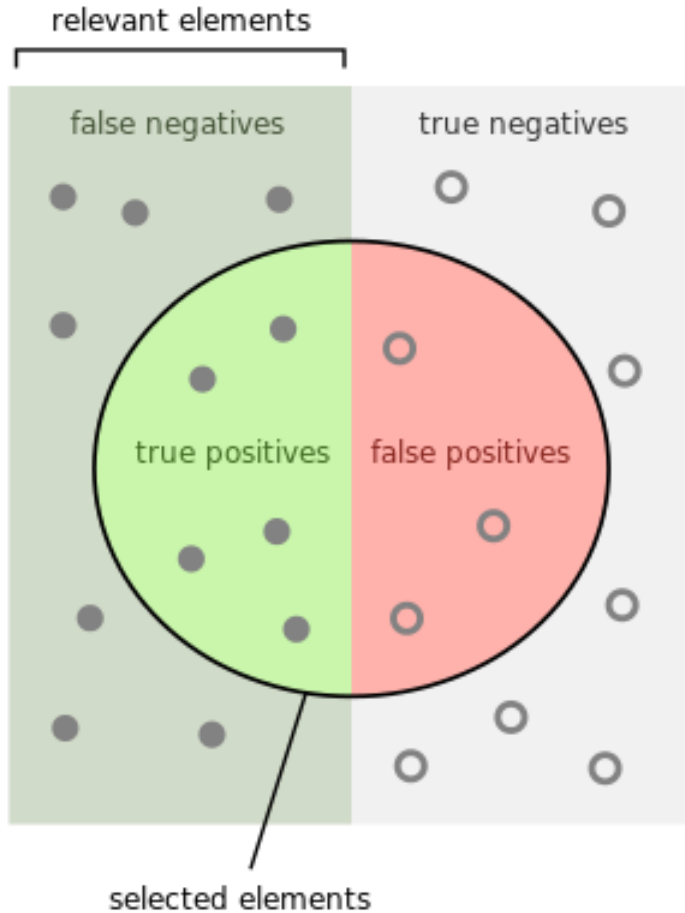
False Negative (FN):

- Reality: A wolf threatened.
- Shepherd said: "No wolf."
- Outcome: The wolf ate all the sheep.

True Negative (TN):

- Reality: No wolf threatened.
- Shepherd said: "No wolf."
- Outcome: Everyone is fine.

Results Metrics



How many relevant items are selected?

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Sensitivity, TPR, Recall, Completeness

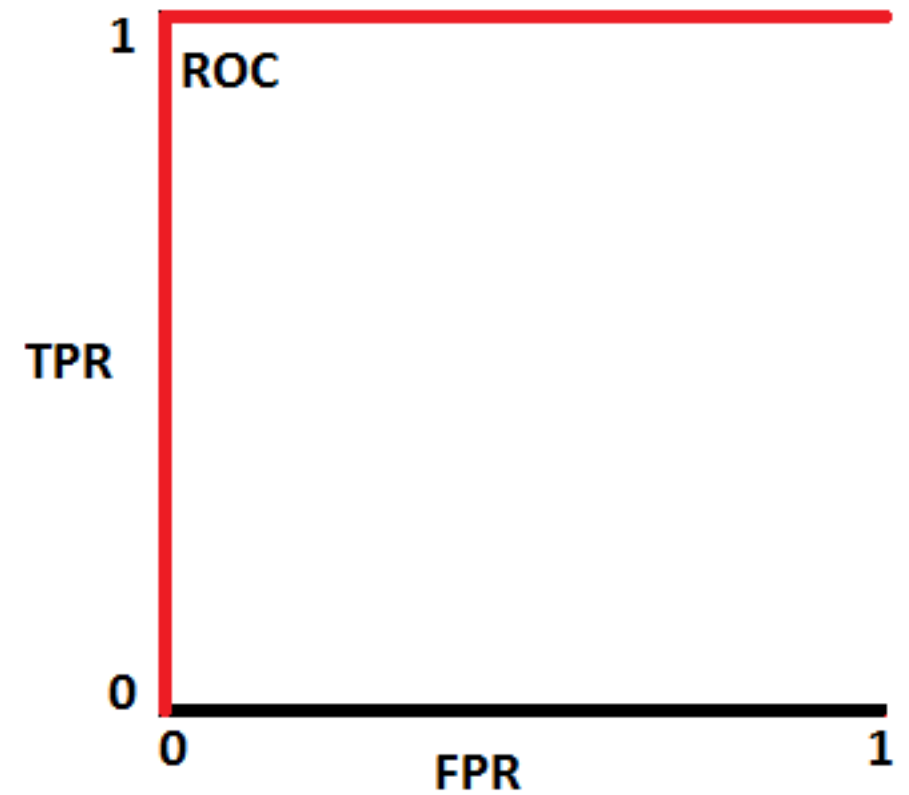
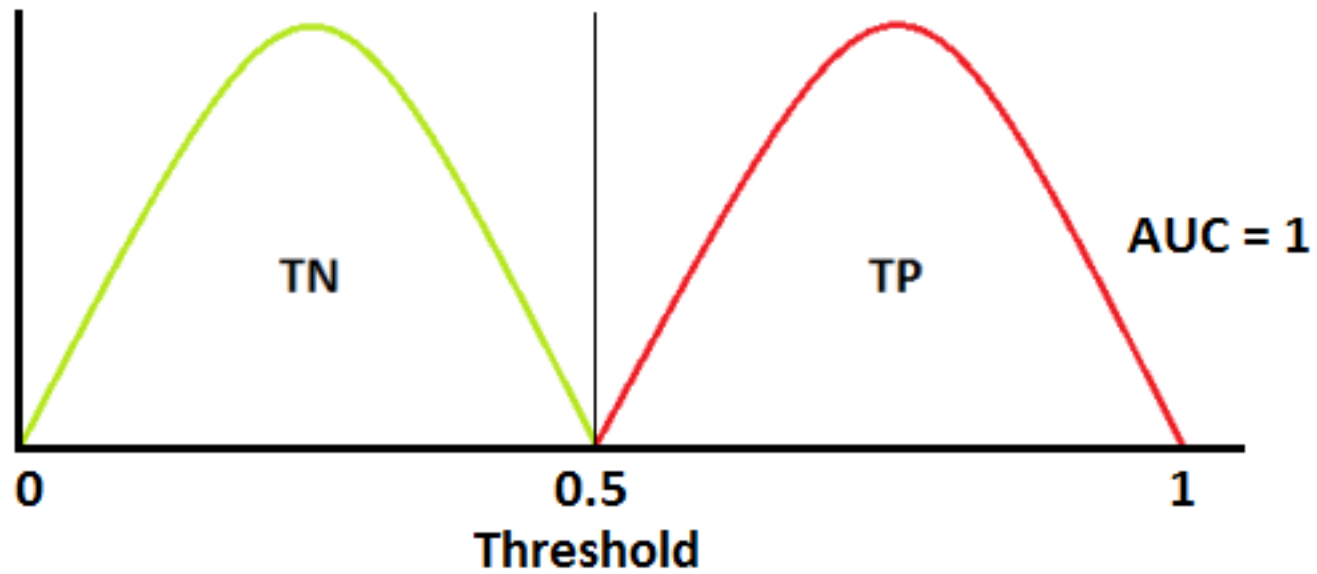
$$\text{TPR} = \frac{\text{TP}}{\text{P}} = \frac{\text{TP}}{\text{TP} + \text{FN}} = 1 - \text{FNR}$$

$$\text{False Alarm rate} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

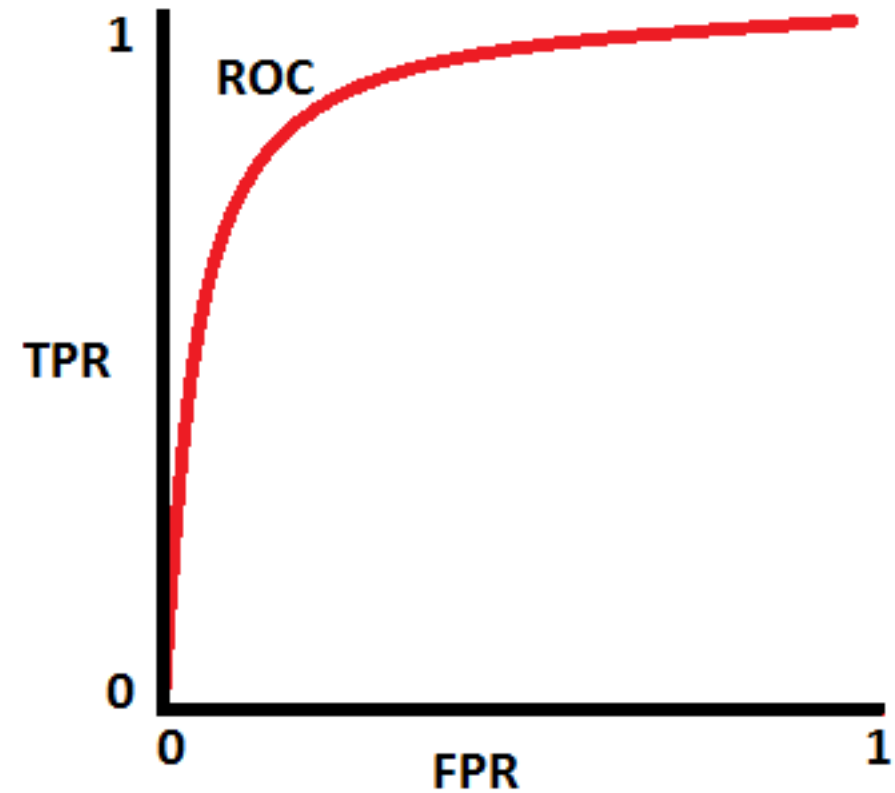
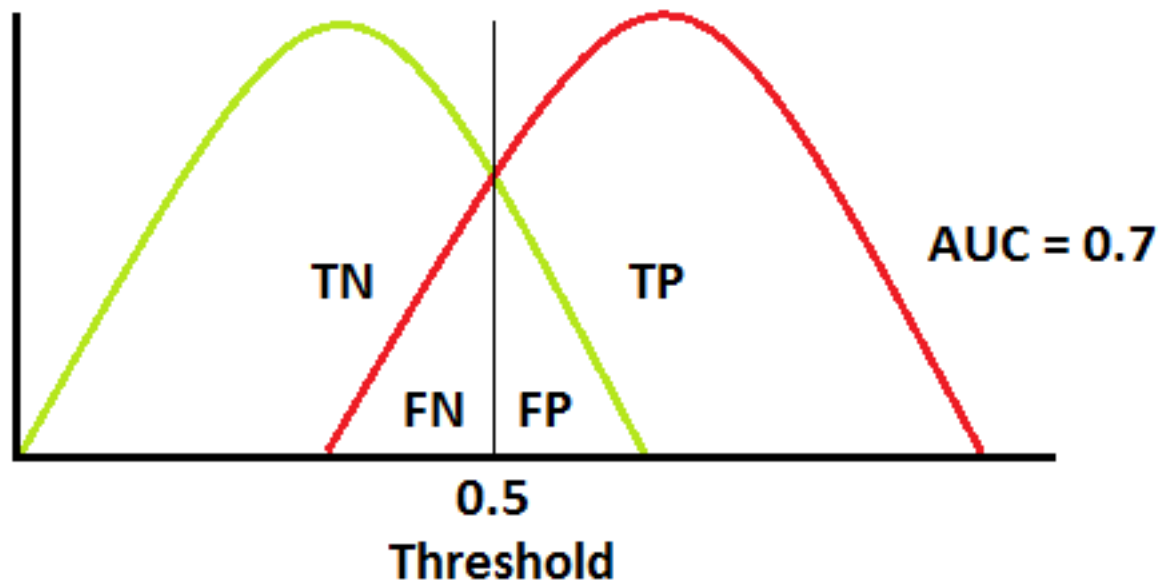
FPR, False Alarm Rate

$$\text{FPR} = \frac{\text{FP}}{\text{N}} = \frac{\text{FP}}{\text{FP} + \text{TN}} = 1 - \text{TNR}$$

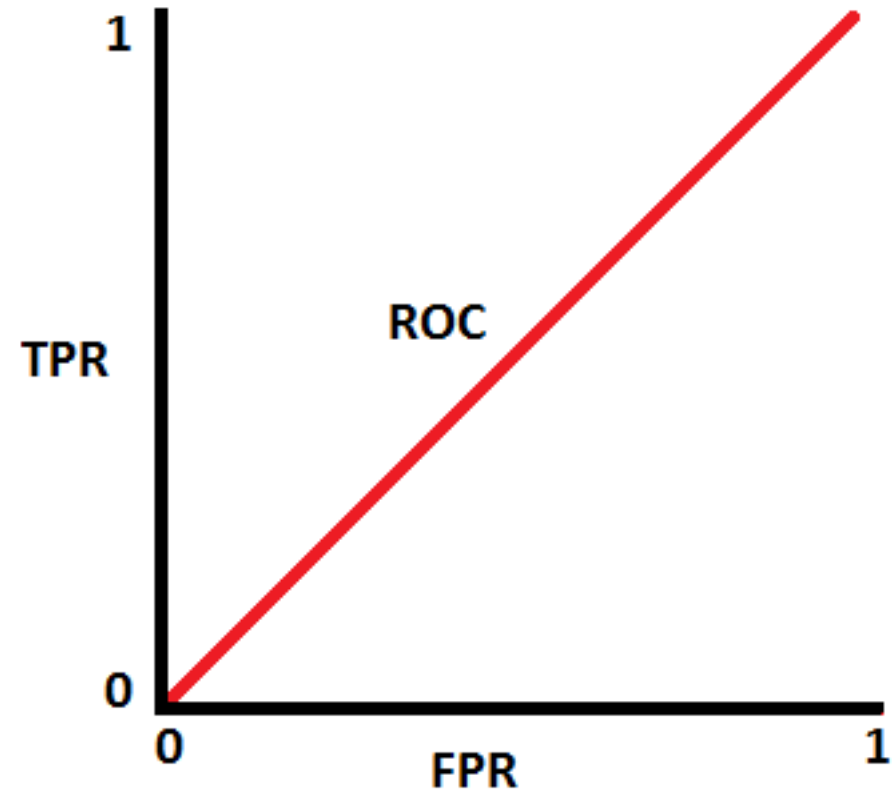
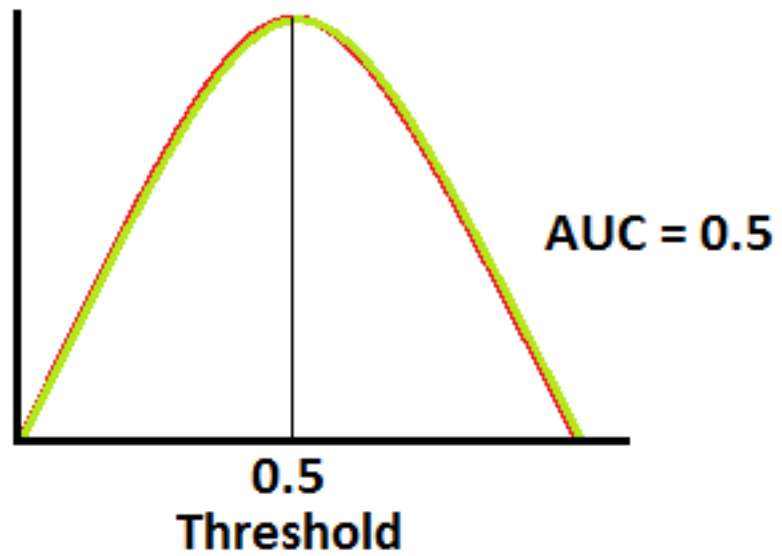
How good are the results?



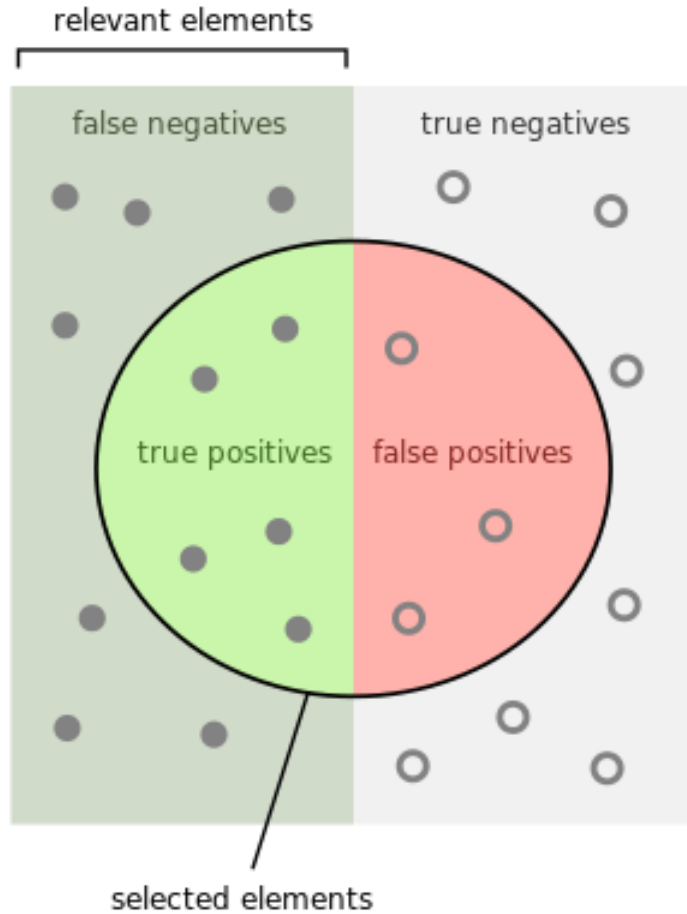
How good are the results?



How good are the results?



Results Metrics



Precision, Purity

How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Sensitivity, TPR, Recall, Completeness


How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Results Metrics


Precision, Purity

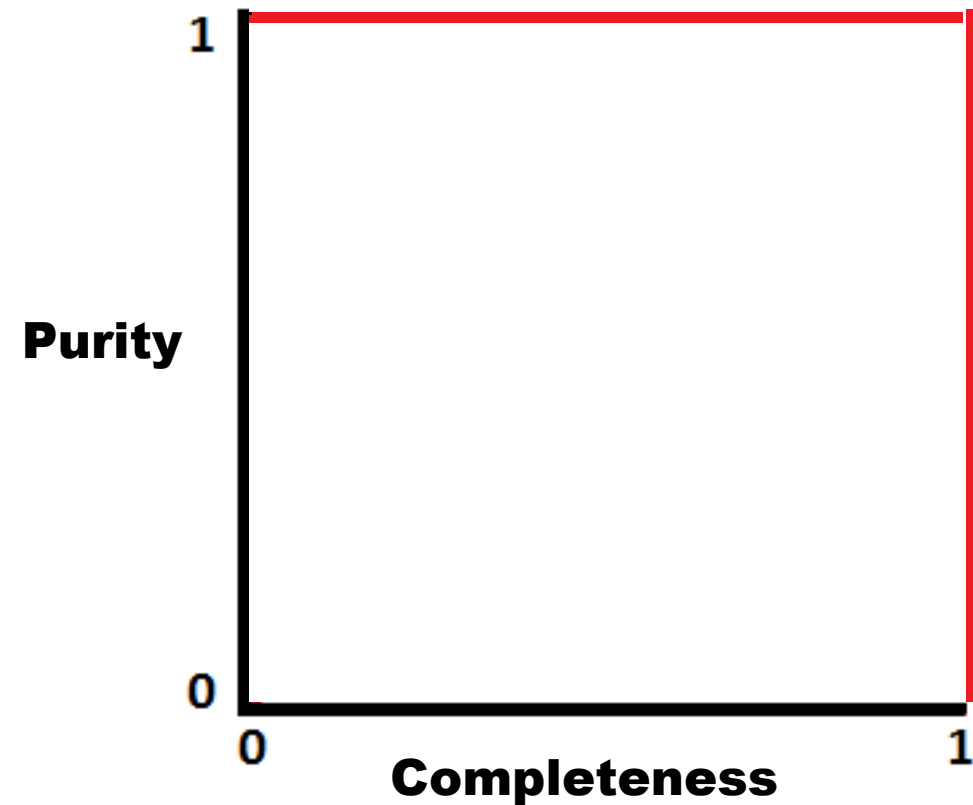
How many selected items are relevant?

$$\text{Precision} = \frac{\text{Green Semi-Circle}}{\text{Green and Red Semi-Circle}}$$


Sensitivity, TPR, Recall, Completeness

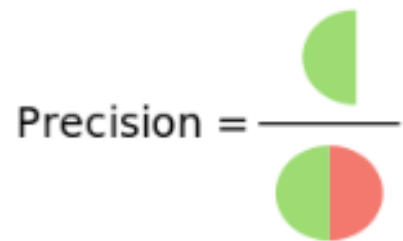
How many relevant items are selected?

$$\text{Recall} = \frac{\text{Green Semi-Circle}}{\text{Green Semi-Circle inside a Green Rectangle}}$$


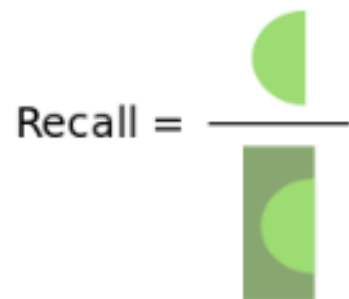


The F- Score

How many selected items are relevant?

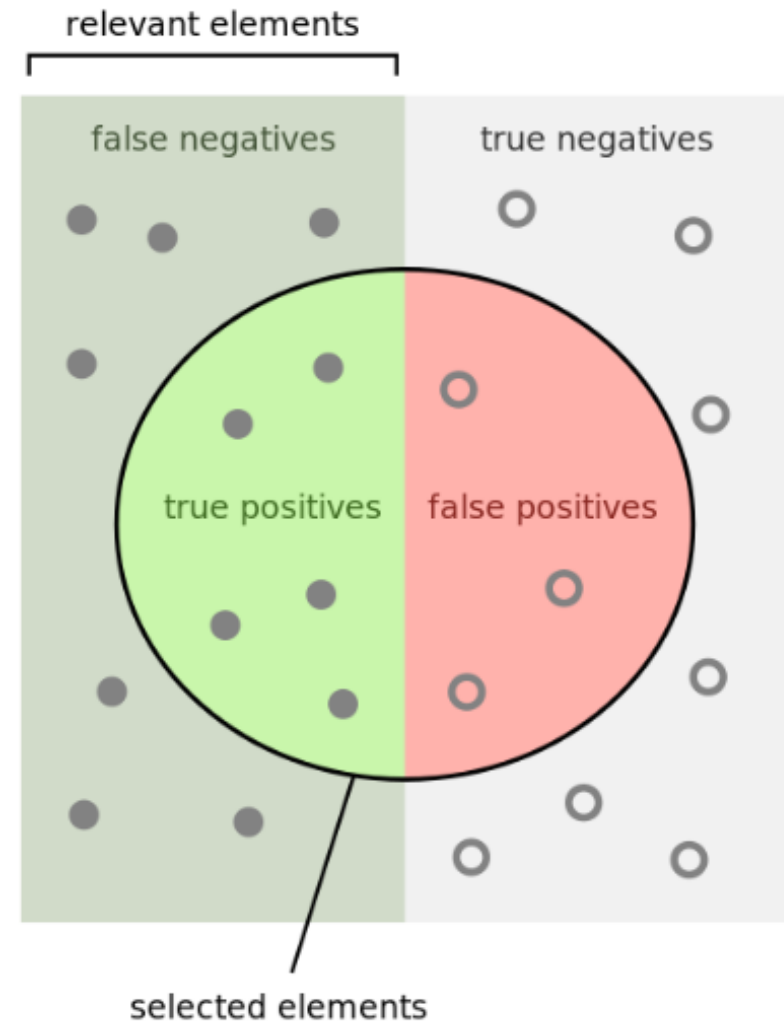


How many relevant items are selected?



$$F_{\beta} = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

$$F_{\beta=0} = \text{precision}, F_{\beta=\infty} = \text{recall}$$



K-Fold Cross Validation

Shuffle the dataset randomly.

Split the dataset into k sets

For each k set:

- Take the group as a hold out or test data set

- Take the remaining groups as a training data set

- Fit a model on the training set and evaluate it on the test set

- Save the evaluation scores in the test set

Summarize the results by defining average (or median) and std on each threshold.

Iteration 1



Iteration 2



Iteration 3



Iteration 4



Iteration 5



K-Fold Cross Validation

Shuffle the dataset randomly.

Split the dataset into k sets

For each k set:

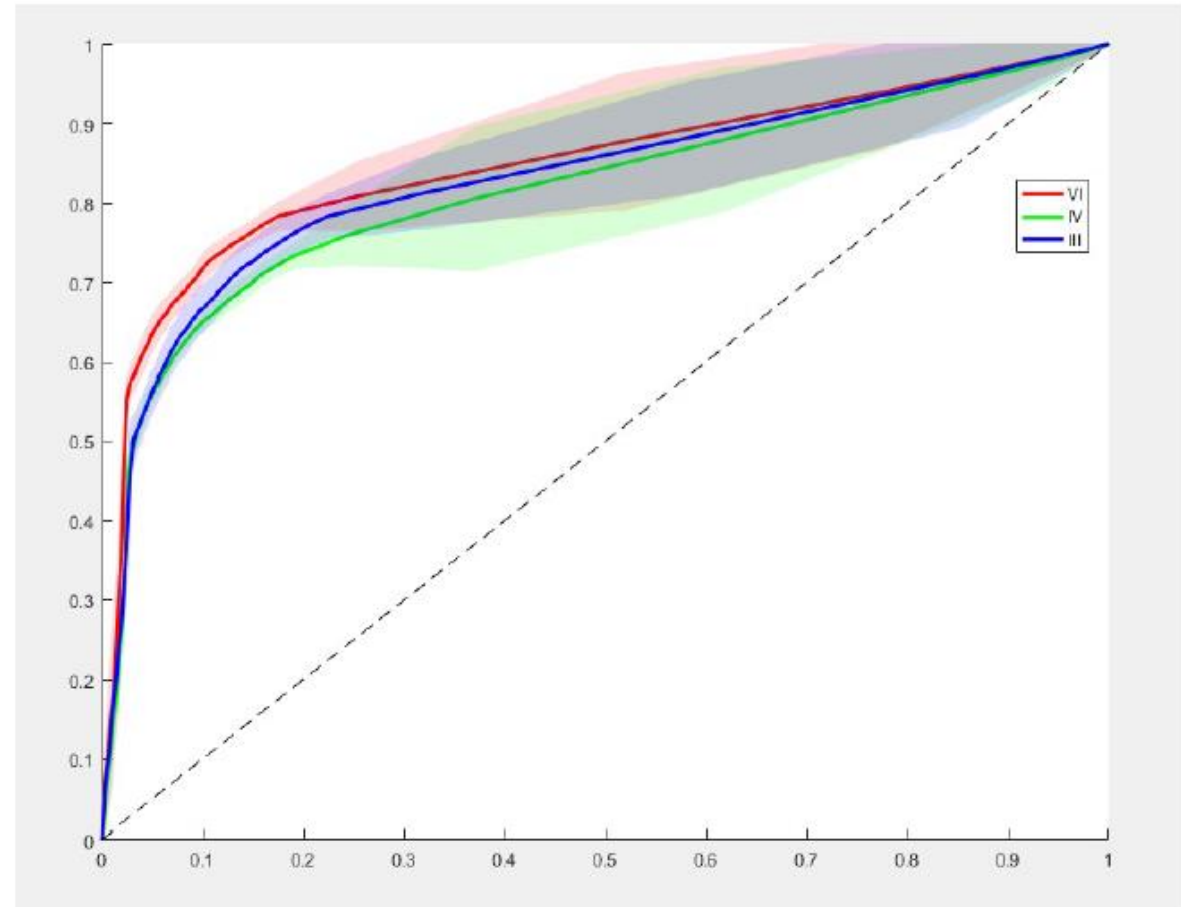
- Take the group as a hold out or test data set

- Take the remaining groups as a training data set

- Fit a model on the training set and evaluate it on
the test set

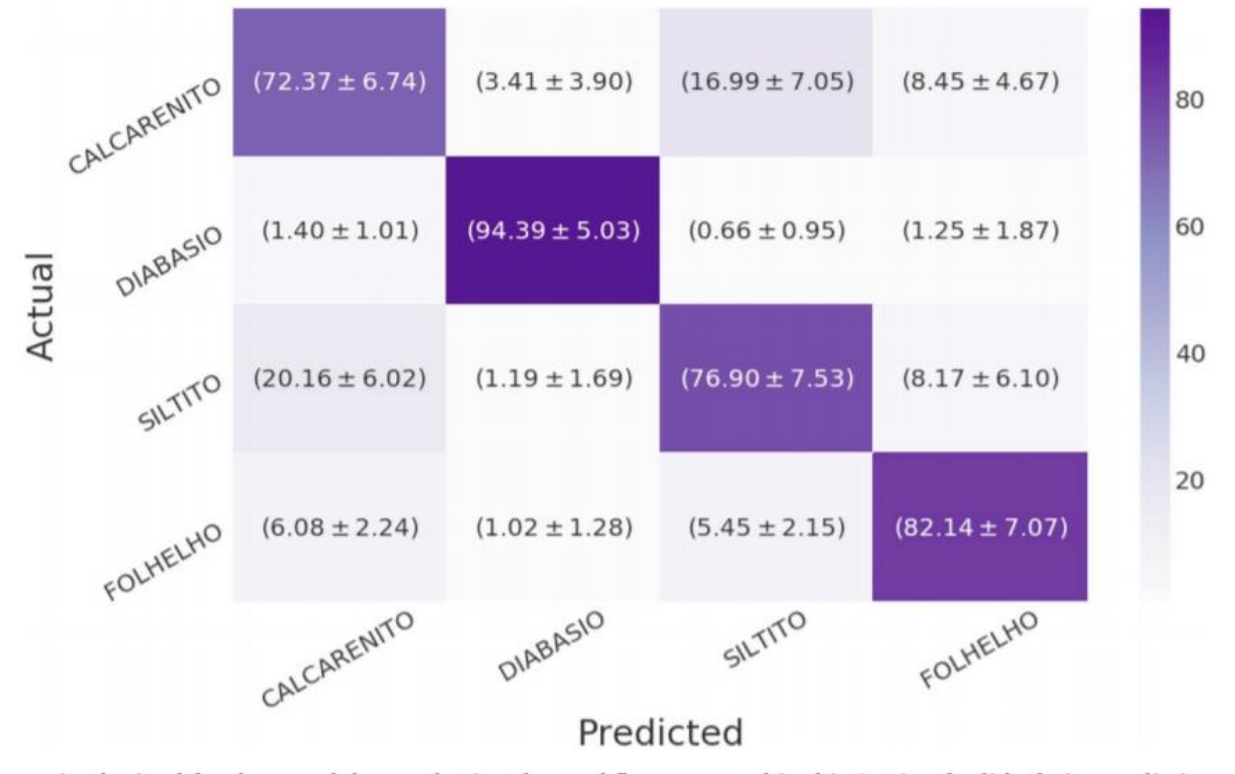
- Save the evaluation scores in the test set

Summarize the results by defining average (or
median) and std on each threshold.



Confusion Matrix

		Actual class		
		Cat	Dog	Rabbit
Predicted class	Cat	5	2	0
	Dog	3	3	2
	Rabbit	0	1	11





Centro Brasileiro de Pesquisas Físicas



Redes Neurais profundas e aplicações Deep Learning

Clécio Roque De Bom – debom@cbpf.br

clearnightsrthebest.com

