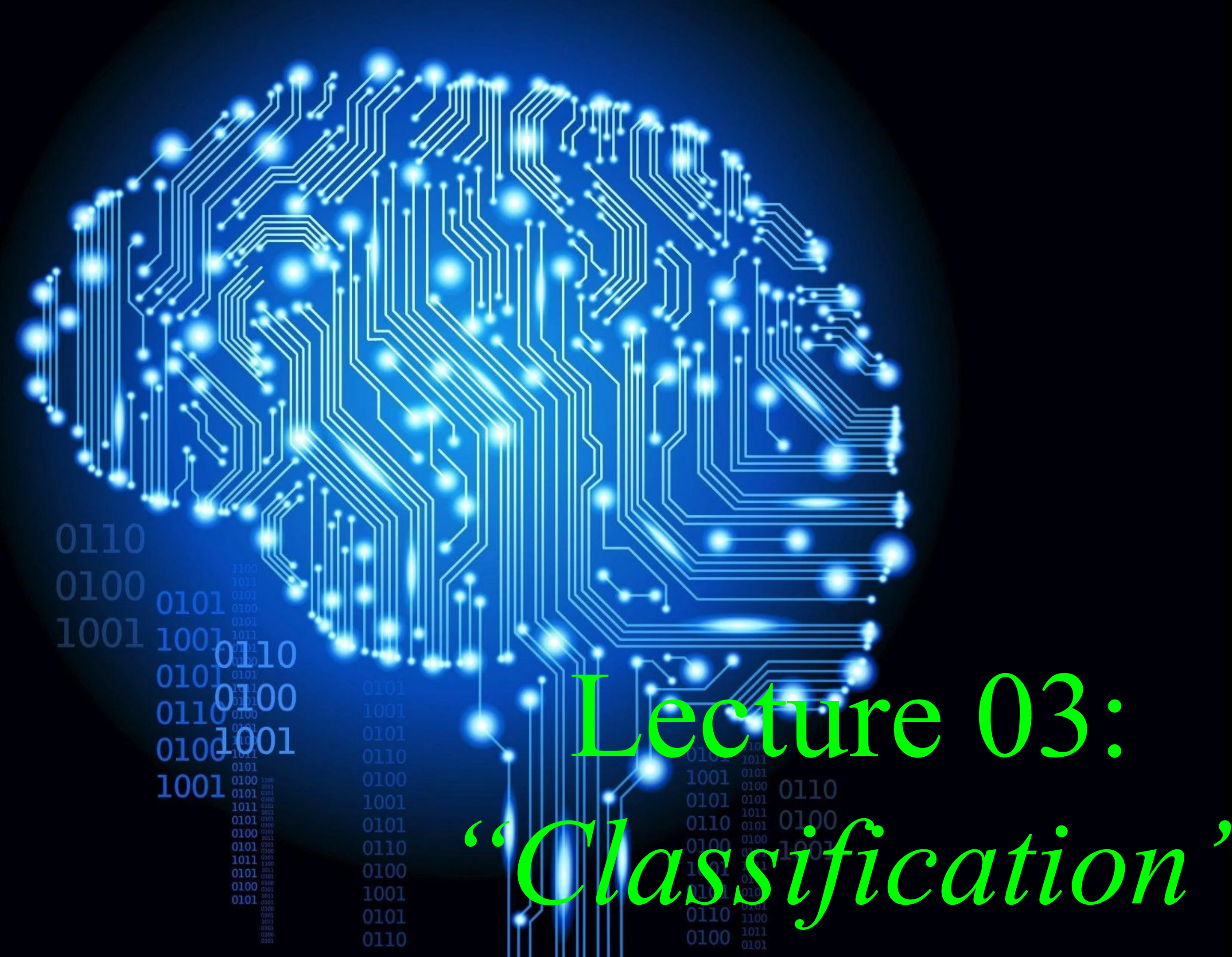
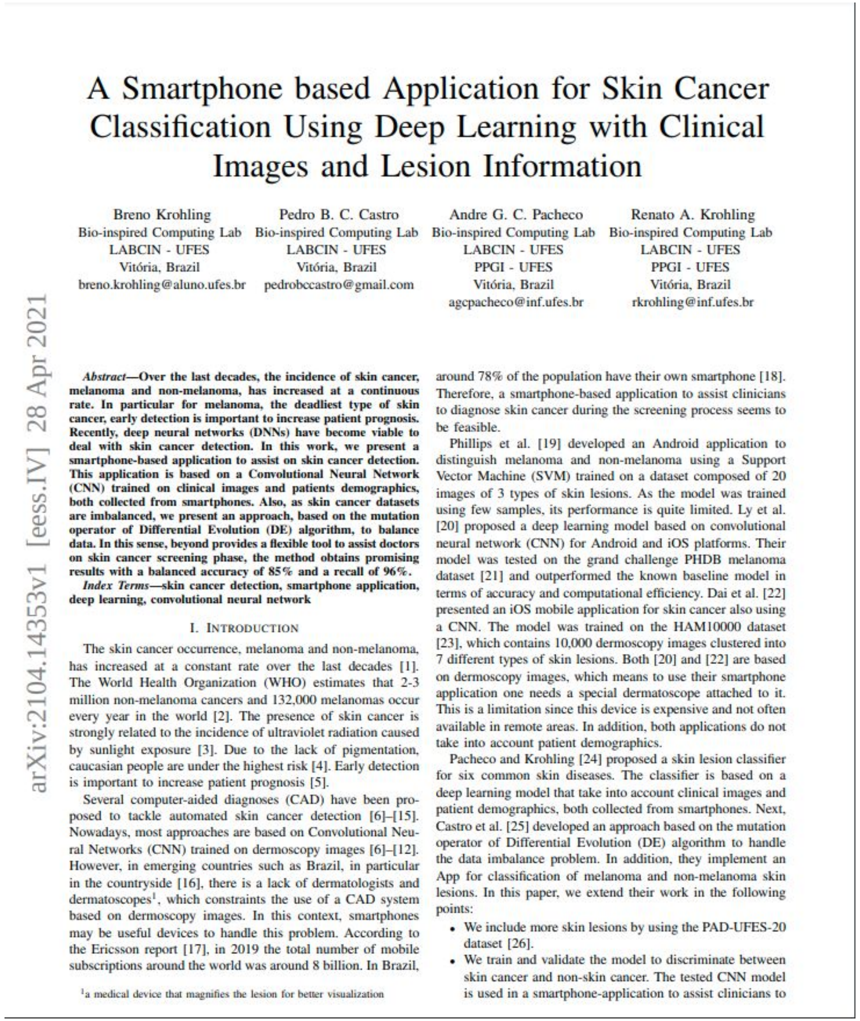


Lecture 03:
“Classification”



A Smartphone based Application for Skin Cancer Classification Using Deep Learning with Clinical Images and Lesion Information

Breno Krohling
Bio-inspired Computing Lab
LABCIN - UFES
Vitória, Brazil
breno.krohling@aluno.ufes.br

Pedro B. C. Castro
Bio-inspired Computing Lab
LABCIN - UFES
Vitória, Brazil
pedrobccastro@gmail.com

Andre G. C. Pacheco
Bio-inspired Computing Lab
LABCIN - UFES
PPGI - UFES
Vitória, Brazil
agcpacheco@inf.ufes.br

Renato A. Krohling
Bio-inspired Computing Lab
LABCIN - UFES
PPGI - UFES
Vitória, Brazil
rkrohling@inf.ufes.br

***Abstract*—Over the last decades, the incidence of skin cancer, melanoma and non-melanoma, has increased at a continuous rate. In particular for melanoma, the deadliest type of skin cancer, early detection is important to increase patient prognosis. Recently, deep neural networks (DNNs) have become viable to deal with skin cancer detection. In this work, we present a smartphone-based application to assist on skin cancer detection. This application is based on a Convolutional Neural Network (CNN) trained on clinical images and patients demographics, both collected from smartphones. Also, as skin cancer datasets are imbalanced, we present an approach, based on the mutation operator of Differential Evolution (DE) algorithm, to balance data. In this sense, beyond provides a flexible tool to assist doctors on skin cancer screening phase, the method obtains promising results with a balanced accuracy of 85% and a recall of 96%.**

***Index Terms*—skin cancer detection, smartphone application, deep learning, convolutional neural network**

I. Introduction

The skin cancer occurrence, melanoma and non-melanoma, has increased at a constant rate over the last decades [1]. The World Health Organization (WHO) estimates that 2-3 million non-melanoma cancers and 132,000 melanomas occur every year in the world [2]. The presence of skin cancer is strongly related to the incidence of ultraviolet radiation caused by sunlight exposure [3]. Due to the lack of pigmentation, caucasian people are under the highest risk [4]. Early detection is important to increase patient prognosis [5].

Several computer-aided diagnoses (CAD) have been proposed to tackle automated skin cancer detection [6]–[15]. Nowadays, most approaches are based on Convolutional Neural Networks (CNN) trained on dermoscopy images [6]–[12]. However, in emerging countries such as Brazil, in particular in the countryside [16], there is a lack of dermatologists and dermatoscopes[1], which constraints the use of a CAD system based on dermoscopy images. In this context, smartphones may be useful devices to handle this problem. According to the Ericsson report [17], in 2019 the total number of mobile subscriptions around the world was around 8 billion. In Brazil, around 78% of the population have their own smartphone [18]. Therefore, a smartphone-based application to assist clinicians to diagnose skin cancer during the screening process seems to be feasible.

Phillips et al. [19] developed an Android application to distinguish melanoma and non-melanoma using a Support Vector Machine (SVM) trained on a dataset composed of 20 images of 3 types of skin lesions. As the model was trained using few samples, its performance is quite limited. Ly et al. [20] proposed a deep learning model based on convolutional neural network (CNN) for Android and iOS platforms. Their model was tested on the grand challenge PHDB melanoma dataset [21] and outperformed the known baseline model in terms of accuracy and computational efficiency. Dai et al. [22] presented an iOS mobile application for skin cancer also using a CNN. The model was trained on the HAM10000 dataset [23], which contains 10,000 dermoscopy images clustered into 7 different types of skin lesions. Both [20] and [22] are based on dermoscopy images, which means to use their smartphone application one needs a special dermatoscope attached to it. This is a limitation since this device is expensive and not often available in remote areas. In addition, both applications do not take into account patient demographics.

Pacheco and Krohling [24] proposed a skin lesion classifier for six common skin diseases. The classifier is based on a deep learning model that take into account clinical images and patient demographics, both collected from smartphones. Next, Castro et al. [25] developed an approach based on the mutation operator of Differential Evolution (DE) algorithm to handle the data imbalance problem. In addition, they implement an App for classification of melanoma and non-melanoma skin lesions. In this paper, we extend their work in the following points:

- We include more skin lesions by using the PAD-UFES-20 dataset [26].
- We train and validate the model to discriminate between skin cancer and non-skin cancer. The tested CNN model is used in a smartphone-application to assist clinicians to

[1]a medical device that magnifies the lesion for better visualization

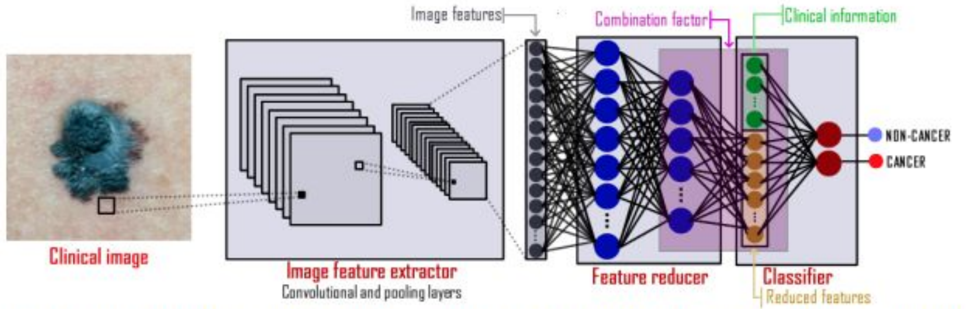
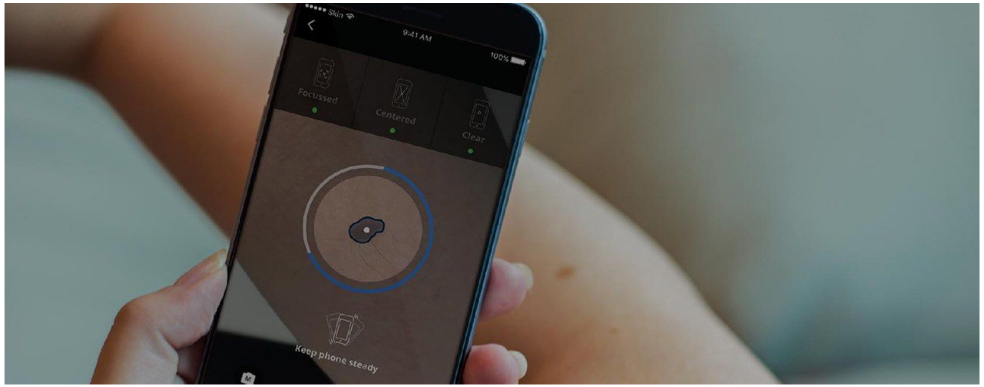
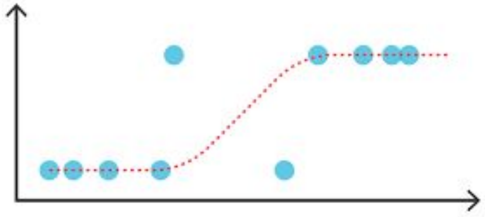


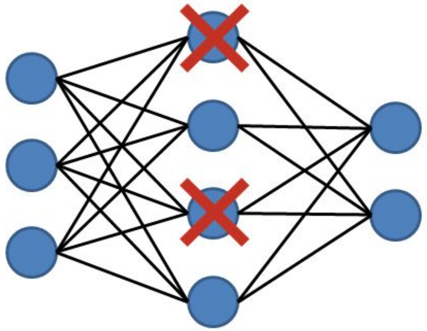
Fig. 1: The illustration of the model proposed by Pacheco and Krohling [24]. In this work, we modified the last layer for skin cancer classification

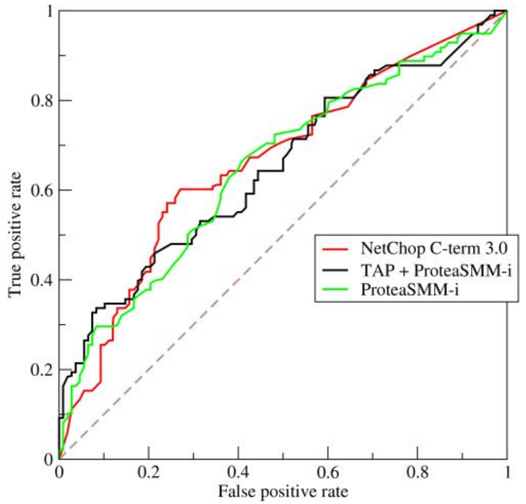


B. Krohling, P. B. C. Castro, A. G. C. Pacheco, and R. A. Krohling, “A Smartphone based Application for Skin Cancer Classification Using Deep Learning with Clinical Images and Lesion Information,” *arXiv [eess.IV]*, Apr. 28, 2021. [Online]. Available: http://arxiv.org/abs/2104.14353

This Week...





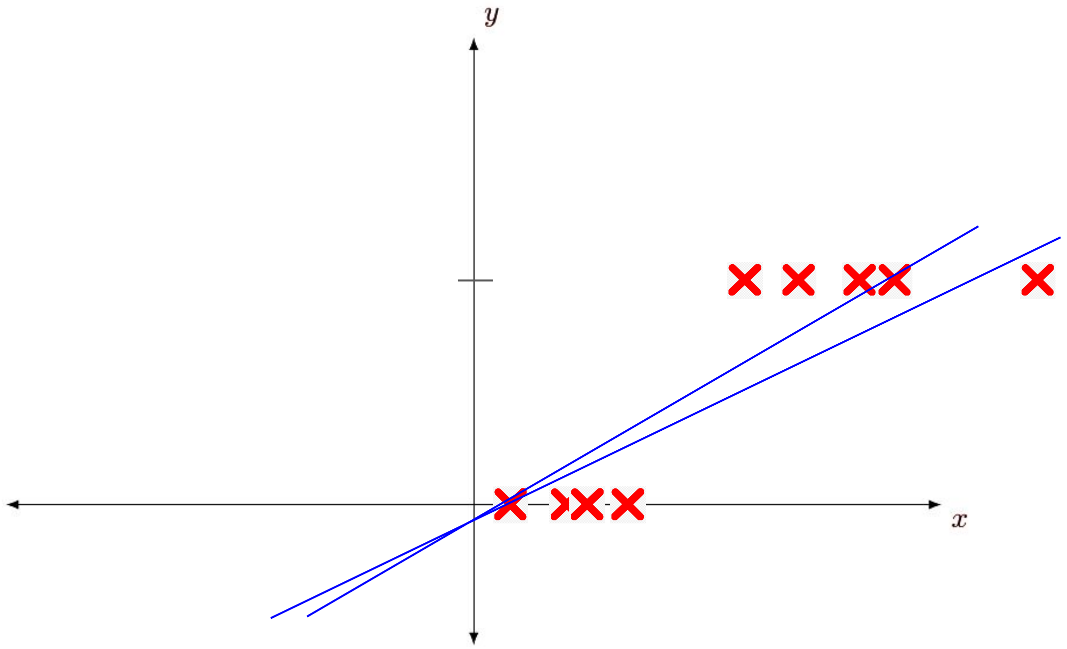


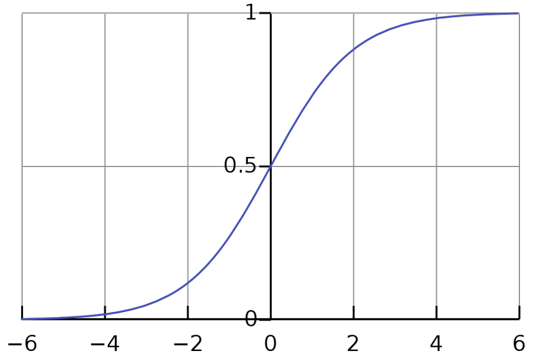
- Logistic Regression
- Regularization
- Performance Evaluation

The Big 2

Regression	Classification
● $\mathscr{Y}$ is continuous: i.e. {0.2,3.4,4.1, 5.3,3.2}	● $\mathscr{Y}$ is discrete, finite: i.e. {0,1},{cat,dog,fish}
● Space of labels is always numeric	● Space of labels is often categorical
● Different loss functions more appropriate (i.e. L1, L2)	● Different loss functions more appropriate (i.e. Cross entropy ~ negative log loss)
● Want Pr(Y | X=x), learn from labeled (x,y) pairs, optimize a loss function or likelihood	● Want Pr(Y | X=x), learn from labeled (x,y) pairs, optimize a loss function or likelihood

Logistic Regression

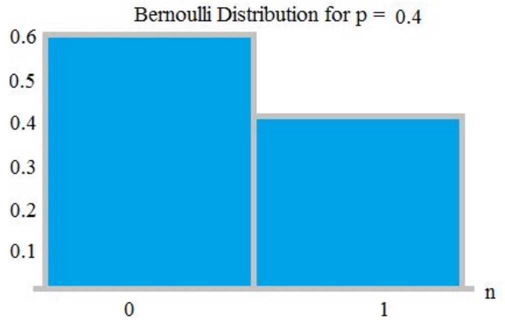




$$\varsigma(x) = \frac{1}{1+e^{-x}}$$

Bernoulli Random Variable

- Discrete, random variable $Y \in \{0,1\}$
- Parameter p is $\Pr(Y=1)$; $(1-p)$ is $\Pr(Y=0)$
- E[Y] is p



Maximum Likelihood Regression

- (Distribution) Define $\Pr(Y|X=x)$ from probability distribution
- (Form) Define $E[Y|X=x]$ as function of x and parameters b_0, b_1, ...
- Given dataset, find b_0, b_1, … which maximize log likelihood

For Bernoulli, $E[Y=1\ |X=x]$ is $\Pr(Y=1|X=x)=p(x)$. Let $p(x;\mathbf{b})=\varsigma(b_0+b_1x)$

Logistic Regression

- A probabilistic classifier which estimates Pr(Y=1|X=x), i.e. $p(x)$ or $p(x;\mathbf{b})$

$$p(x;\mathbf{b}) = \varsigma\left(b_0 + b_1 x\right) = \frac{1}{1 + e^{-\left(b_0 + b_1 x\right)}}$$

$$p(x;\boldsymbol{b}) = Pr(Y=1 \mid X = x;\boldsymbol{b})$$

$$1 - p(x;\boldsymbol{b}) = Pr(Y=0 \mid X = x;\boldsymbol{b})$$

Log Likelihood

$$data = \left\{ \left(x_1, y_1 \right), \left(x_2, y_2 \right), \ldots, \left(x_n, y_n \right) \right\}$$

$$\ell(\mathbf{b}; data) = \sum_i \log\left(Pr\left(Y = y_i \,\middle|\, X = x_i; \mathbf{b} \right) \right)$$

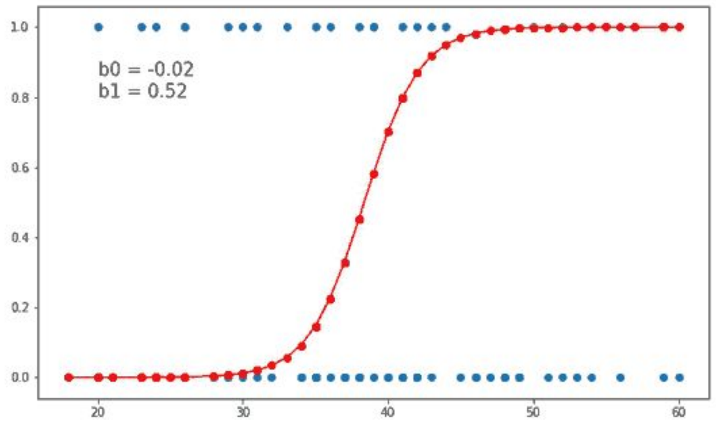
$$\ell(\mathbf{b}; data) = \sum_i \log \begin{cases} p\left(x_i; \mathbf{b} \right), & y_i = 1 \\ 1 - p\left(x_i; \mathbf{b} \right), & y_i = 0 \end{cases}$$

Log Likelihood

$$\ell(\mathbf{b}; data) = \sum_i \log \begin{cases} p(x_i; \mathbf{b}), & y_i = 1 \\ 1 - p(x_i; \mathbf{b}), & y_i = 0 \end{cases}$$

$$\ell(\mathbf{b}; data) = \sum_i \log\left[y_i \, p(x_i; \mathbf{b}) + (1 - y_i)(1 - p(x_i; \mathbf{b})) \right]$$

$$\boxed{\ell(\mathbf{b}; y_1, y_2, \ldots, y_n) = \sum_i \log\left[y_i \, p(x_i; \mathbf{b}) + (1 - y_i)(1 - p(x_i; \mathbf{b})) \right]}$$


b0 = -0.02
b1 = 0.52
1.0
0.8
0.6
0.4
0.2
0.0
20
30
40
50
60

Regularization

Consider a dataset of two samples: (1, 1), (-1,0)

$$\ell\left(\mathbf{b};y_1,y_2,\ldots,y_n\right)=\sum_i \log\left[y_i\,p\left(x_i;\mathbf{b}\right)+\left(1-y_i\right)\left(1-p\left(x_i;\mathbf{b}\right)\right)\right]$$

$$\ell=\log(\,p(\,1;\mathbf{b})\,)+\log(\,1-p(\,-1;\mathbf{b})\,)$$

$$\ell=\log\left(\frac{1}{1+e^{-\left(b_0+b_1\right)}}\right)+\log\left(1-\frac{1}{1+e^{-\left(b_0-b_1\right)}}\right)$$

For a fixed b$_0$, what is the best b$_1$?

Answer: ∞ (no matter what b$_0$ is)

https://www.geogebra.org/m/gpqjqwck

Regularization

- Regularization = Optimize likelihood + penalty (on size of parameters b_0, b_1)
- The penalty prevents infinite optimal solutions, big parameters → **instability, overfitting**
- $objective(\mathbf{b}) = -\ell(\mathbf{b}; data) + penalty(\mathbf{b})$

L1 (LASSO) penalty:

$$\sum_j |b_j| = \|\mathbf{b}\|_1$$

L2 (Ridge) penalty:

$$\sum_j b_j^2 = \|\mathbf{b}\|^2$$

https://www.geogebra.org/m/gpqjqwck

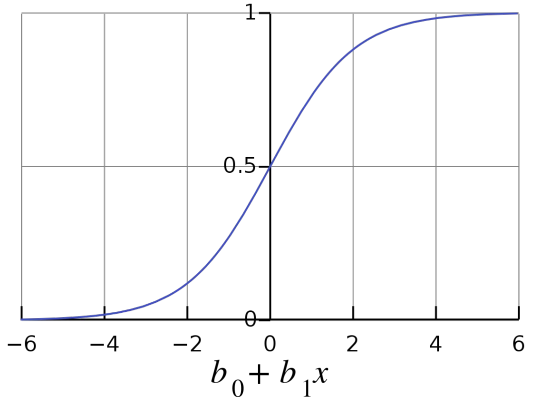
Evaluation: Decision Boundary

- Likelihood good for training, but difficult for evaluating a classifier
- Instead we usually focus on “how many predictions were right vs. wrong”
- How do we know right vs wrong? We need to **threshold** and create a **decision boundary**

Threshold ex:

$$\hat{y} = \begin{cases} 1, & Pr(Y=1 \mid X=x) \geq 0.5 \\ 0, & Pr(Y=1 \mid X=x) < 0.5 \end{cases}$$

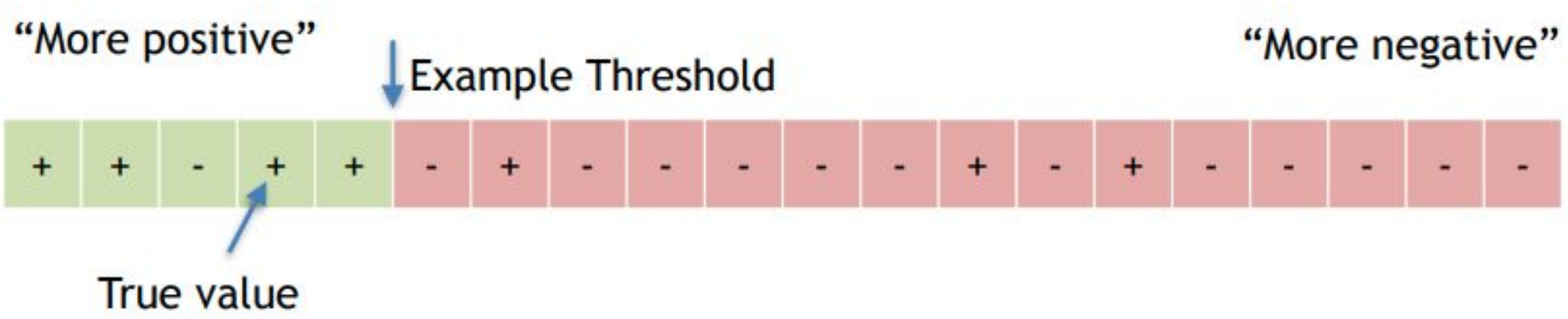
Corresponding Decision Boundary:



$$b_0 + b_1 x = 0$$

Evaluation: Ranking

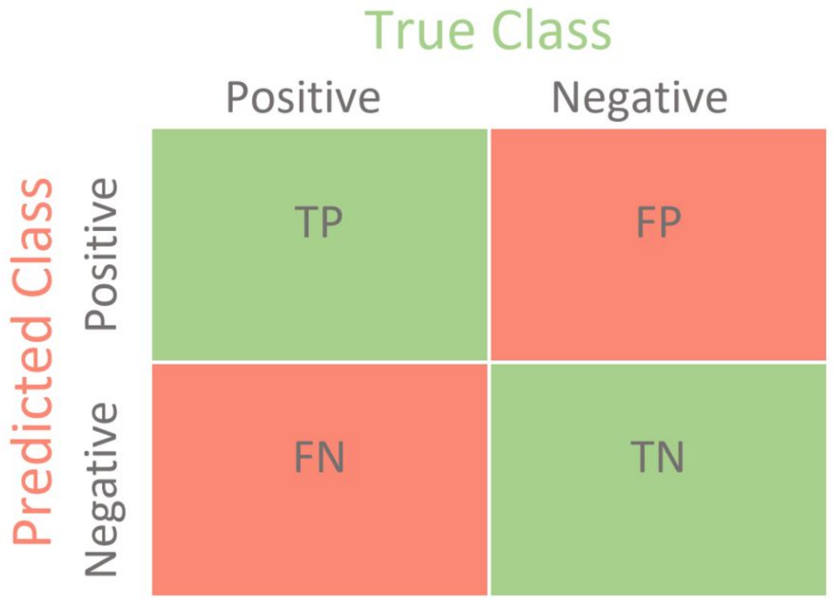
- In some cases we can rank data samples from “most positive” to “most negative”



Evaluation: Probabilities

- In logistic regression our classifier produces probabilities. What if it mis-classifies with high confidence? (i.e. Pr(Y=1 | X=x)=.99 but y=0)
- One solution: penalize misclassifications with high confidence more than misclassifications with lower confidence.

Confusion Matrix (2 Class)



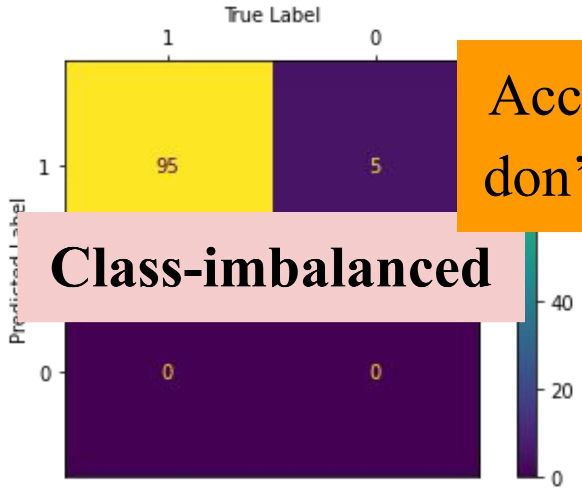
- TP = “True Positive”. i.e. your classifier predicted 1; it was correct.
- FP = “False Positive”. i.e. your classifier predicted 1; it was incorrect (Type I error).
- FN = “False Negative”. i.e. your classifier predicted 0; it was incorrect (Type II error).
- TN = “True Negative”. i.e. your classifier predicted 0; it was correct.

Accuracy

$$\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n}\begin{cases}1 \text{ if } y_i = \hat{y}_i \\ 0 \text{ if } y_i \neq \hat{y}_i\end{cases}$$

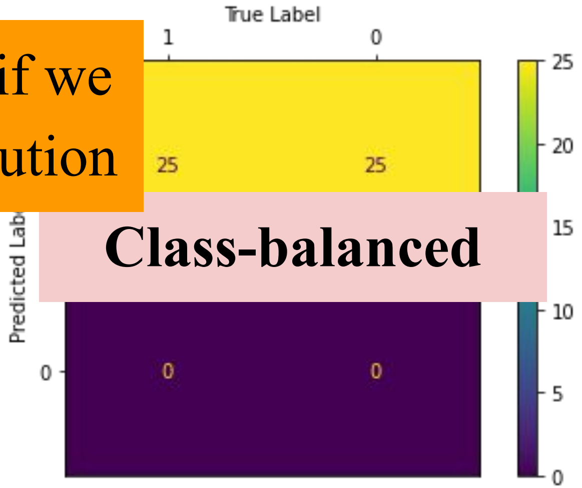
An alternative metric is called the "error rate" or "0-1 Loss". This is (1 - Accuracy).

Can you think of any issues?



Class-imbalanced

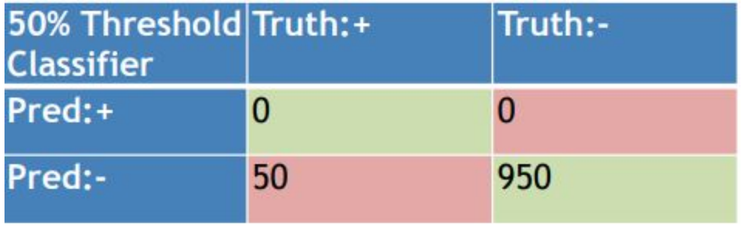
Accuracy isn't meaningful if we don't know the class distribution



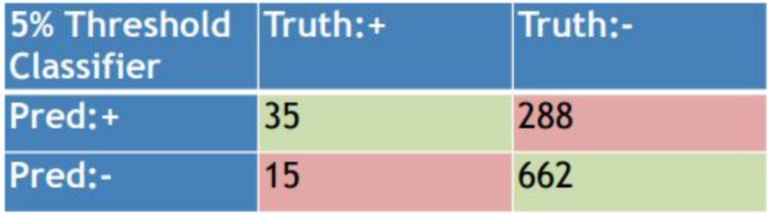
Class-balanced

Accuracy

- Say we have a dataset of 950 negative-class samples, 50 positive-class samples
- If we **lower our positive class threshold**, our logistic regression accuracy changes



50% Threshold Classifier	Truth:+	Truth:-
Pred:+	0	0
Pred:-	50	950

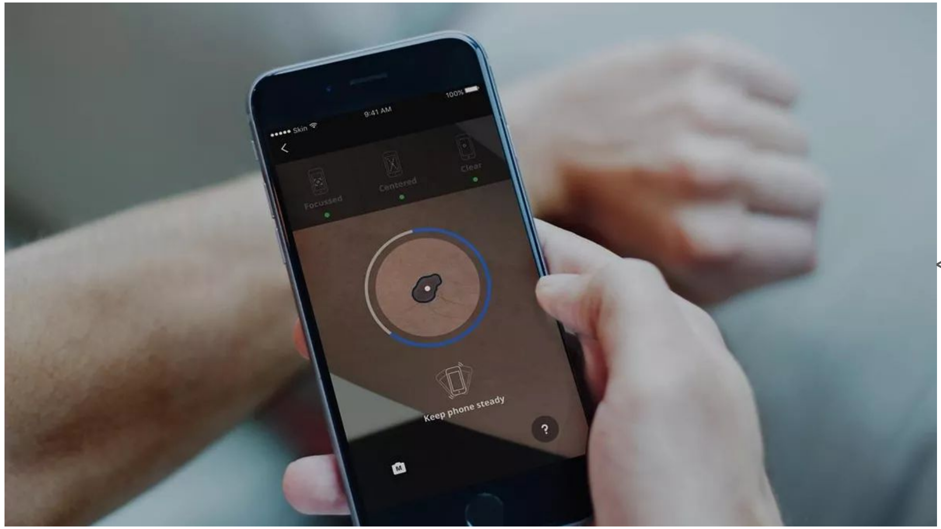


5% Threshold Classifier	Truth:+	Truth:-
Pred:+	35	288
Pred:-	15	662

If we allow more samples to be classified as positive, we will correctly identify some of the 50 positive samples; however, we will also incorrectly label some of the negative samples.

Domain-dependence

- Different domains will value different measures and results: medicine is one example
- What is more harmful? Type I error or Type II error?



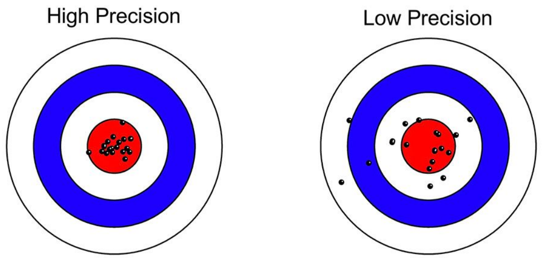
Predict disease incorrectly (Type I)

Fail to detect the disease (Type II)

Precision

- What % of all positive predictions were correct?

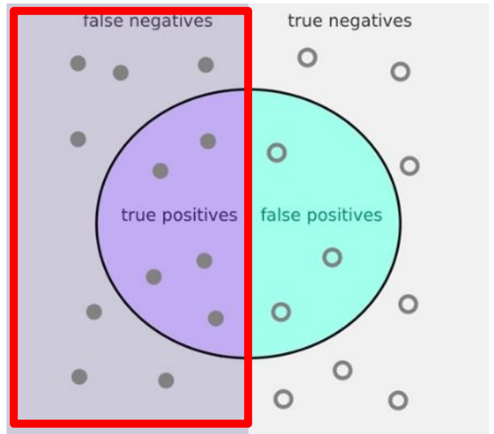
$$Precision = \frac{\#\ True\ Positive}{\#\ Predicted\ Positive}$$



Recall

- What % of all positive samples were recalled?

$$Recall = \frac{\#\ True\ Positive}{\#\ Class\ Positive}$$



F-Measure

- A combined metric which accounts for precision and recall in a single measure.

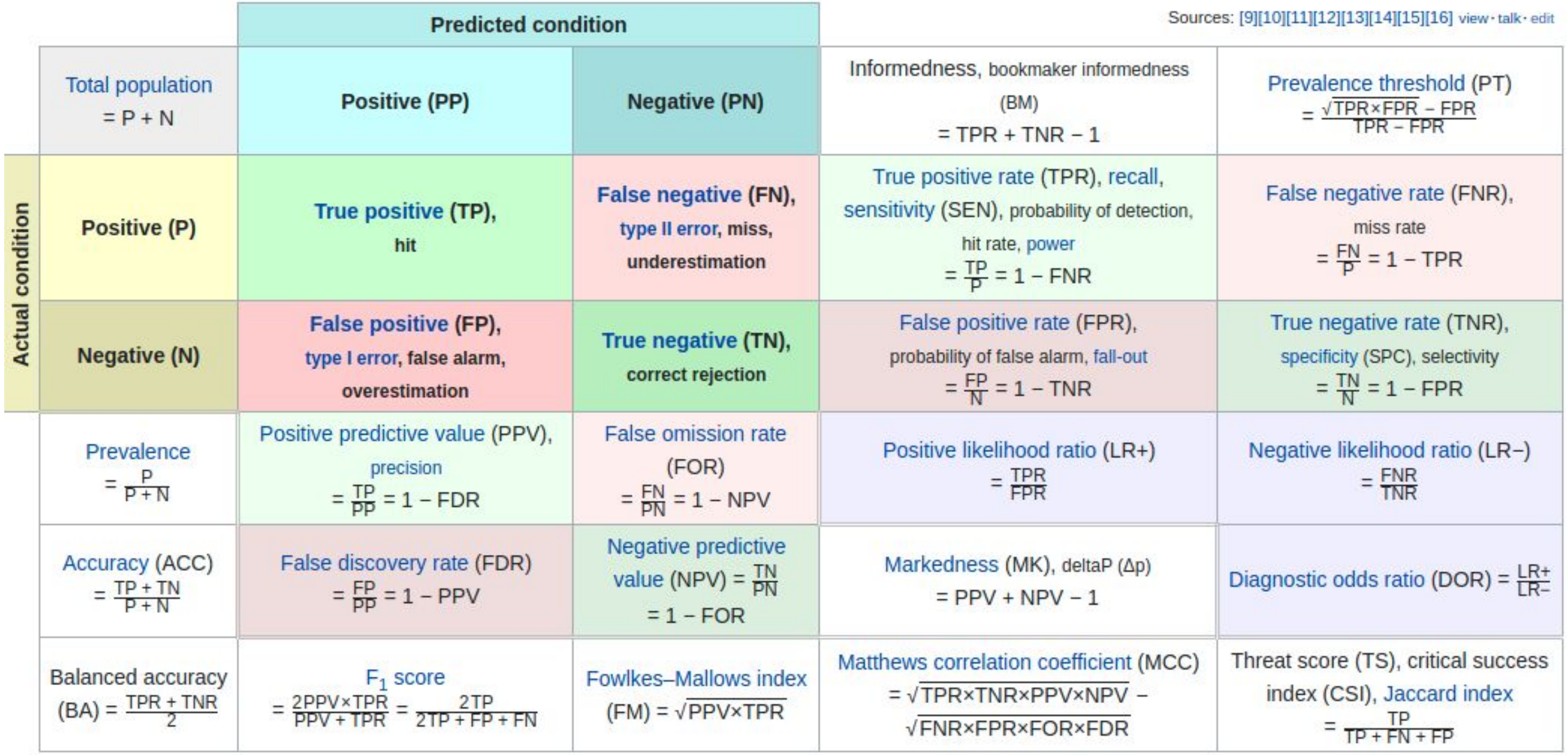
$$F - measure = 2 \cdot \left(\frac{Precision \cdot Recall}{Precision + Recall} \right)$$

Sensitivity, Specificity

- Sensitivity is the same as Recall.

$$\text{Specificity} = \frac{\#\ True\ Negative}{\#\ Class\ Negative}$$

Other Measures



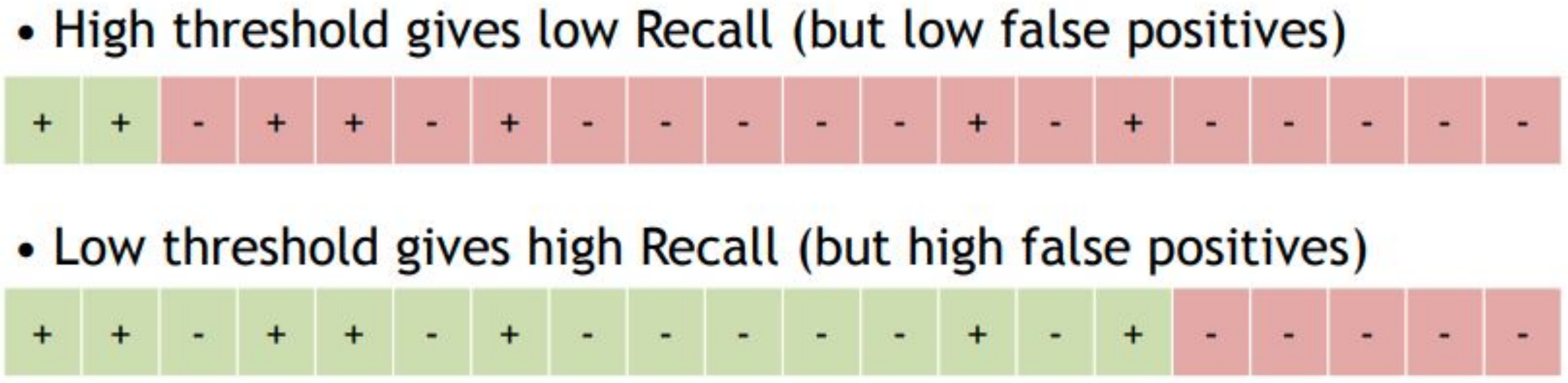
Sources: [9][10][11][12][13][14][15][16] view · talk · edit

		Predicted condition			
	Total population $= P + N$	**Positive (PP)**	**Negative (PN)**	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	**Positive (P)**	**True positive (TP), hit**	**False negative (FN), type II error, miss, underestimation**	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
	Negative (N)	**False positive (FP), type I error, false alarm, overestimation**	**True negative (TN), correct rejection**	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN}$ $= 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2 PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

https://en.wikipedia.org/wiki/Evaluation_of_binary_classifiers

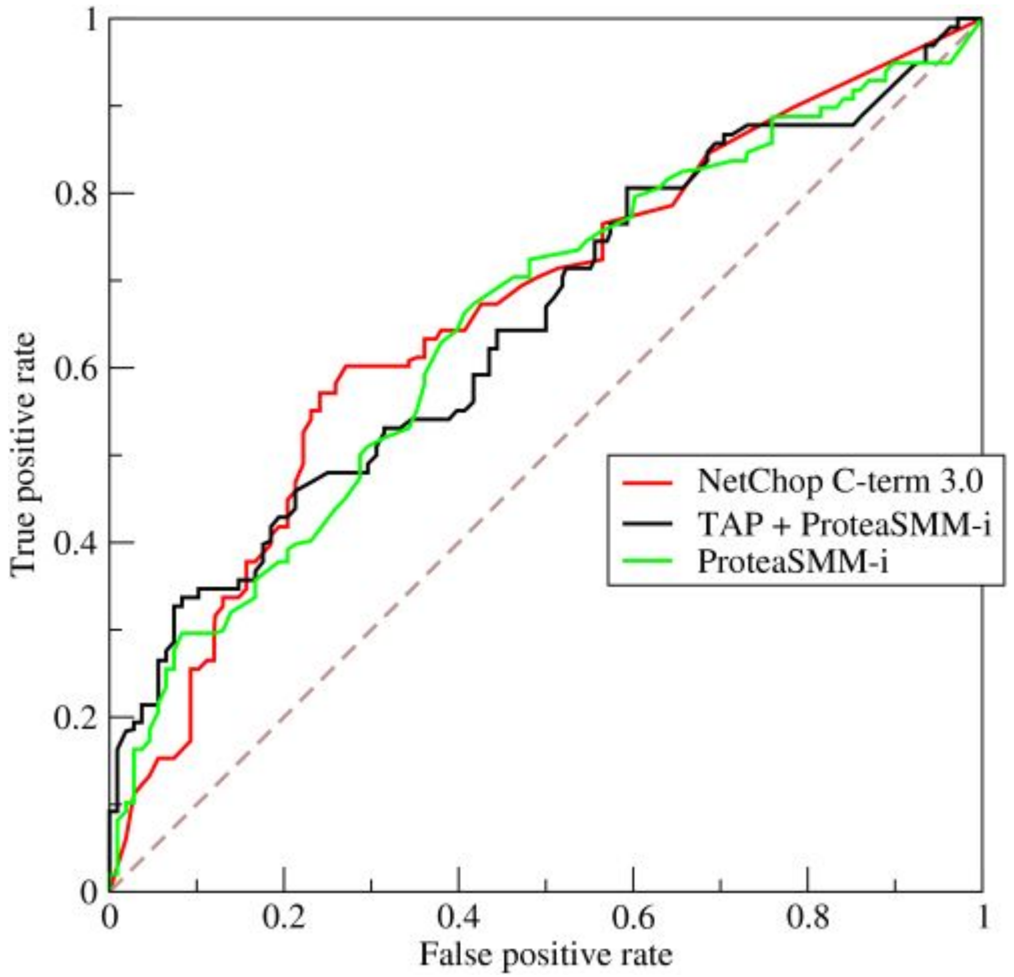
ROC Curves

- ROC=Receiver Operating Characteristic. What does it do?
- Answer: **It shows the performance of a classifier at all classification thresholds**



- High threshold gives low Recall (but low false positives)
- Low threshold gives high Recall (but high false positives)

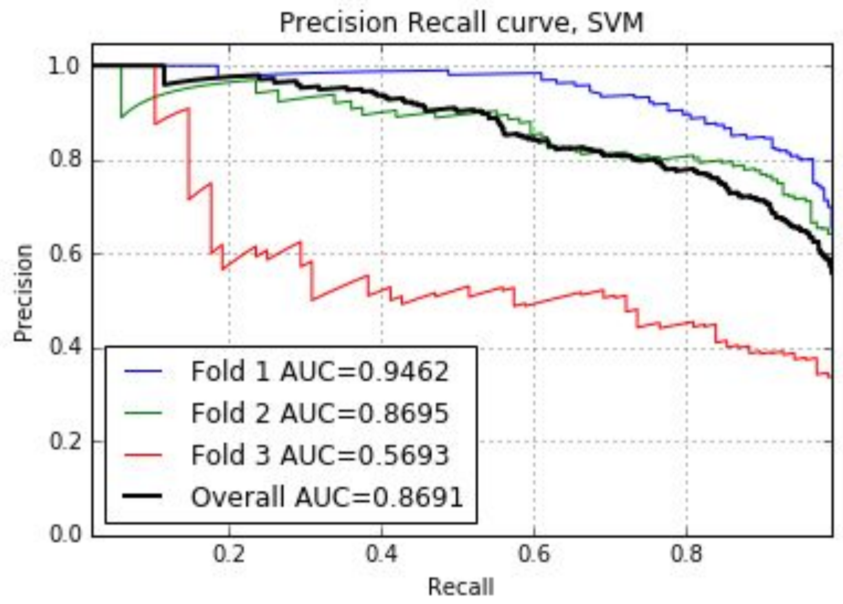
- An ROC Curve plots True Positive Rate or TPR (Recall, also called Sensitivity) vs. False Positive Rate or FPR (1-Specificity). TPR=TP/P, FPR=FP/N

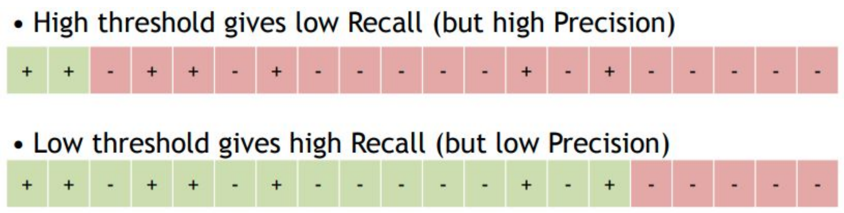


- If Area Under Receiver Operating Characteristic Curve (AUROC) is 1, classifier is perfect
- If AUROC is 0.5, classifier is no better than random chance.
- If AUROC is 0, classifier is worst possible (TPR=0, FPR=1).

PRC

- PRC=Precision-Recall Curve. Plot Precision on y-axis, Recall on x-axis for all possible thresholds





• High threshold gives low Recall (but high Precision)

+	+	-	+	+	-	+	-	-	-	-	-	+	-	+	-	-	-	-	-

• Low threshold gives high Recall (but low Precision)

+	+	-	+	+	-	+	-	-	-	-	-	+	-	+	-	-	-	-	-

- If Area Under PRC (AUPRC) is 1, classifier is perfect

https://stackoverflow.com/questions/45311863/compare-classifiers-using-precision-recall-curve

Multiple Features

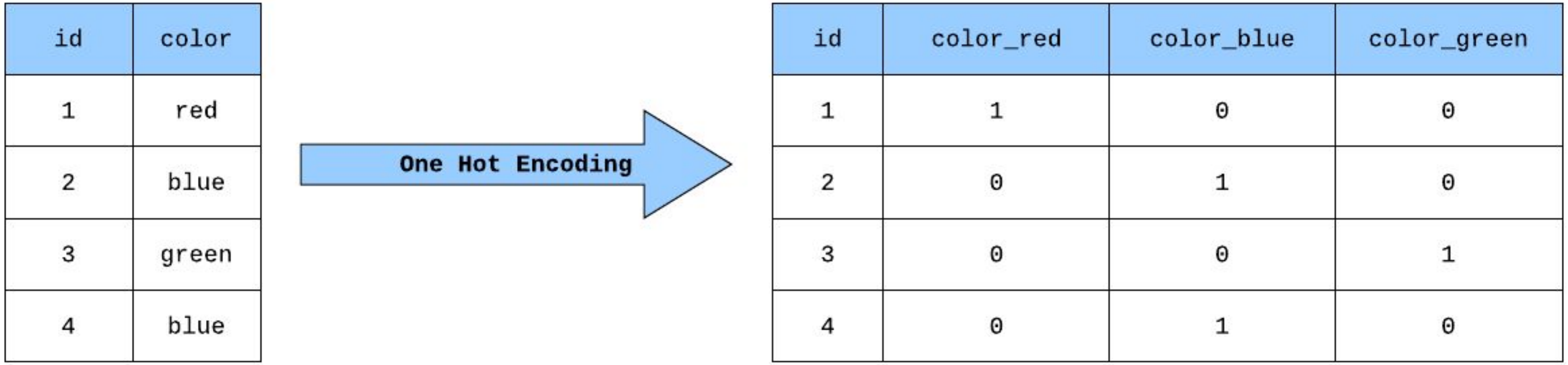
- In this lecture we have thought only of models whereby predictions are of the form b_0+b_1x
- Let’s now consider predictions wherein we use multiple features $x_0,x_1,...,x_n$
- i.e. for logistic regression:

$$p(\mathbf{x};\mathbf{b}) = \varsigma(\mathbf{x}^T\mathbf{b})$$

- **x** here is a column vector of n features and $x_0=1$ (intercept term). **b** is a column vector containing the model parameters (same dimension as **x**). Note: $\mathbf{x}^T\mathbf{b}$ is a scalar value.

Multiclass Classifiers

- Define K classes numbered 1,...,K. Also define random variables $Y_1,...,Y_K$ where $y_k=1$ for a class k observation (realization is **y**) → we call this **one-hot encoding (ohe)**.


One Hot Encoding

id	color
1	red
2	blue
3	green
4	blue

id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

- Define $p_k(\mathbf{x};\mathbf{B}) = Pr\left(Y_k = 1 \mid X = \mathbf{x}\right)$ for $j = 1, 2, \ldots, k$

Log Likelihood (Multiclass)

Note: this is a vector of features

$$\text{data} = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$$

$$\ell(\mathbf{B}; \text{data}) = \sum_i \sum_k y_{ik} \log\Big(Pr\big(Y_{ik} = 1 \mid X = x_i; \mathbf{B}\big)\Big)$$

$$\varsigma(\mathbf{z})_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$$

$$p_k(\mathbf{x}; \mathbf{B}) = \varsigma(\mathbf{z})_k$$

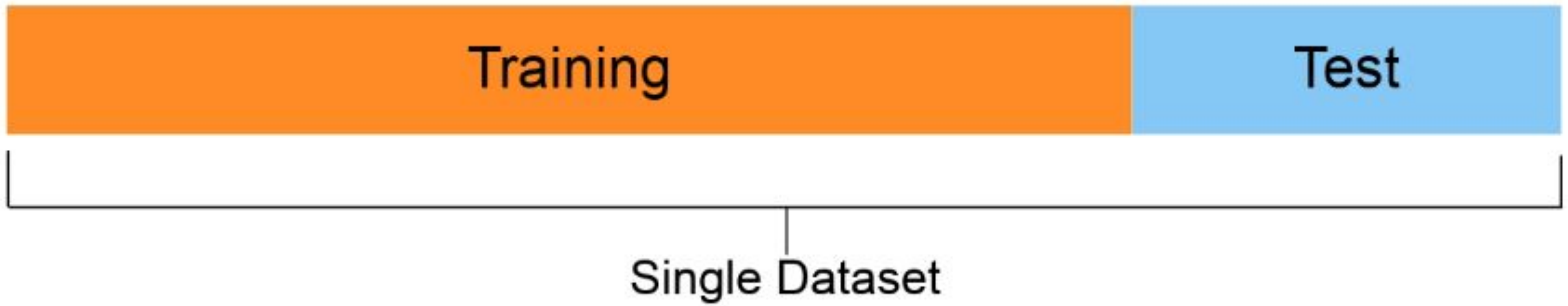
where

$$z_k = \mathbf{x}^\top \mathbf{b}_k \text{ for } k = 1 \ldots K-1$$

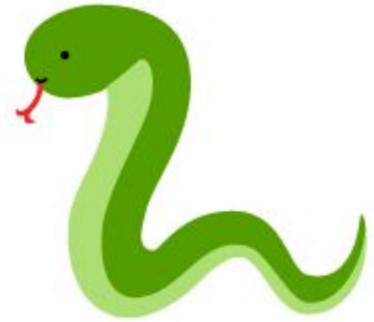
$$z_K = 0$$

The Double-Dipping Problem

- Our data comes from the joint distribution of (X,Y). We want our model to make predictions from unseen data
- PROBLEM: Our training data comes from here. If we **evaluate** our model with the same data we **trained on**, our estimates will have **optimistic bias** = double dipping
- SOLUTION: split data into a training set and a test set. Though, note that if we do this we will have less data for training → how to fix? We will discuss this shortly.



Let’sss try it in Python...



Summary

- Logistic Regression
- Sigmoid Function
- Regularization
- Classifier Evaluation
  - Probabilistic Evaluation
  - Label Evaluation
  - Ranking Evaluation
- Class Imbalances
- Plots: Confusion, ROC, AUC
- Measures (Precision, Recall, F1, …)
- Multi-class Classification