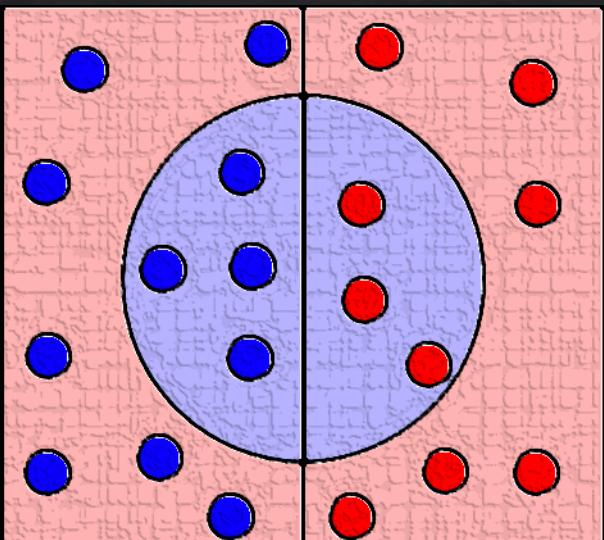


# Supervised Learning – Performance Evaluation

Machine Learning Methods – Lecture 4



June 2021



# Multiclass Classification (K-NN)

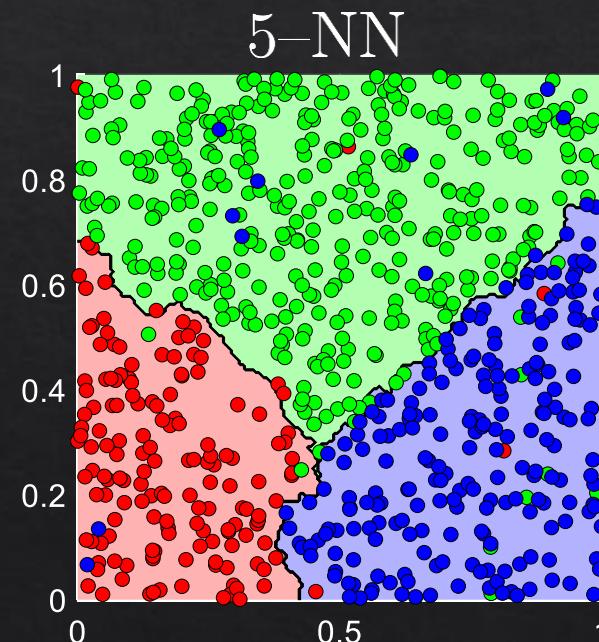
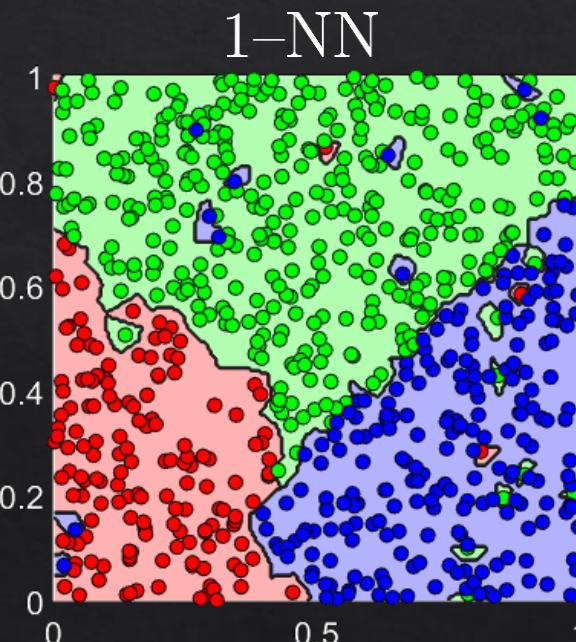
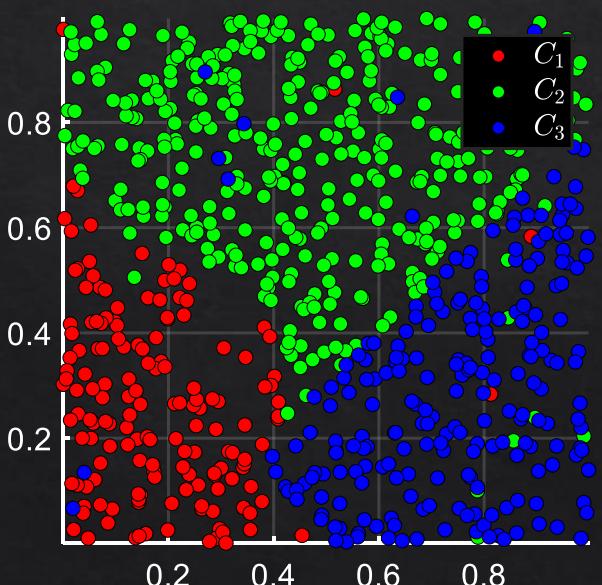
- Consider the multi-class classification task:

$$\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N, \quad y_i \in \mathcal{Y} = \{C_1, C_2, \dots, C_K\}$$

- K-NN and other classifiers can be immediately generalized into multiclass classification.

K-NN

$$f_{1-\text{NN}}(\mathbf{x}) = \hat{y}_{i(\mathbf{x})}$$
$$\hat{i}(\mathbf{x}) = \arg \min_i d(\mathbf{x}, \mathbf{x}_i)$$



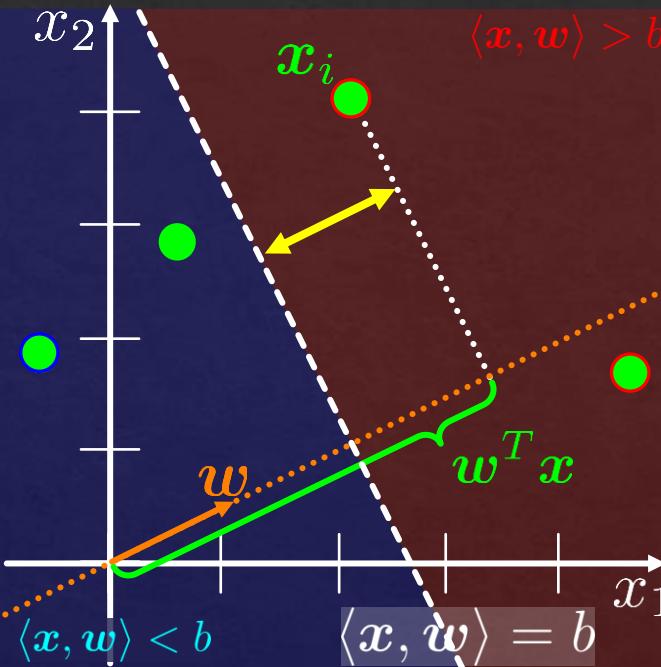
What about binary linear classifiers (such as SVM)?

# Classification Score (Confidence)

- A linear classifier:

$$f(\mathbf{x}) = \text{sign} \left( \underbrace{\mathbf{w}^T \mathbf{x} - b}_{\text{distance from the decision boundary}} \right)$$

distance from the  
decision boundary



- The classification score of linear classifiers:

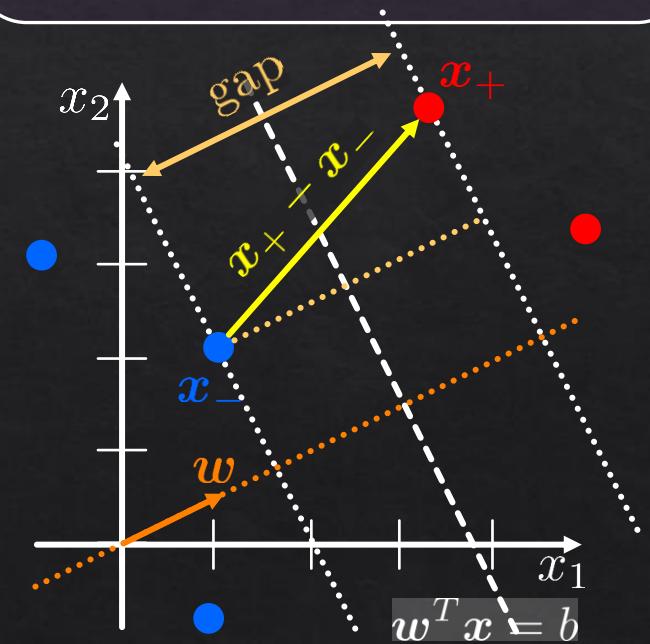
$$\text{classification score} := \mathbf{w}^T \mathbf{x} - b$$

$$\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$$
$$y_i \in \mathcal{Y} = \{C_1, C_2, \dots, C_K\}$$

SVM

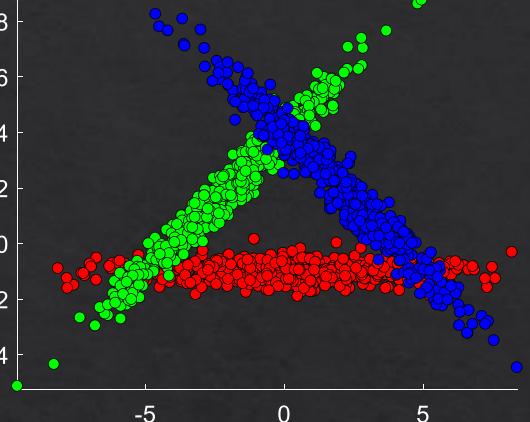
$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i$$

$$\xi_i := \max \{0, 1 - y_i (\mathbf{w}^T \mathbf{x}_i - b)\}$$



- Most classifiers has some classification score (confidence).

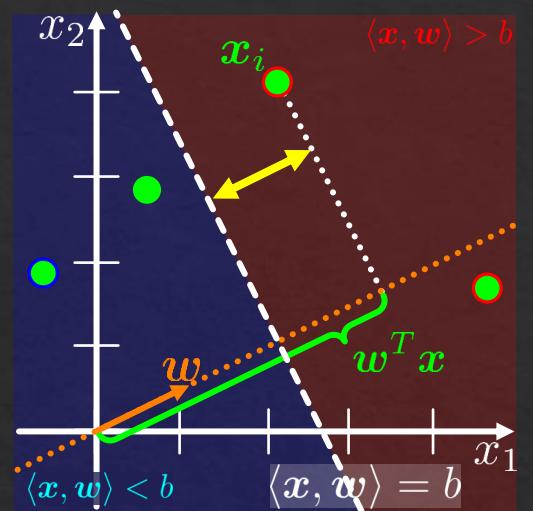
# Multiclass Classification (Coding)



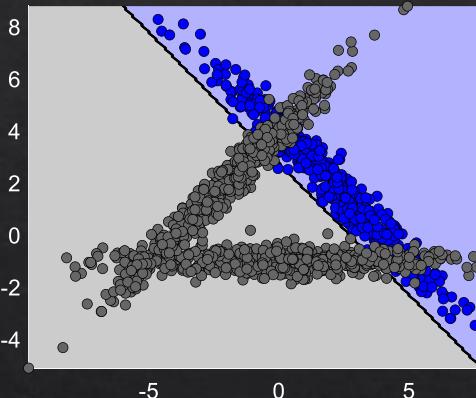
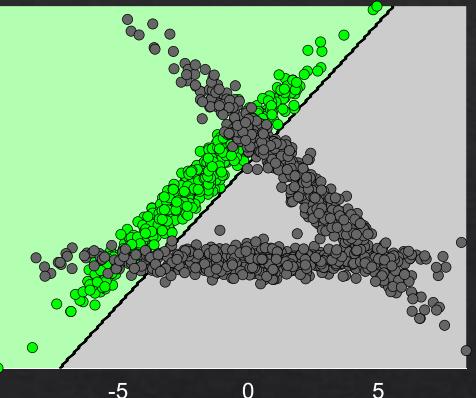
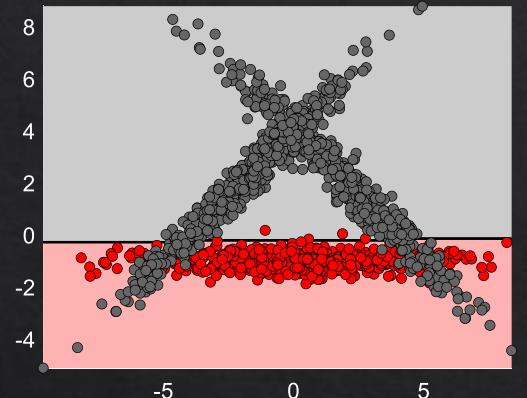
$$y_i \in \mathcal{Y} = \{C_1, C_2, \dots, C_K\}$$

$$f(\mathbf{x}) = \text{sign} \underbrace{\left( \mathbf{w}^T \mathbf{x} - b \right)}_{\text{distance from the decision boundary}}$$

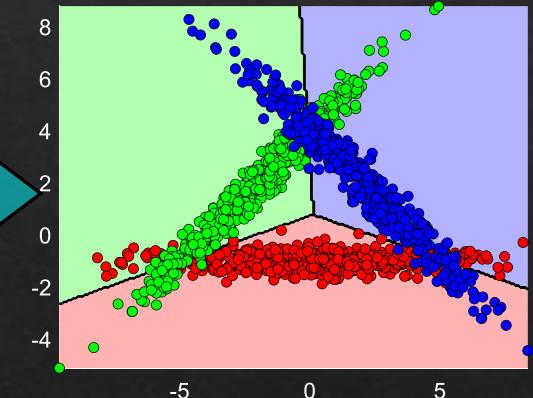
$$\text{classification score} := \mathbf{w}^T \mathbf{x} - b$$



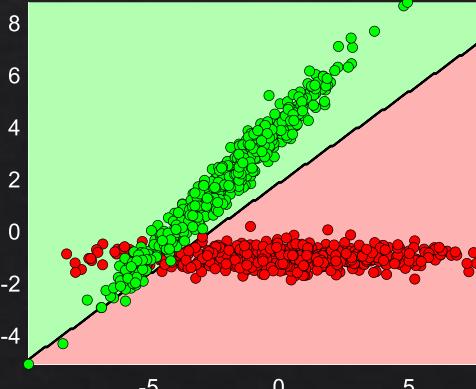
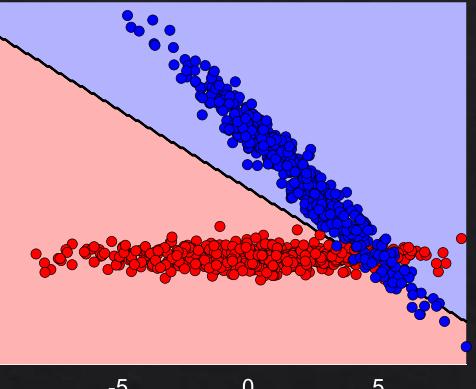
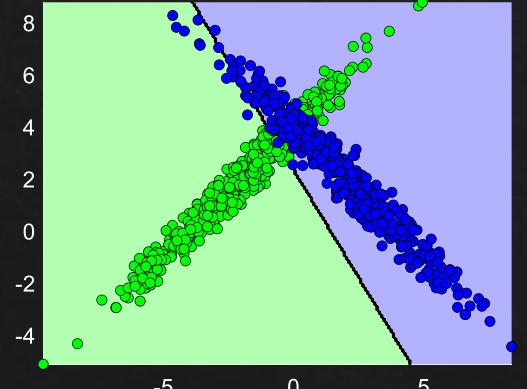
- One vs All:



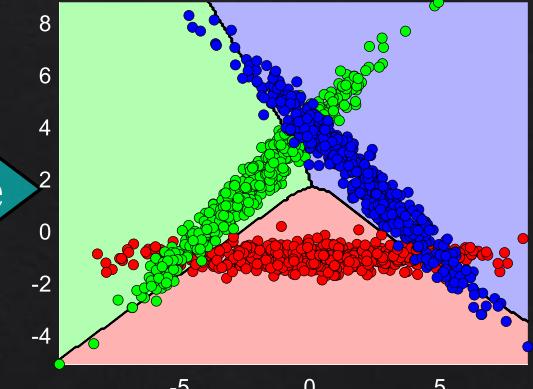
$K$  classifiers  
One vs All  
highest score



- One vs One:



$\frac{1}{2}K(K-1)$  classifiers  
One vs One  
most predictions



# MNIST

- The MNIST data set:



$$\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{60,000}$$

$$\mathcal{D}_{\text{test}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{10,000}$$



$$28 \in \mathbb{R}^{28 \times 28} = \mathbb{R}^{784}$$

$$\mathbf{x}_i \in \mathbb{R}^{784}$$

$$y_i \in \{0, 1, 2, \dots, 9\}$$

# Confusion Matrix



- Training an SVM classifier on  $N = 3,000$  samples with  $C = 0.0004$ :

Train accuracy: 80.8333%

True Class	0	1	2	3	4	5	6	7	8	9		87.1%	12.9%	
	278		1	3		7	12	1	14	3		97.7%	2.3%	
0		344	1	2			1		3	1		78.8%	21.2%	
1			2	12	238	4	6		8	7	21	4	80.7%	19.3%
2	2	12	238	4	6			8	7	21	4		55.0%	45.0%
3	2	9	12	247	1	4	3	2	19	7			56.3%	43.7%
4	1	7			144			5			105		90.2%	9.8%
5	3	28		41	1	142	10	1	13	13			82.1%	17.9%
6		11	4			10	275	1	4				79.3%	20.7%
7	1	11	2		2			257	2	38			90.5%	9.5%
8	1	15	3	16			5	1	234	20			95.9%	77.3%
9	2	8	1	7	4	2		1	3	266			4.1%	22.7%
	95.9%	77.3%	90.8%	77.2%	91.1%	86.1%	86.2%	94.8%	74.8%	58.2%				
	4.1%	22.7%	9.2%	22.8%	8.9%	13.9%	13.8%	5.2%	25.2%	41.8%				

Test accuracy: 79.9333%

True Class	0	1	2	3	4	5	6	7	8	9		86.9%	13.1%
	253		5	2		14	9	1	6	1		97.6%	2.4%
0		331										73.1%	26.9%
1			34	223	7	4						83.6%	16.4%
2	2	34	223	7	4			7	4	20	4	58.2%	41.8%
3		4	6	250			10	1	7	17	4	50.0%	50.0%
4	1	7	1		164			3		3	103	89.0%	11.0%
5	3	23	1	57	1	137	16	4	13	19		81.3%	18.7%
6	4	5	8		3	9	260	2			1	81.8%	18.2%
7	1	22	7	1				269	6	25		93.7%	75.2%
8	3	10		17		3	4	3	243	14		6.3%	24.8%
9	3	4	3	2	3			3	4	268		12.2%	25.6%
	93.7%	75.2%	87.8%	74.4%	93.7%	79.2%	86.7%	91.8%	75.9%	61.0%			
	6.3%	24.8%	12.2%	25.6%	6.3%	20.8%	13.3%	8.2%	24.1%	39.0%			

SVM

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i$$

$$\xi_i := \max \left\{ 0, 1 - y_i (\mathbf{w}^T \mathbf{x}_i - b) \right\}$$

K-NN

$$f_{1-\text{NN}}(\mathbf{x}) = \hat{y}_{i(\mathbf{x})}$$

$$\hat{i}(\mathbf{x}) = \arg \min_i d(\mathbf{x}, \mathbf{x}_i)$$

$C$  and  $K$  are hyper-parameters  
What is the optimal value of  $C$ ?

# Confusion Matrix (SVM and the hyperparameter C)

Train accuracy ( $C = 0.004$ ): 80.8333%

True Class	0	1	2	3	4	5	6	7	8	9
Predicted Class	0	278								
1		344								
2	2	12	238	4	6		8	7	21	4
3	2	9	12	247	1	4	3	2	19	7
4	1	7			144		5			105
5	3	28		41	1	142	10	1	13	13
6		11	4			10	275	1	4	
7	1	11	2		2			257	2	38
8	1	15	3	16			5	1	234	20
9	2	8	1	7	4	2		1	3	266

Underfit

Train accuracy ( $C = 0.1$ ): 99%

True Class	0	1	2	3	4	5	6	7	8	9
Predicted Class	0	318								
1		350			1					1
2			299			1			1	1
3				303			1		1	1
4	1				260					1
5	1					250				
6		1					304			
7			1		2			309		1
8		2		2		3			288	
9				3				1	1	289

Fit

Train accuracy ( $C = 1$ ): 100%

True Class	0	1	2	3	4	5	6	7	8	9
Predicted Class	0	319								
1		352								
2			302							
3				306						
4					262					
5						252				
6							305			
7								313		
8									295	
9										294

Overfit

What is the optimal value of  $C$ ?

good training performance  $\Rightarrow$  good test performance

Test accuracy ( $C = 0.004$ ): 79.9333%

True Class	0	1	2	3	4	5	6	7	8	9
Predicted Class	0	253								
1		331								
2	2	34	223	7	4		7	4	20	4
3		4	6	250		10	1	7	17	4
4	1	7	1		164		3		3	103
5	3	23	1	57	1	137	16	4	13	19
6	4	5	8		3	9	260	2		1
7	1	22	7	1				269	6	25
8	3	10		17		3	4	3	243	14
9	3	4	3	2	3			3	4	268

Underfit

Test accuracy ( $C = 0.1$ ): 91.5333%

True Class	0	1	2	3	4	5	6	7	8	9
Predicted Class	0	273								
1		332		1						
2	4	2	271	4		4	2	7	2	9
3	1		3	264			14	1	8	5
4	3	1			265			6		7
5	3		2	17	2	231	2	2	8	7
6	2		3		3	9	274	1		
7		3	9	1	2			303	1	12
8	1	2	2	5		11	1	3	268	4
9	1	4	1	4	5			9	1	265

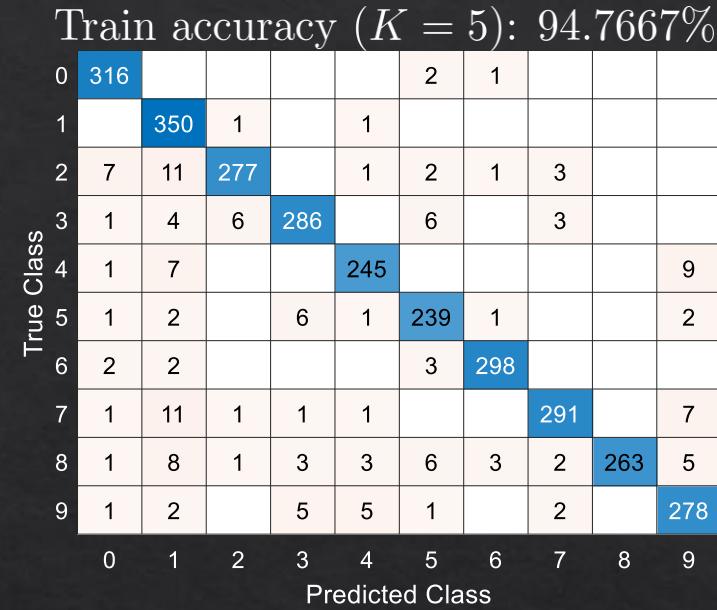
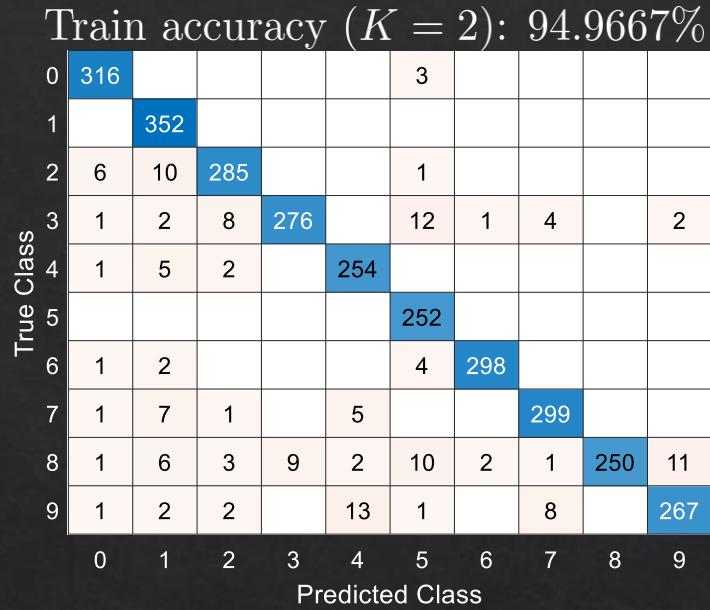
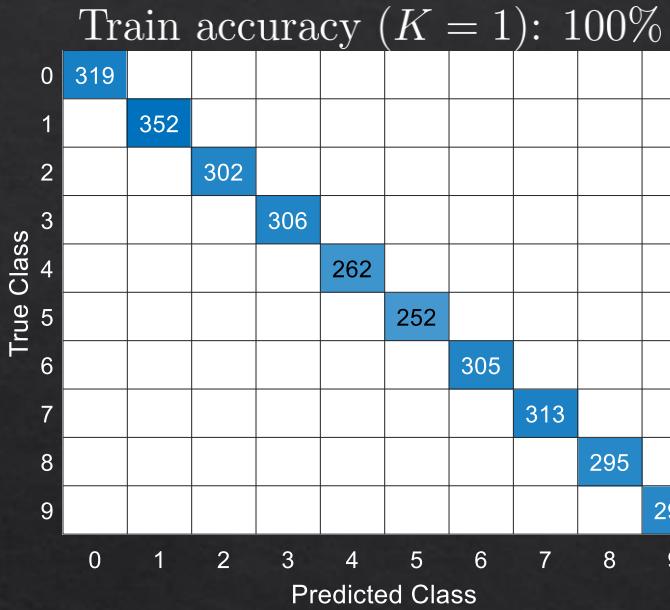
Fit

Test accuracy ( $C = 1$ ): 90.6333%

True Class	0	1	2	3	4	5	6	7	8	9
Predicted Class	0	270								
1		332		1						4
2	4	3	266	8	3	2	6	4	9	
3			3	266				16	2	7
4	3	1			260			6		12
5	3		2	18	2	225	2	2	12	8
6	2		3			5	9	272	1	
7	3	9	1	2				306		10
8	1	3	2	6		12	1	2	265	5
9	1	4	1	4	12			10	1	257

Overfit

# Confusion Matrix (K-NN and the hyperparameter K)



What is the optimal value of  $K$ ?

Test accuracy ( $K = 1$ ): 91.3%

True Class	0	1	2	3	4	5	6	7	8	9
0	282		2		1	2	2		1	1
1		338			1					
2	6	10	272	5	1		3	5	3	
3		1		274		12	1	5	4	2
4	1	7			238		4	1	2	29
5	2	1		17		237	7	1	6	3
6	3		1			1	286		1	
7		15	4	1	1			297		13
8	3	3	2	17	2	15	2	4	247	2
9		2		3	9		7	1		268

Test accuracy ( $K = 2$ ): 90.4333%

True Class	0	1	2	3	4	5	6	7	8	9
0	284		1		1	3	2			
1		339								
2	9	25	264	2		1		3	1	
3		1		258		26	3	5	1	5
4	2	11			257		2	1		9
5				6		258	3	1		1
6	7	1	1			3	280			
7		20	3		3			300		5
8	5	5	3	20	4	28	2	4	222	4
9	2	4	1	1	21			10		251

Test accuracy ( $K = 5$ ): 92.1%

True Class	0	1	2	3	4	5	6	7	8	9
0	285		1			1	4			
1		339								
2	6	24	265	2	1				5	2
3		3	1	279				9		4
4	1	9				248		3		21
5	5	5			10			248	3	1
6	4	1						2	285	
7	1	21	1	1	1				295	11
8	4	5			11	2	14	1	3	251
9	3	4			3	6		5	1	268

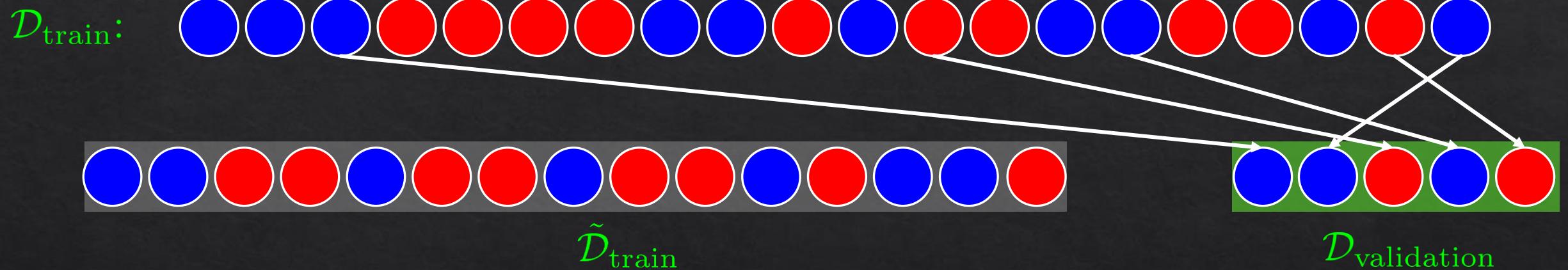
good training performance  $\Rightarrow$  good test performance

# Validation Set

- A good practice is to split the training set into:

- A (smaller) training set  $\tilde{\mathcal{D}}_{\text{train}}$ .
- A validation set  $\mathcal{D}_{\text{validation}}$ .

$$\tilde{\mathcal{D}}_{\text{train}} \sqcup \mathcal{D}_{\text{validation}} = \mathcal{D}_{\text{train}}$$

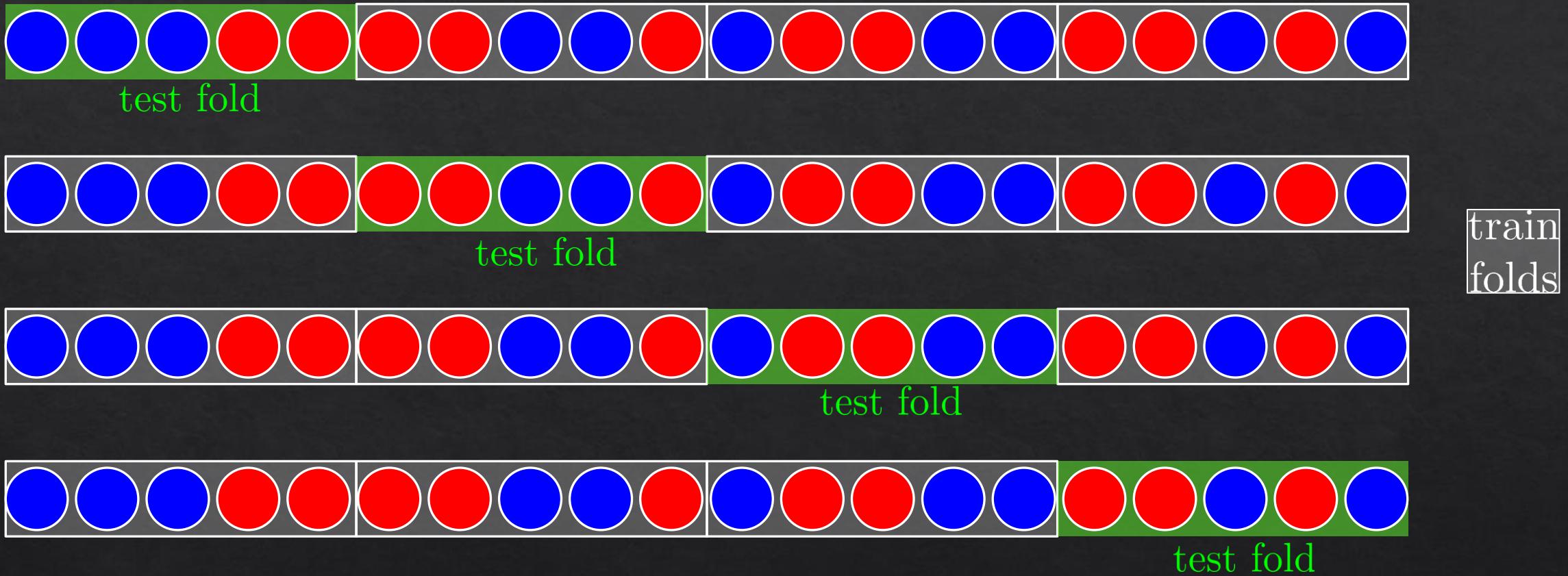


- Then, we check how different classifiers and different values of hyperparameters perform on the validation set.

- Make sure that  $\tilde{\mathcal{D}}_{\text{train}}$  and  $\mathcal{D}_{\text{validation}}$  are shuffled and balanced.

# Cross Validation (K Fold)

- We can also split the data into  $K$  folds:



- We then train the same model  $K$  times and compute the average performance of the  $K$  folds.
- The case  $K = N$  is known as “leave one out cross validation”.

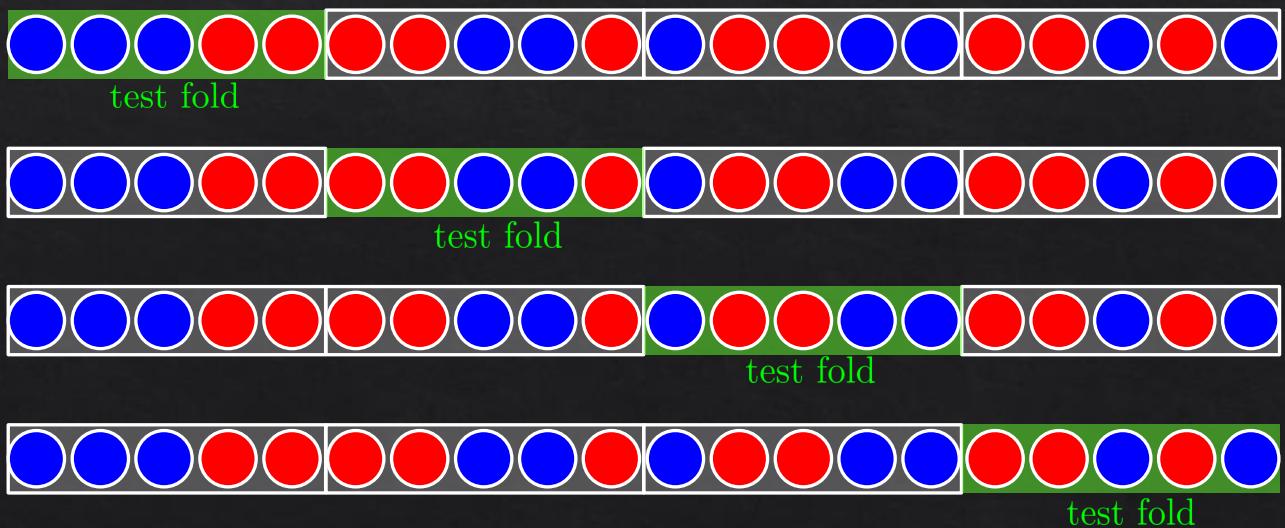
# Confusion Matrix and Cross-Validation – Example



## Confusion and Cross-validation

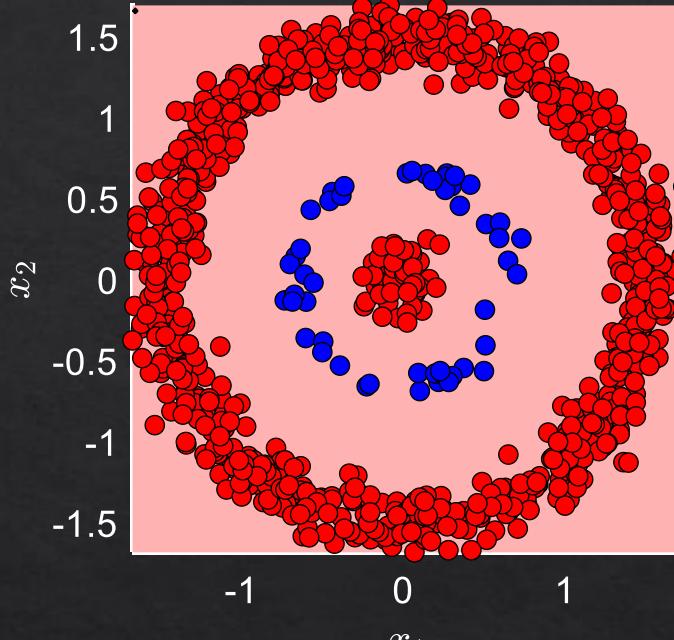
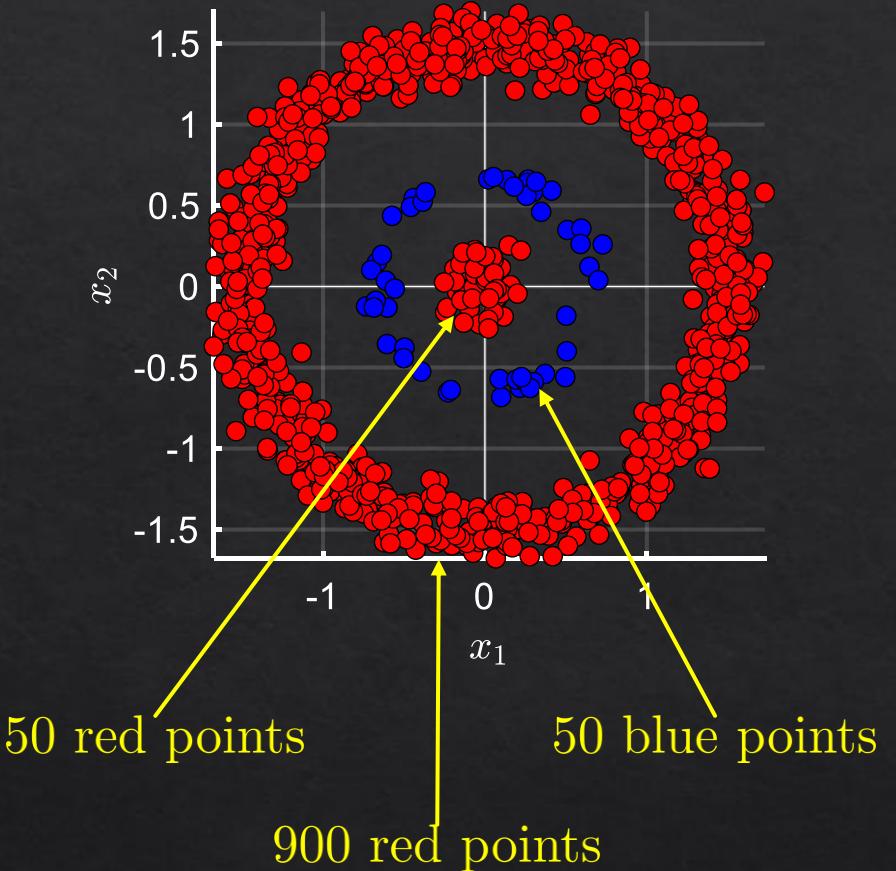
Test accuracy: 79.9333%

True Class	0	1	2	3	4	5	6	7	8	9	86.9%	13.1%
Predicted Class	0	1	2	3	4	5	6	7	8	9	97.6%	2.4%
0	253		5	2		14	9	1	6	1	73.1%	26.9%
1		331							8		83.6%	16.4%
2	2	34	223	7	4		7	4	20	4	58.2%	41.8%
3		4	6	250		10	1	7	17	4	50.0%	50.0%
4	1	7	1		164		3		3	103	89.0%	11.0%
5	3	23	1	57	1	137	16	4	13	19	81.3%	18.7%
6	4	5	8		3	9	260	2		1	81.8%	18.2%
7	1	22	7	1				269	6	25	92.4%	7.6%
8	3	10		17		3	4	3	243	14		
9	3	4	3	2	3			3	4	268		
	93.7%	75.2%	87.8%	74.4%	93.7%	79.2%	86.7%	91.8%	75.9%	61.0%		
	6.3%	24.8%	12.2%	25.6%	6.3%	20.8%	13.3%	8.2%	24.1%	39.0%		

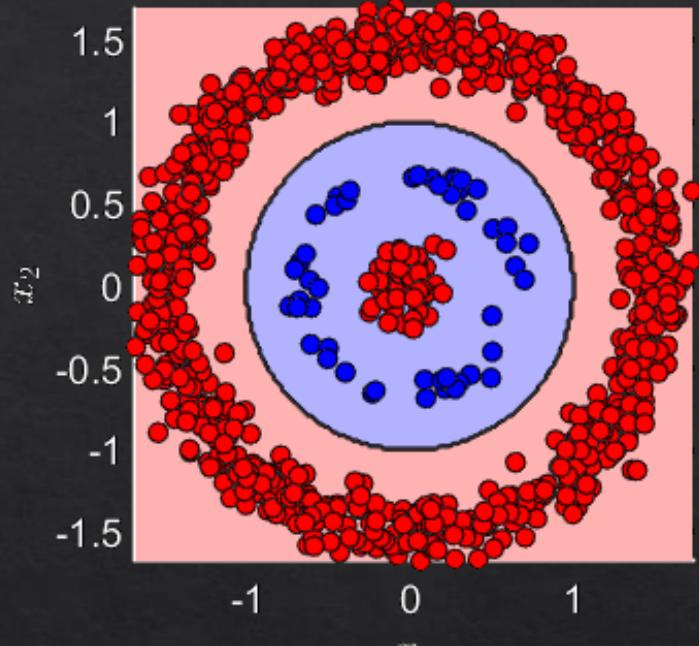


# Precision and Recall – Introduction

- Consider the following imbalanced data ( $N = 1,000$  points):



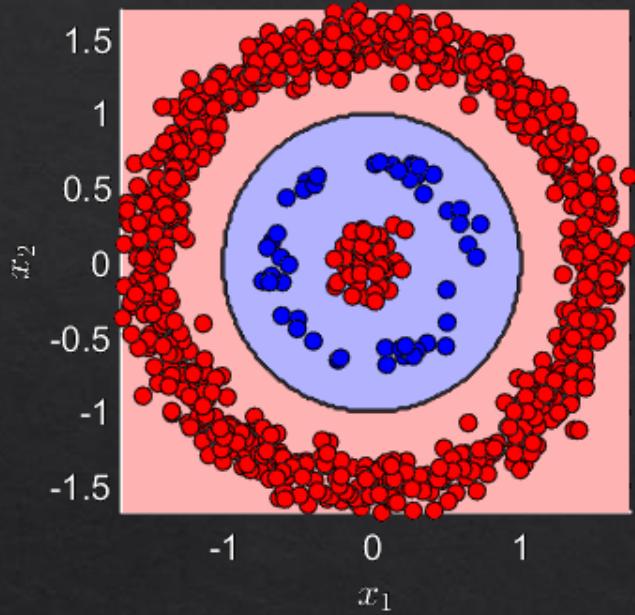
A trivial classifier: always red  
accuracy: 95%



Non trivial classifier  
accuracy: 95%

- For imbalanced data, accuracy might be misleading.
- Precision and recall might be more informative.

# Precision and Recall

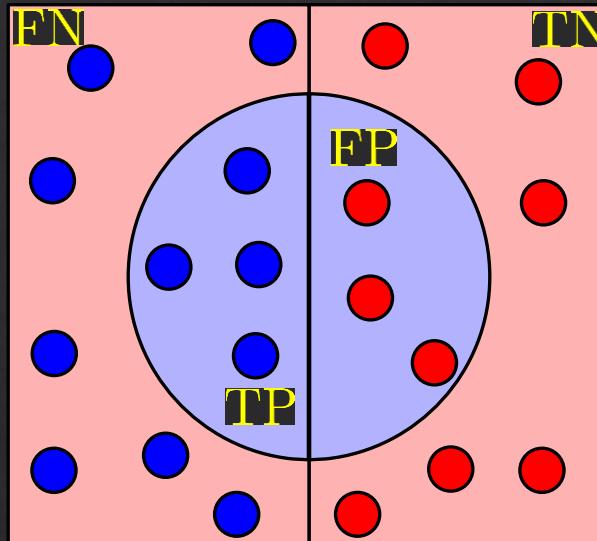


$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{50}{50 + 0} = 1$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{50}{50 + 50} = \frac{1}{2}$$

- The  $F_1$  measure, combines the two into a single number:

$$F_1 = 2 \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

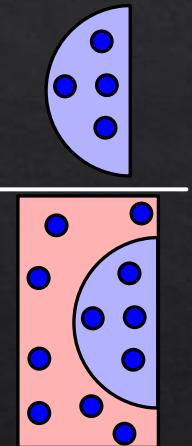


● blue positive  
● red negative

TP – True Positive  
FN – False Negative  
FP – False Positive  
TN – True Negative

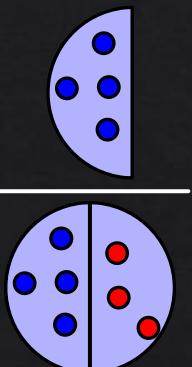
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

how many relevant items are selected



$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

how many selected items are relevant



# Receiver Operating Characteristic (ROC) – Introduction

- A linear classifier:

$$f(\mathbf{x}) = \text{sign}(\underbrace{\mathbf{w}^T \mathbf{x} - b}_{\text{distance from the decision boundary}}) = \begin{cases} 1 & \mathbf{w}^T \mathbf{x} - b \geq 0 \\ -1 & \mathbf{w}^T \mathbf{x} - b < 0 \end{cases}$$

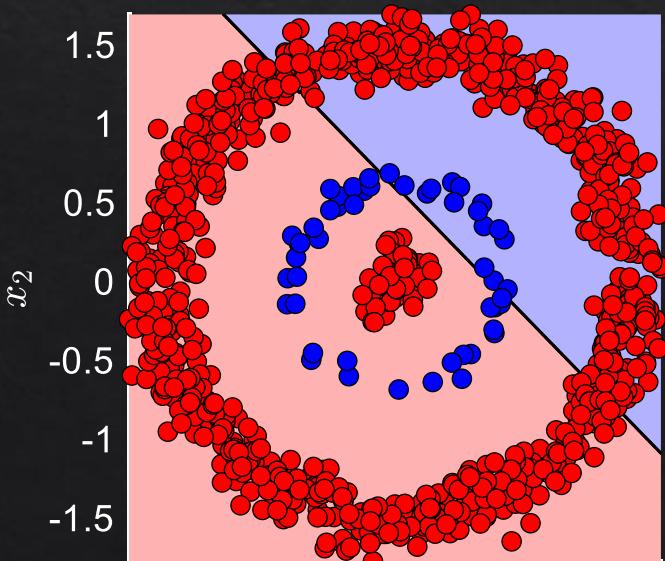
- We can control the sensitivity of the classifier by changing the threshold:

$$\tilde{f}(\mathbf{x}) = \begin{cases} 1 & \mathbf{w}^T \mathbf{x} - b \geq \alpha \\ -1 & \mathbf{w}^T \mathbf{x} - b < \alpha \end{cases}$$

$$\alpha \in (-\infty, \infty)$$

$$\alpha = \infty \implies \tilde{f}(\mathbf{x}) = -1$$

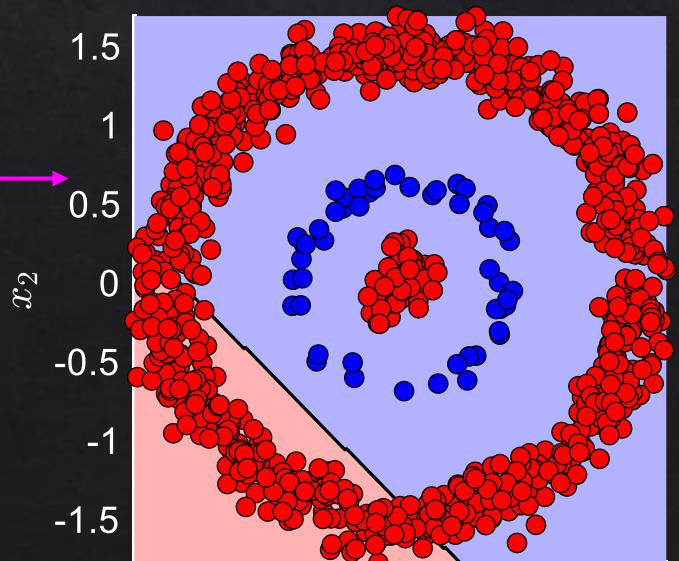
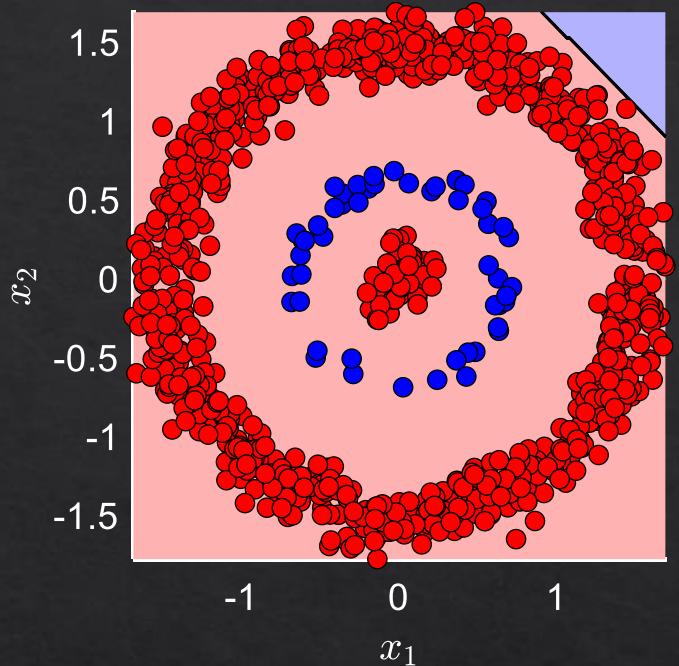
$$\alpha = -\infty \implies \tilde{f}(\mathbf{x}) = 1$$



$\leftarrow \tilde{f}(\mathbf{x}) \text{ with } \alpha = -2$

$\tilde{f}(\mathbf{x}) \text{ with } \alpha = -4 \rightarrow$

$\alpha$  controls the trade-off between  
false positive rate and true positive rate



# ROC Curve

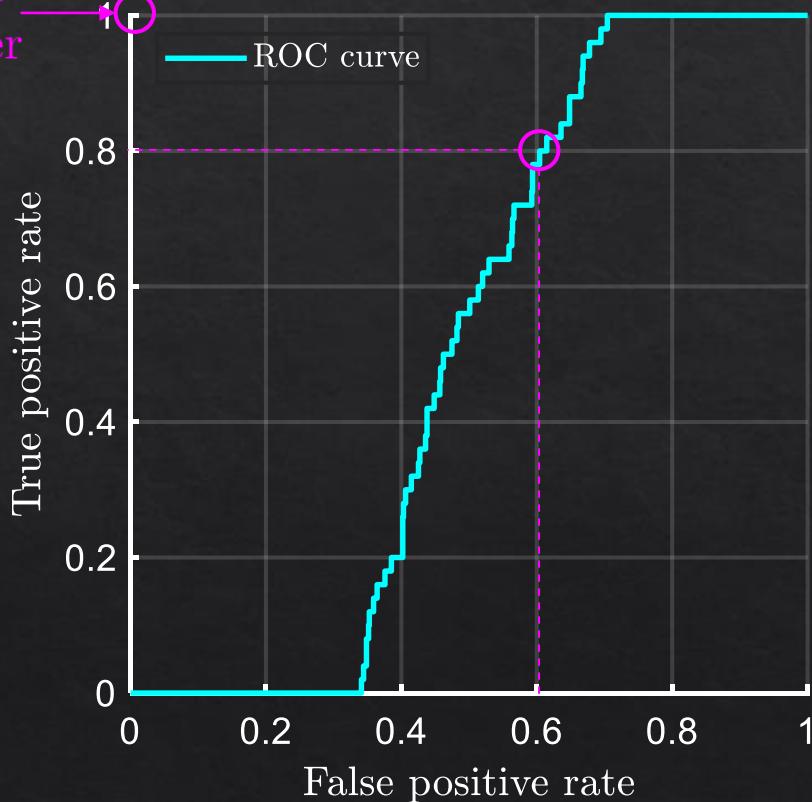
$$\tilde{f}(\mathbf{x}) = \begin{cases} 1 & \mathbf{w}^T \mathbf{x} - b \geq \alpha \\ -1 & \mathbf{w}^T \mathbf{x} - b < \alpha \end{cases}$$

$$\begin{aligned}\alpha &\in (-\infty, \infty) \\ \alpha = \infty &\implies \tilde{f}(\mathbf{x}) = -1 \\ \alpha = -\infty &\implies \tilde{f}(\mathbf{x}) = 1\end{aligned}$$

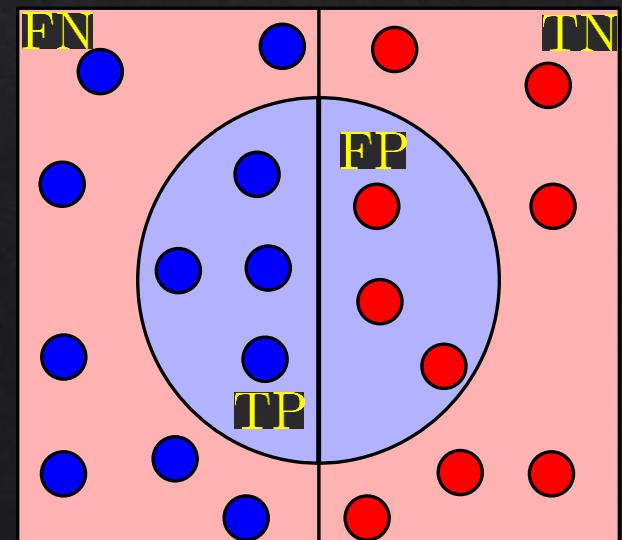
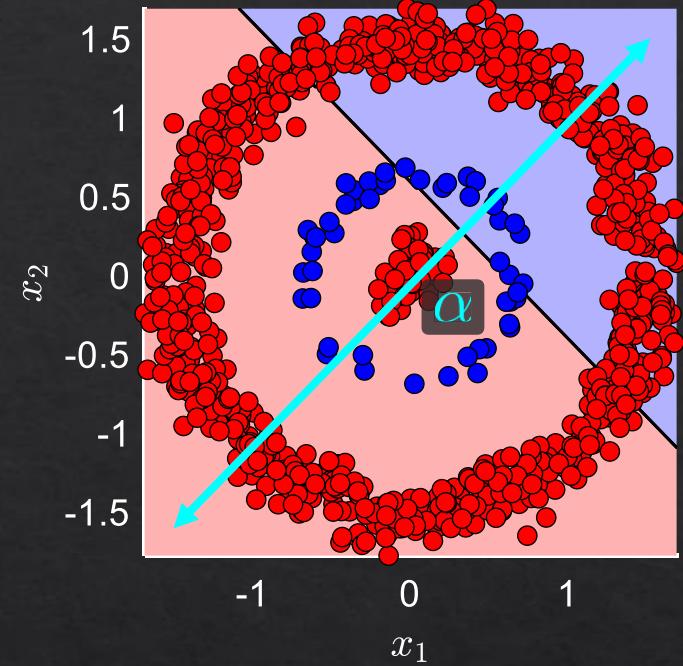
$$\text{TP rate} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \text{Recall}$$

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

perfect classifier

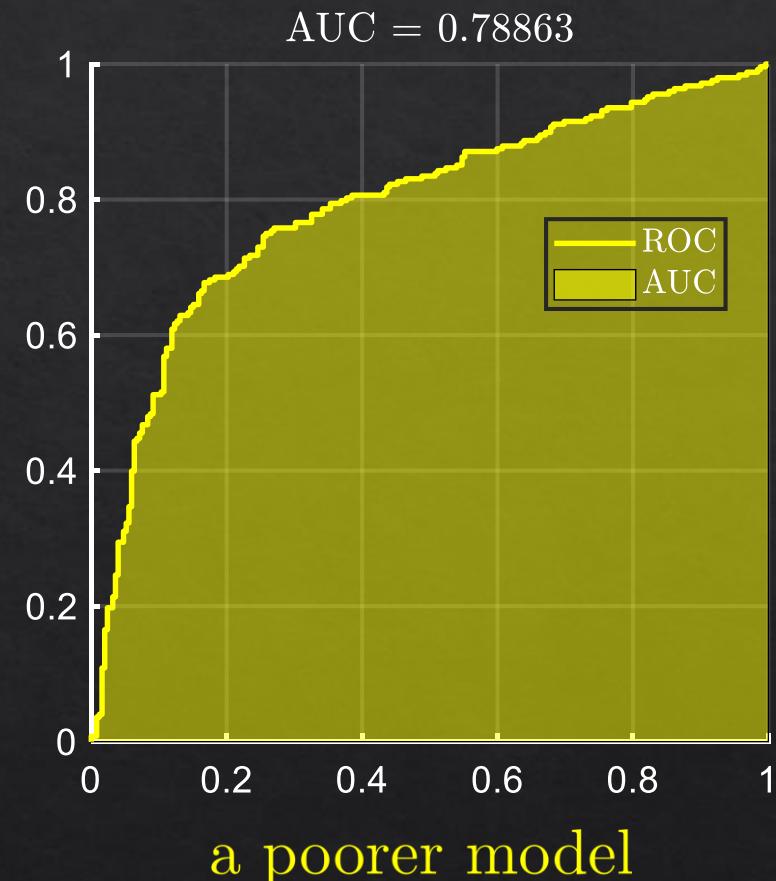
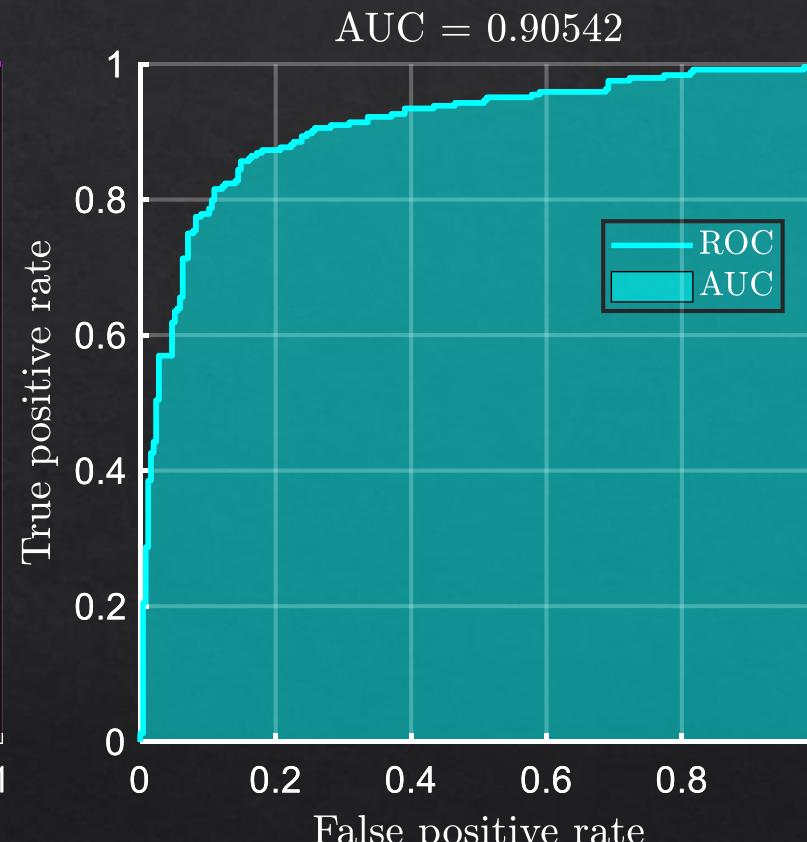
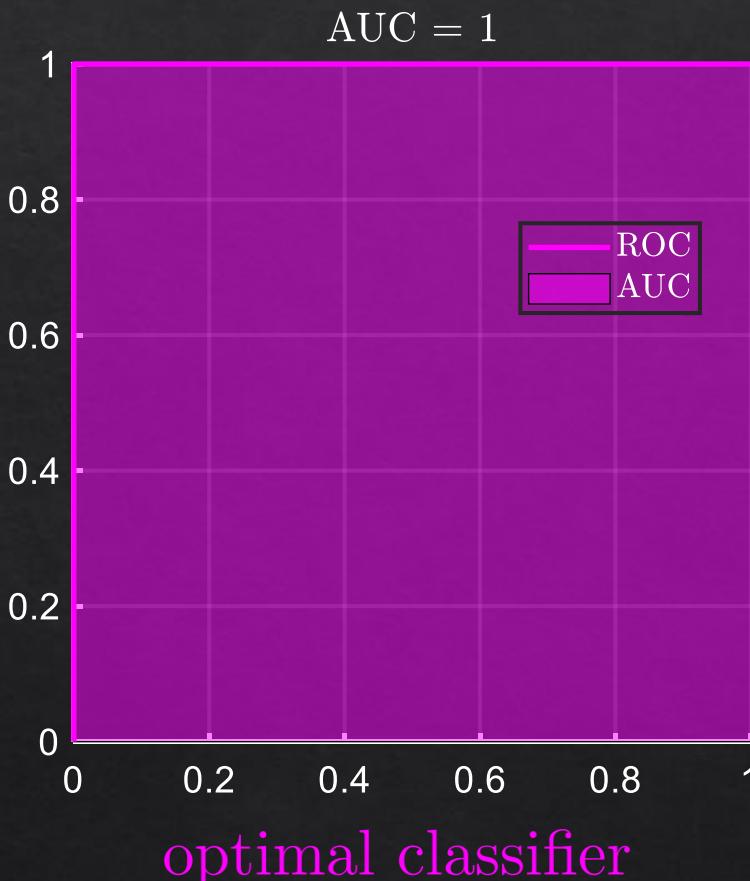


the curve is obtain by swiping over  $\alpha \in (-\infty, \infty)$



# Area Under the Curve (AUC)

- The AUC is the area under the ROC curve:



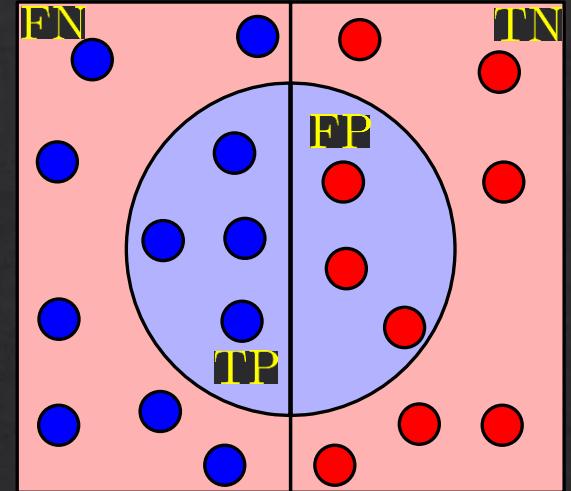
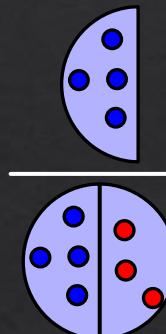
- We can use the AUC as a score for a model.

# Precision, Recall, ROC, and AUC – Example

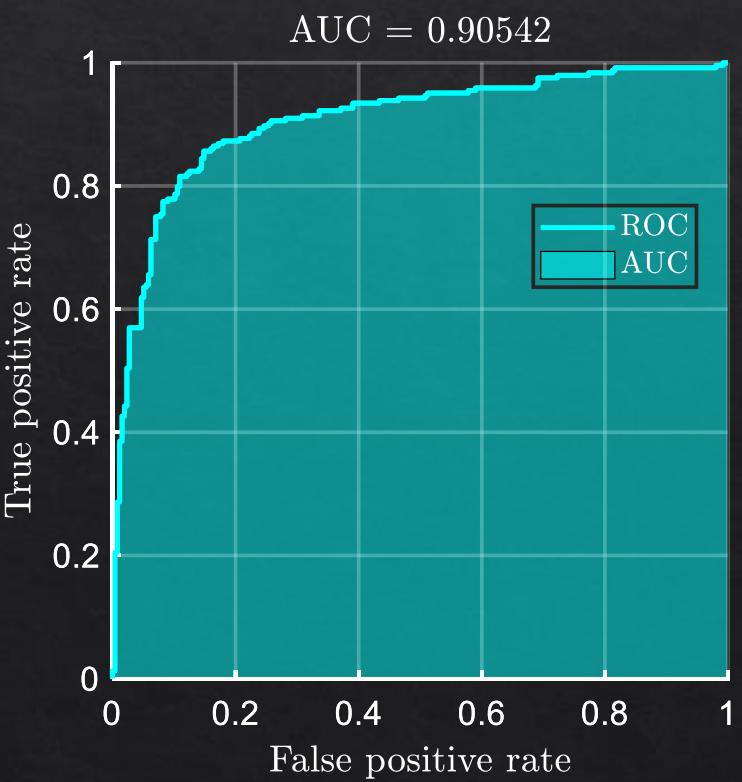
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$



$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$



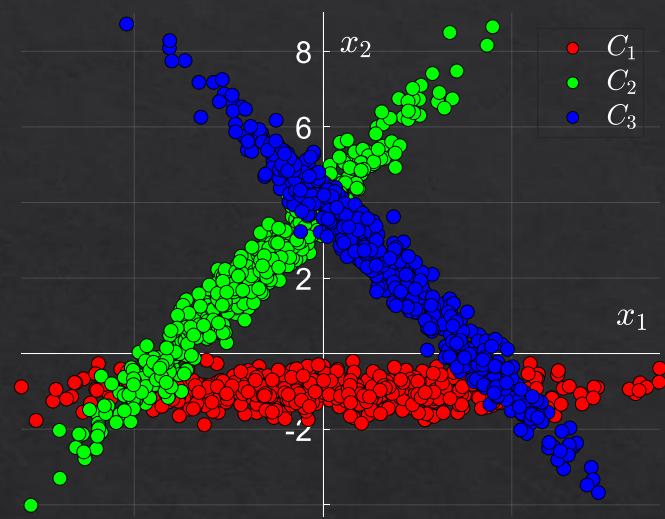
Precision, Recall, ROC, and AUC



# Weighted Loss (Cost)

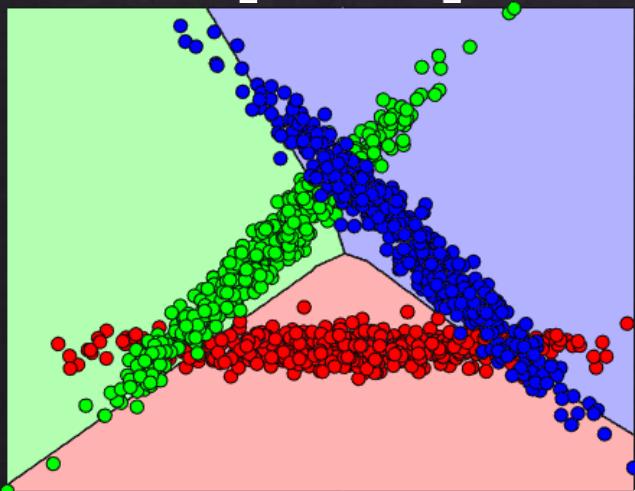
- (Reminder) Loss function:  $\ell(\hat{y}_i, y_i) \geq 0$

$$\bullet \text{ Matrix form: } L = \begin{bmatrix} 0 & \ell(\hat{C}_1, C_2) & \ell(\hat{C}_1, C_3) \\ \ell(\hat{C}_2, C_1) & 0 & \ell(\hat{C}_2, C_3) \\ \ell(\hat{C}_3, C_1) & \ell(\hat{C}_3, C_2) & 0 \end{bmatrix}$$
$$L[i, j] = \ell(\hat{C}_i, C_j)$$



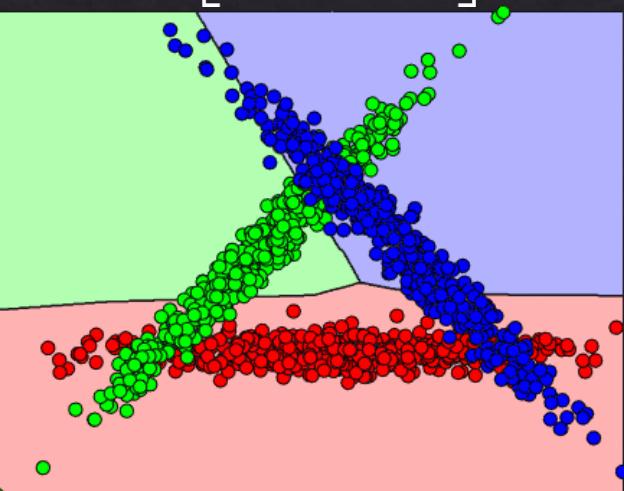
Hamming 0 – 1 loss:

$$L = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$



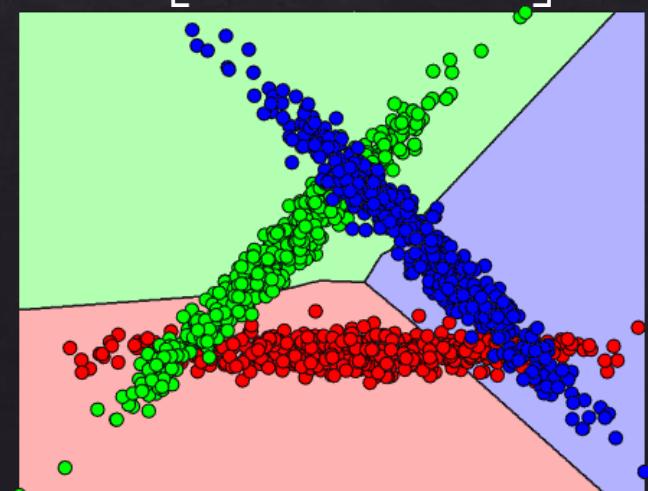
high cost for mislabeling  $C_1$ :

$$L = \begin{bmatrix} 0 & 1 & 1 \\ 100 & 0 & 1 \\ 100 & 1 & 0 \end{bmatrix}$$

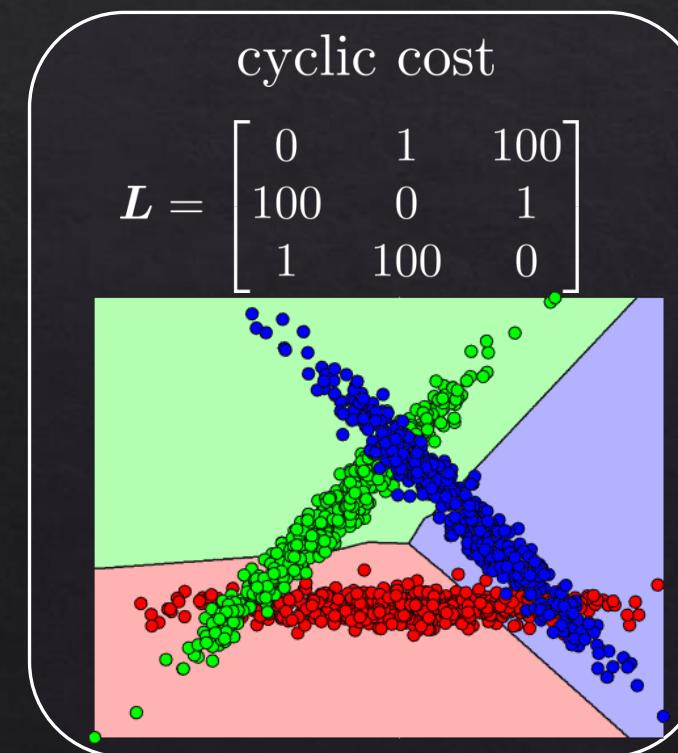
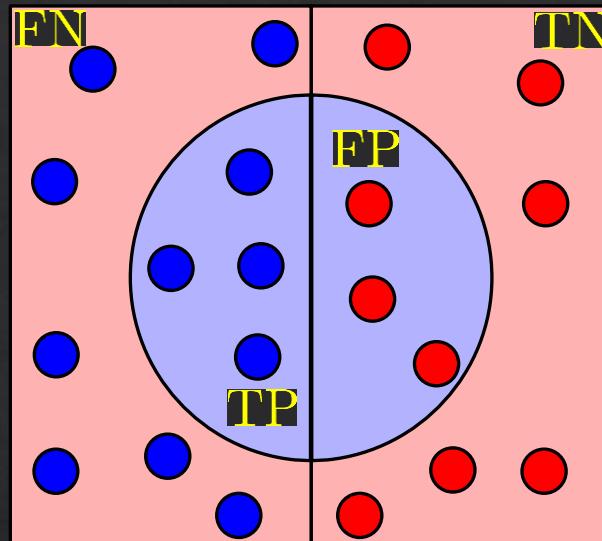
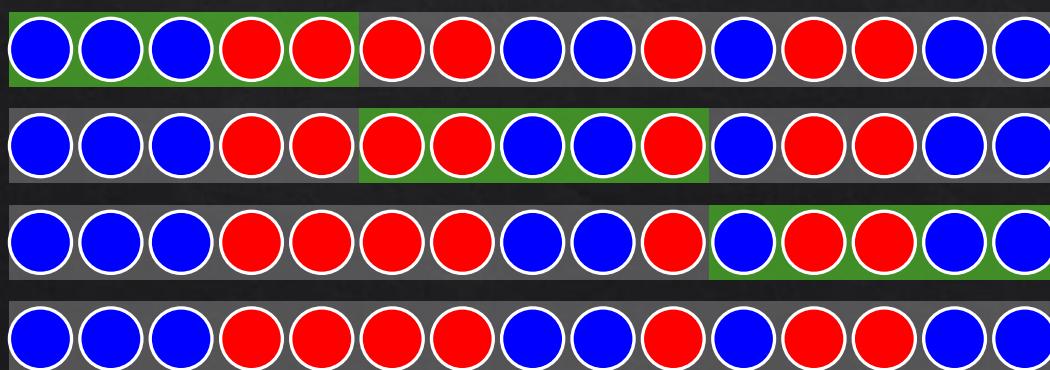
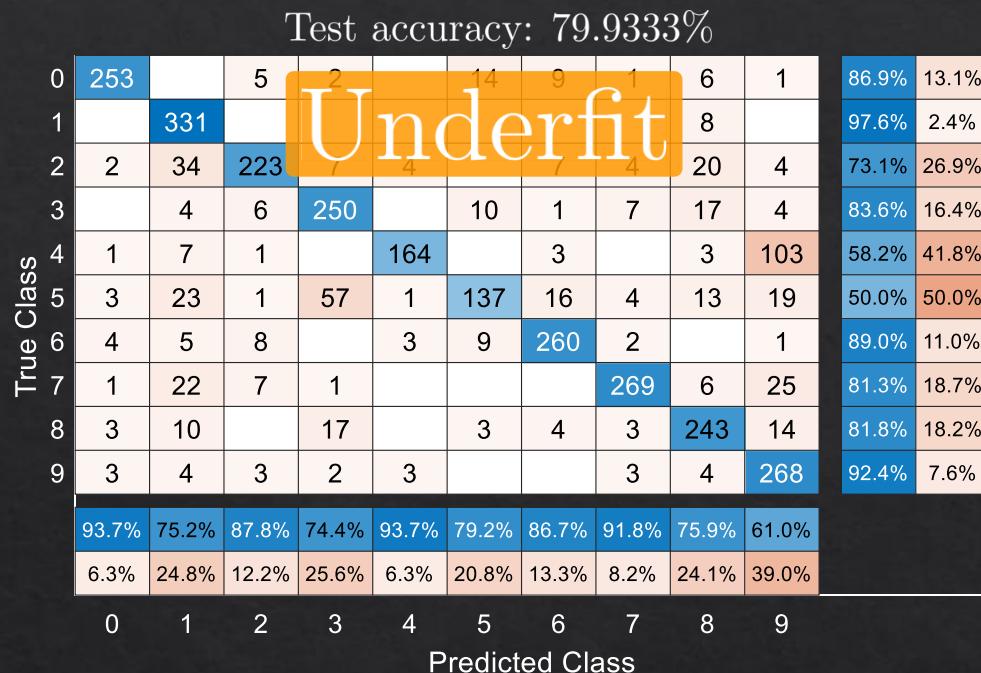


cyclic cost

$$L = \begin{bmatrix} 0 & 1 & 100 \\ 100 & 0 & 1 \\ 1 & 100 & 0 \end{bmatrix}$$



# Questions



fixel your pixels