EECE 5644: Linear Classification & Linear Discriminant Analysis (LDA)

Mark Zolotas

E-mail: m.zolotas@northeastern.edu
Webpage: https://coe.northeastern.edu/people/zolotas-mark/

Tentative Course Outline (Wks. 1-2)

Topics	Dates	Assignments	Additional Reading
~~Course Overview~~ ~~Machine Learning Basics~~	~~07/05~~	**Optional Homework 0** released on Canvas on 07/08 but please do NOT submit on Canvas	~~Chpt. 1~~ ~~Murphy 2012~~
~~Foundations: Linear Algebra, Probability, Numerical Optimization (Gradient Descent), Regression~~	~~07/06-12~~		~~Stanford LA Review~~ ~~Stanford Prob. Review~~ ~~Chpt. 8 Murphy 2022~~
~~*Quick Python Tutorial*~~	~~07/12~~	**Homework 1** released on Canvas on 07/15 **Due 07/25**	~~N/A~~
Linear Classifier Design, Linear Discriminant Analysis and Principal Component Analysis (PCA)	07/13-15		Chpts. 9.2.6 & 20.1 Murphy 2022
Bayesian Decision Theory: Empirical Risk Min, Max Likelihood (ML), Max a Posteriori	07/14		Chpt. 2 Duda & Hart 2001 Deniz Erdogmus Notes

Least Squares Regression GD Recap

1. Initialize $\boldsymbol{\theta}^{(0)}$

2. Repeat until convergence:

Average scaling for MSE, 2 cancels out in derivative

Vector derivative

Batch GD update rule

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \nabla \boxed{\frac{1}{2N}} \mathcal{L}(\boldsymbol{\theta}^{(t)})$$

$$= \boldsymbol{\theta}^{(t)} - \alpha \left(\frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{\theta}^{\intercal (t)} \tilde{\mathbf{x}}^{(i)} - y^{(i)}) \tilde{\mathbf{x}}^{(i)} \right)$$

$$= \boldsymbol{\theta}^{(t)} - \alpha \left(\frac{1}{N} \mathbf{X}^{\intercal} (\mathbf{X} \boldsymbol{\theta}^{(t)} - \mathbf{y}) \right)$$

Equivalent matrix derivative

3. When $\mathbf{g}(\boldsymbol{\theta}) = \mathbf{0}$, then we have derived $\boldsymbol{\theta}^{*}$ using GD

Linear Classifiers

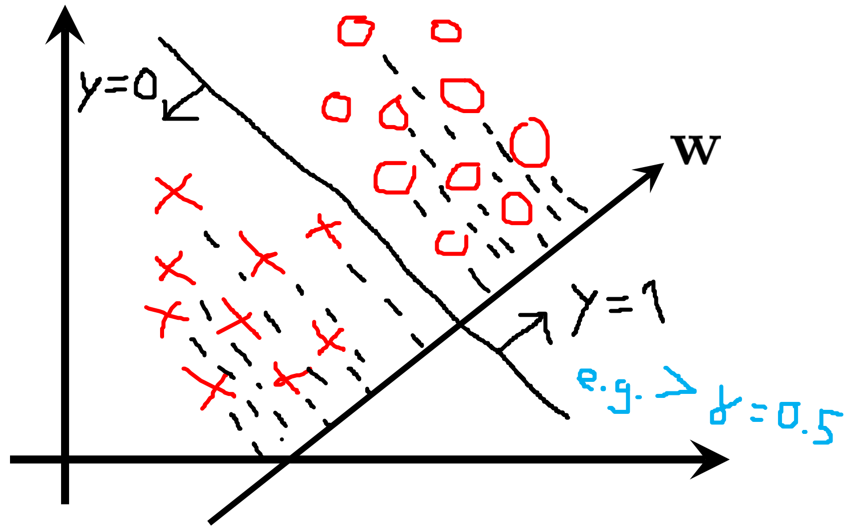
Let $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, N training samples
Inputs $\mathbf{x} \in \mathbb{R}^n$, discrete valued labels $y \in \{1, \ldots, C\}$

- Find a linear combo. of $\mathbf{x}$ to separate classes
- A **linear classifier** is typically of the form:

$$y = f(\mathbf{x}; \boldsymbol{\theta}) = f(\mathbf{w}^\intercal \mathbf{x})$$

- Project inputs onto a line with direction $\mathbf{w}$
- Decision rule:

$$f(\mathbf{x}; \boldsymbol{\theta}) = \begin{cases} 1 & \text{if } \mathbf{w}^\intercal \mathbf{x} > \gamma \\ 0 & \text{otherwise} \end{cases}$$

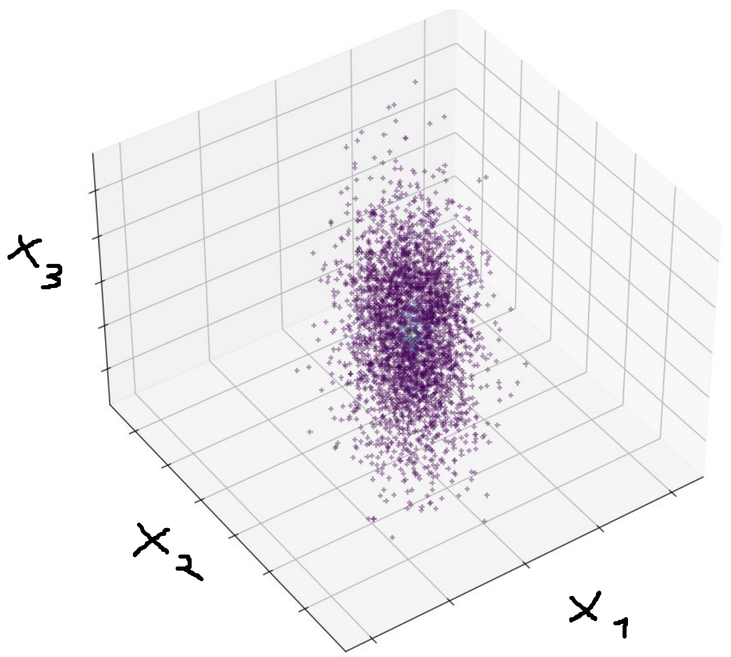


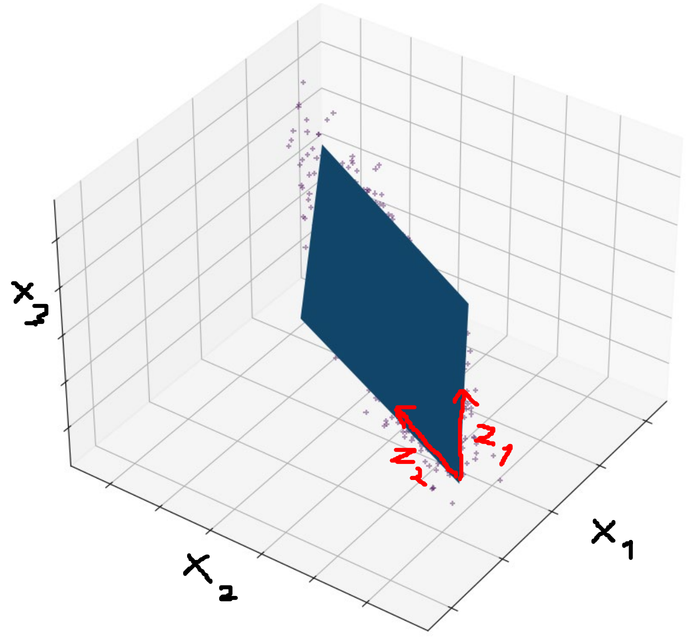
Hyperplane
in n-dims.

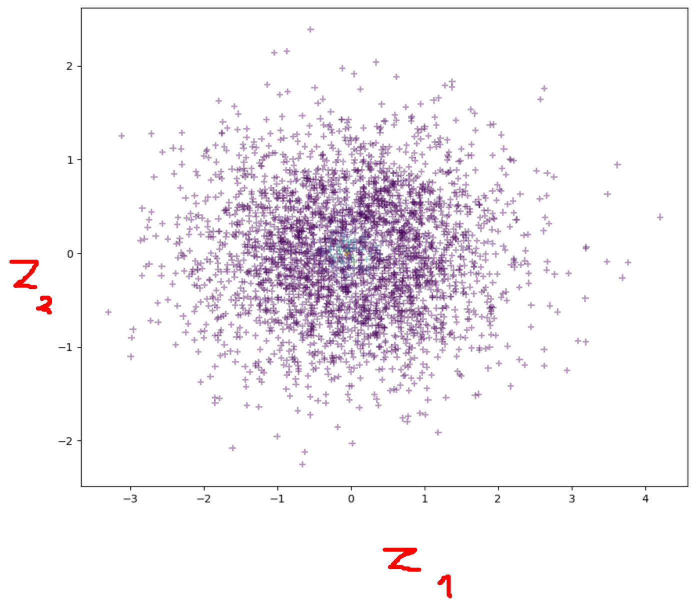
Dimensionality Reduction

$\mathbf{x} \in \mathbb{R}^n \rightarrow \mathbf{z} \in \mathbb{R}^k$ where $k < n$

$x \in \mathbb{R}^3$ $z \in \mathbb{R}^2$





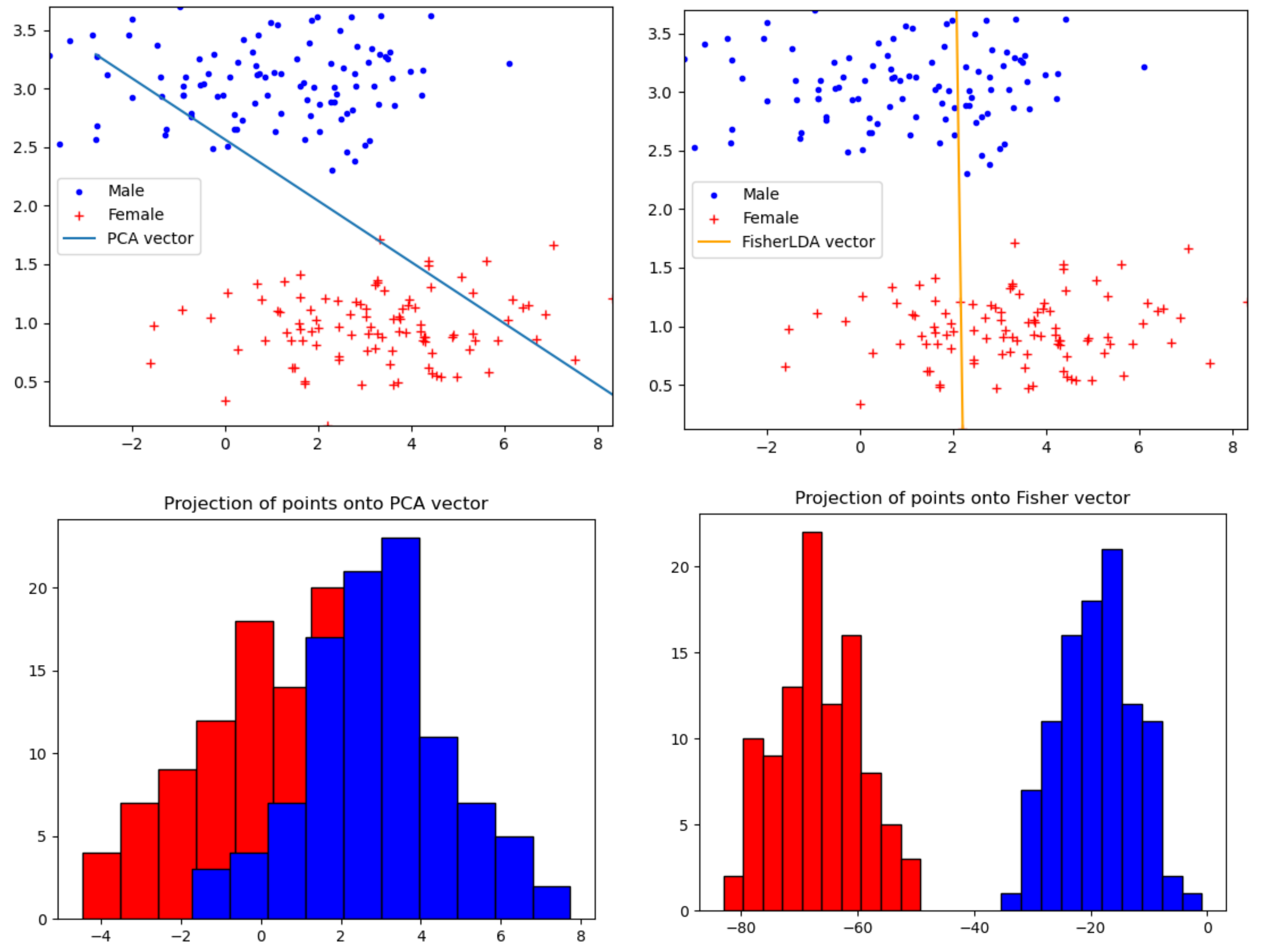


- Consider a linear classifier that also performs **dimensionality reduction**
  - ❖ Project n-dimensional input vector down to a k-dimensional space ($k \ll n$)
  - ❖ Create decision boundary for classification in k-space (e.g. line separation)

Data Representation vs Data Classification

- **Data Representation:** Project data to lower dimensional space that *most accurately represents* the original data, e.g., PCA projects in directions of maximum variance

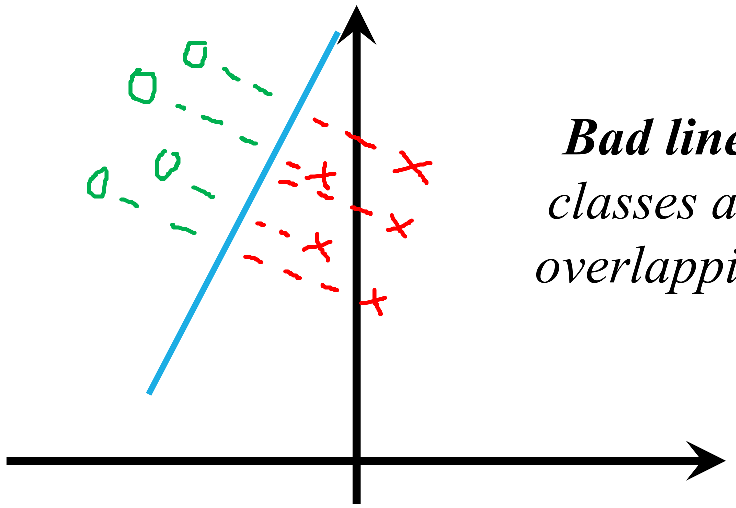
- **Data Classification:** Project data to a low dimensional space that preserves structure useful for classification, e.g., Fisher's LDA!



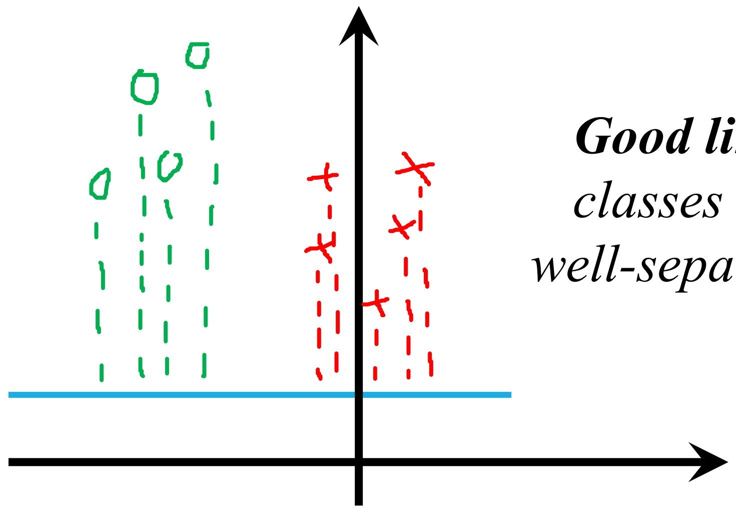
Kevin Murphy, *"Probabilistic Machine Learning: An Introduction"*, 2022

Fisher's LDA – Key Idea

- Find projection to a line that **maximizes separability** between classes



Bad line:
classes are overlapping



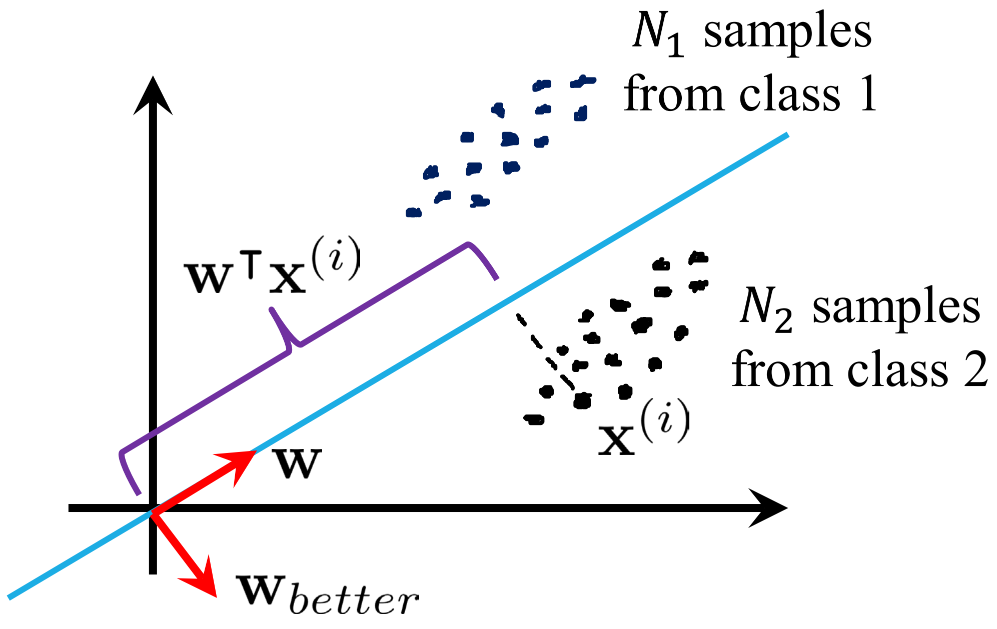
Good line:
classes are well-separated

- Note Fisher's LDA is a <u>specific</u> "discriminant", often used interchangeably

Fisher’s LDA – Two-class Setting

- Suppose we have **two** classes (**binary** classification) and samples $\mathbf{x} \in \mathbb{R}^n$
- Assume N_1 and N_2 samples drawn from classes 1 and 2, respectively
- Consider projection of arbitrary sample $\mathbf{x}^{(i)}$ onto line with direction $\mathbf{w}$

$\mathbf{w}^\top \mathbf{x}^{(i)}$ is the scalar distance from origin of $\mathbf{x}^{(i)}$ projected onto a one-dimensional line



N_1 samples from class 1
$\mathbf{w}^\top \mathbf{x}^{(i)}$
N_2 samples from class 2
$\mathbf{x}^{(i)}$
$\mathbf{w}$
$\mathbf{w}_{better}$

Projections of samples from different classes should have minimal overlap

Fisher's LDA – Maximum Separability

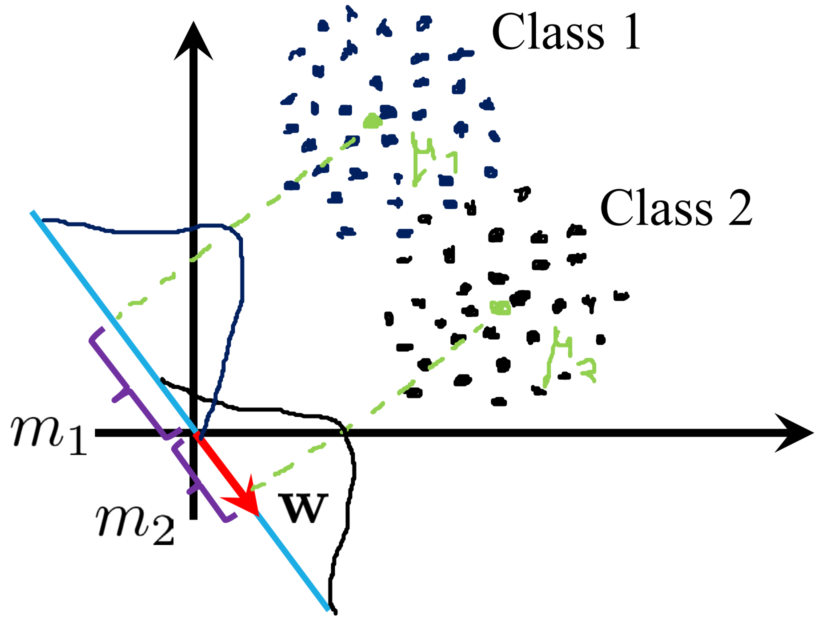
- How do we establish **maximum separability** between the classes?
- Consider class-specific mean vectors: $\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i:y^{(i)}=k} \mathbf{x}^{(i)}$ for $k = 1, 2$
- Define mean projections: $m_k = \mathbf{w}^\intercal \boldsymbol{\mu}_k$

Could define separability as the **distance between projected means:**

$$(m_2 - m_1)^2$$

Maximize^

But what about their **spread…**?



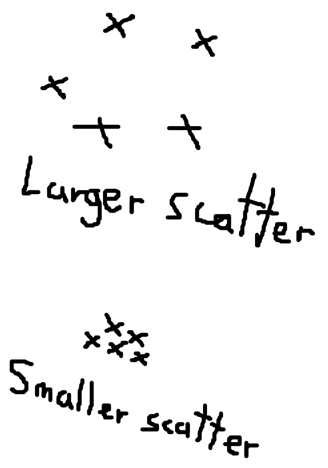
Fisher's LDA – Objective Function

- Also want to ensure that the clusters are **"tight"** to reduce overlap
- For each projected point $z^{(i)} = \mathbf{w}^\intercal \mathbf{x}^{(i)}$ class **variance** is proportional to:

$$s_k^2 = \sum_{i:y^{(i)}=k} (z^{(i)} - m_k)^2 = \sum_{i:y^{(i)}=k} (\mathbf{w}^\intercal \mathbf{x}^{(i)} - \mathbf{w}^\intercal \boldsymbol{\mu}_k)^2$$

$$= \sum_{i:y^{(i)}=k} \mathbf{w}^\intercal (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^\intercal \mathbf{w} = \mathbf{w}^\intercal \mathbf{S}_k \mathbf{w}$$

Scatter *matrix*



- **Objective Function:**

Separate maximally

$$\mathcal{L}(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

While simultaneously minimizing variance

Fisher's LDA – Fisher's Criterion

- Express in terms of **w**:

$$\mathcal{L}(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{(\mathbf{w}^\intercal \boldsymbol{\mu}_2 - \mathbf{w}^\intercal \boldsymbol{\mu}_1)^2}{\mathbf{w}^\intercal \mathbf{S}_1 \mathbf{w} + \mathbf{w}^\intercal \mathbf{S}_2 \mathbf{w}}$$

$$= \frac{\mathbf{w}^\intercal (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^\intercal \mathbf{w}}{\mathbf{w}^\intercal (\mathbf{S}_1 + \mathbf{S}_2) \mathbf{w}} = \frac{\mathbf{w}^\intercal \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\intercal \mathbf{S}_W \mathbf{w}}$$

Between-class *scatter matrix*

Within *scatter matrix*

- Maximization problem on ratio of two scalars:

$$\mathbf{w}^* = \max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \max_{\mathbf{w}} \boxed{\frac{\mathbf{w}^\intercal \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\intercal \mathbf{S}_W \mathbf{w}}}$$

Fisher's Criterion

Fisher's LDA – Generalized Eigenvalue Problem

- Require **derivative** of objective:

$$\frac{\delta \mathcal{L}}{\delta \mathbf{w}} = \frac{2\mathbf{S}_B\mathbf{w}(\mathbf{w}^\intercal\mathbf{S}_W\mathbf{w}) - 2\mathbf{S}_W\mathbf{w}(\mathbf{w}^\intercal\mathbf{S}_B\mathbf{w})}{(\mathbf{w}^\intercal\mathbf{S}_W\mathbf{w})^2} = 0$$

- Can rearrange to obtain:

$$\mathbf{S}_B\mathbf{w} = \left(\frac{\mathbf{w}^\intercal\mathbf{S}_B\mathbf{w}}{\mathbf{w}^\intercal\mathbf{S}_W\mathbf{w}}\right)\mathbf{S}_W\mathbf{w}$$

$$\mathbf{S}_B\mathbf{w} = \lambda\mathbf{S}_W\mathbf{w}$$ ***Generalized Eigenvalue Problem***

- If $\mathbf{S}_W^{-1}$ exists: $$\mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{w} = \lambda\mathbf{w}$$ ***Regular Eigenvalue Problem***

Optimal $\mathbf{w}^*$ is eigenvector of $S_W^{-1}S_B$ with **largest eigenvalue**

Coding Break

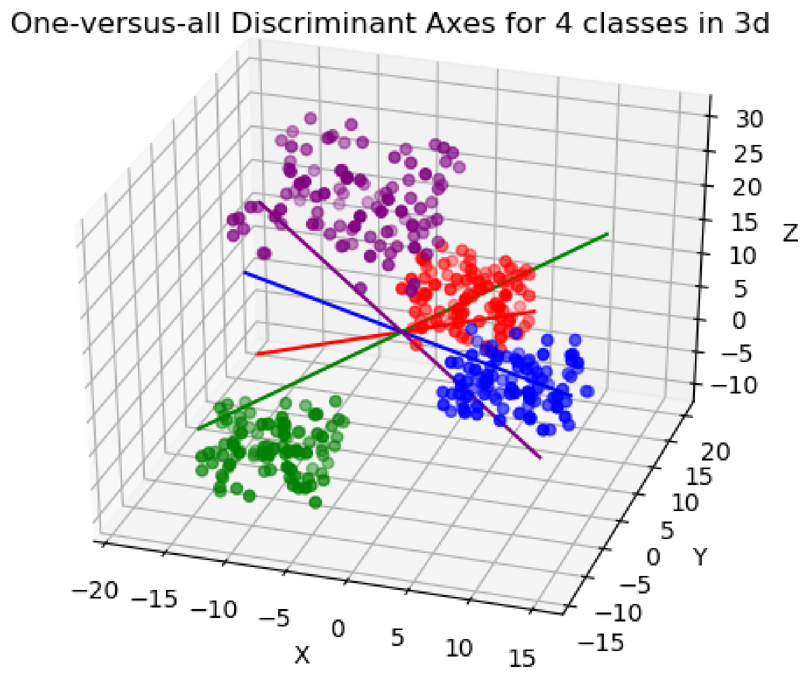
Fisher's LDA – Multiple Class Setting

- Can extend Fisher's LDA to case where $C > 2$
- Instead derive a projection matrix $\mathbf{W}$ that consists of $C - 1$ projections $\mathbf{z}$

$$\mathbf{z}^{(i)} = \mathbf{W}\mathbf{x}^{(i)}$$

- This matrix maps from $n \rightarrow k$ dimensions, assuming at most $k \leq (C - 1)$
- **"One vs Rest" (OvR) classification:** train C binary classifiers and pick best as label

https://en.wikipedia.org/wiki/Linear_discriminant_analysis#Fisher.27s_linear_discriminant



Fisher's LDA – Classification

Let $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, N training samples
Inputs $\mathbf{x} \in \mathbb{R}^n$, binary valued labels $y \in \{0, 1\}$

- Given Fisher's solution: $\mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{w}_{\text{LDA}} = \lambda\mathbf{w}_{\text{LDA}}$
- Remember linear classification setting: $\hat{y} = f(\mathbf{x}; \boldsymbol{\theta}) = \begin{cases} 1 & \text{if } \mathbf{w}^\intercal\mathbf{x} > \gamma \\ 0 & \text{otherwise} \end{cases}$
- Fisher's LDA projection:

$$\underset{\in \mathbb{R}}{\text{LDA score}} \longrightarrow \mathbf{w}_{\text{LDA}}^\intercal\mathbf{x} \quad \begin{matrix} \hat{y} = 1 \\ > \\ < \\ \hat{y} = 0 \end{matrix} \quad \gamma$$

How do we set a threshold γ for class 0 or 1?

Classifier Evaluation – Confusion Matrix

- **Confusion matrix** to summarize performance of classifier
- Allows confusion between classes to be easily identified
- Most performance measures can be computed from this confusion matrix
- 2×2 matrix for binary classification:

TP: True Positive $p(\hat{y} = 1|y = 1)$

FP: False Positive $p(\hat{y} = 1|y = 0)$

TN: True Negative $p(\hat{y} = 0|y = 0)$

FN: False Negative $p(\hat{y} = 0|y = 1)$

Predicted Class

Actual Class		**1**	**0**
	1	TP	FN
	0	FP	TN

Classifier Evaluation – Accuracy/Error

- **Classification Rate/Accuracy:** # correctly classified divided by # examples

$$\text{accuracy} = \frac{\#\text{correct predictions}}{\#\text{total predictions}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- **Probability of Error:** $\Pr(\text{error}) = 1 - \Pr(\text{correct})$

TP: True Positive $p(\hat{y} = 1 | y = 1)$

FP: False Positive $p(\hat{y} = 1 | y = 0)$

TN: True Negative $p(\hat{y} = 0 | y = 0)$

FN: False Negative $p(\hat{y} = 0 | y = 1)$

Predicted Class

Actual Class		**1**	**0**
	1	TP	FN
	0	FP	TN

Classifier Evaluation – Recall

- **Recall:** # correctly classified positive divided by # total actual positive
- Fraction of positives detected; Pr(correct | positive example)
- **High Recall:** Positive class is correctly recognized (small FN)
- Coverage of actual positive samples (**quantity**)

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

a.k.a. **True Positive Rate/Hit Rate/Sensitivity**

		Predicted Class	
		1	**0**
Actual Class	**1**	TP	FN
	0	FP	TN

Classifier Evaluation – Precision

- **Precision:** # correctly classified positive divided by # total predicted positive
- Fraction of detections are positive; Pr(positive | classified positive)
- **High Precision:** Positive predictions are indeed positive (small FP)
- How accurate are the positive predictions (**quality**)

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

a.k.a. **Positive Predictive Value**

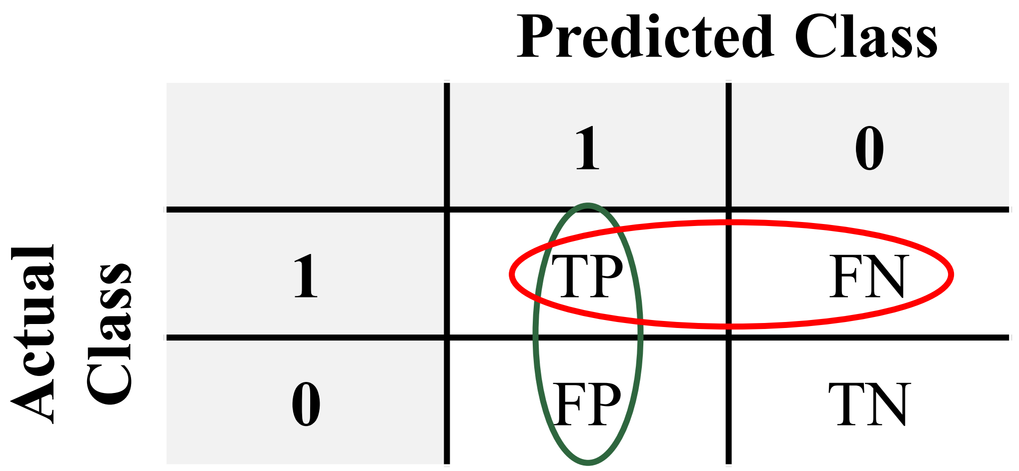
		Predicted Class	
		1	**0**
Actual Class	**1**	TP	FN
	0	FP	TN

Classifier Evaluation – Recall/Precision

- **High recall, low precision:** Most positive examples are correctly recognized (low FN) but there are a lot of false positives (high FP)
- **Low recall, high precision:** Miss a lot of positive examples (high FN) but those predicted as positive are indeed positive (low FP)

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

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



		Predicted Class	
		1	**0**
Actual Class	**1**	TP	FN
	0	FP	TN

Classifier Evaluation – F1-Score

- Combine precision and recall into a **single metric**
- **F1-score** is the harmonic mean: $$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$
- Weights precision and recall equally

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

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

Predicted Class

Actual Class		**1**	**0**
	1	TP	FN
	0	FP	TN

Classifier Evaluation – Multiple Classes

- Also useful for **multiple classes**; treat one class as positive and rest negative
- Compute performance measures in same way, e.g., **classification accuracy** is still # correctly classified (**trace** of matrix) divided by # examples
- Recall/Precision/F-1 Score metrics all still computed per class, e.g., for class 1:

Predicted Class

Actual Class		**1**	**2**	**3**
	1	TP	FN	FN
	2	FP	TN	?
	3	FP	?	TN

Classifier Evaluation – Class Imbalance

		Predicted Class	
		1	0
Actual Class	1		
	0		

- What if dataset consists of severely **imbalanced** classes? E.g. 99% of N = 100 people are healthy ($y = 0$) and 1% have a disease ($y = 1$)?

- Use a classifier that always predicts "healthy" ($\hat{y} = 0$):

$$\text{accuracy} = \qquad \text{recall} = \qquad \text{precision} =$$

- Use a classifier that always predicts "disease" ($\hat{y} = 1$):

$$\text{accuracy} = \qquad \text{recall} = \qquad \text{precision} =$$

- **Challenge:** metrics affected severely by majority class

Classifier Evaluation – ROC Curves (1)

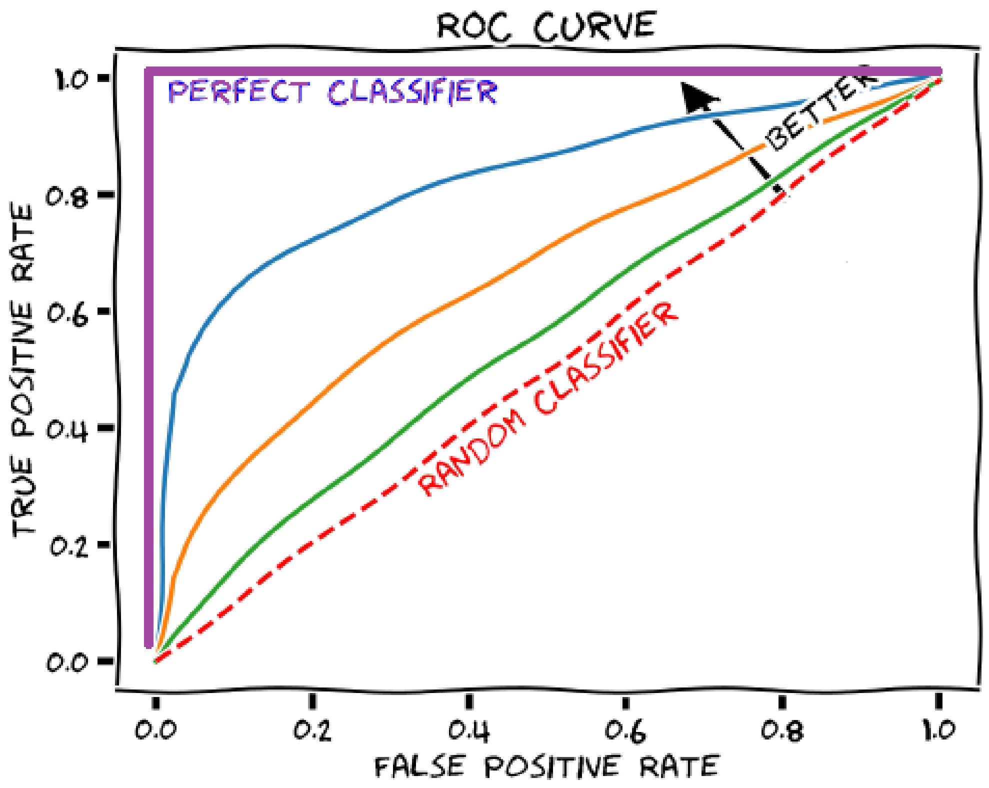
- How do we set a **decision threshold** γ?
- Use an **ROC curve** of TPR vs FPR
- Pick a set of different thresholds for γ and observe how TP and FP rates vary:

$$TPR_\gamma = p(\hat{y} = 1 | y = 1, \gamma) = \frac{TP_\gamma}{TP_\gamma + FN_\gamma}$$

Recall/Hit Rate

$$FPR_\gamma = p(\hat{y} = 1 | y = 0, \gamma) = \frac{FP_\gamma}{FP_\gamma + TN_\gamma}$$

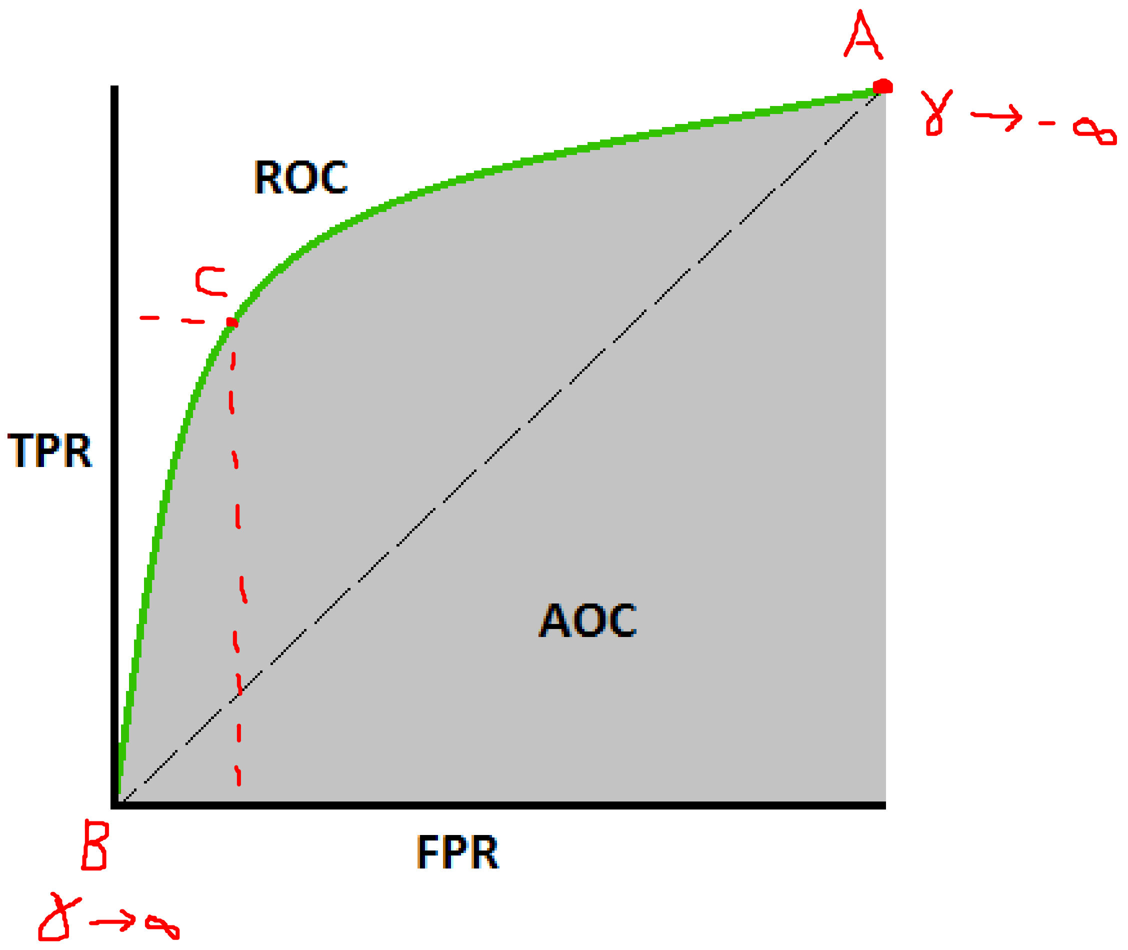
False Alarm Rate



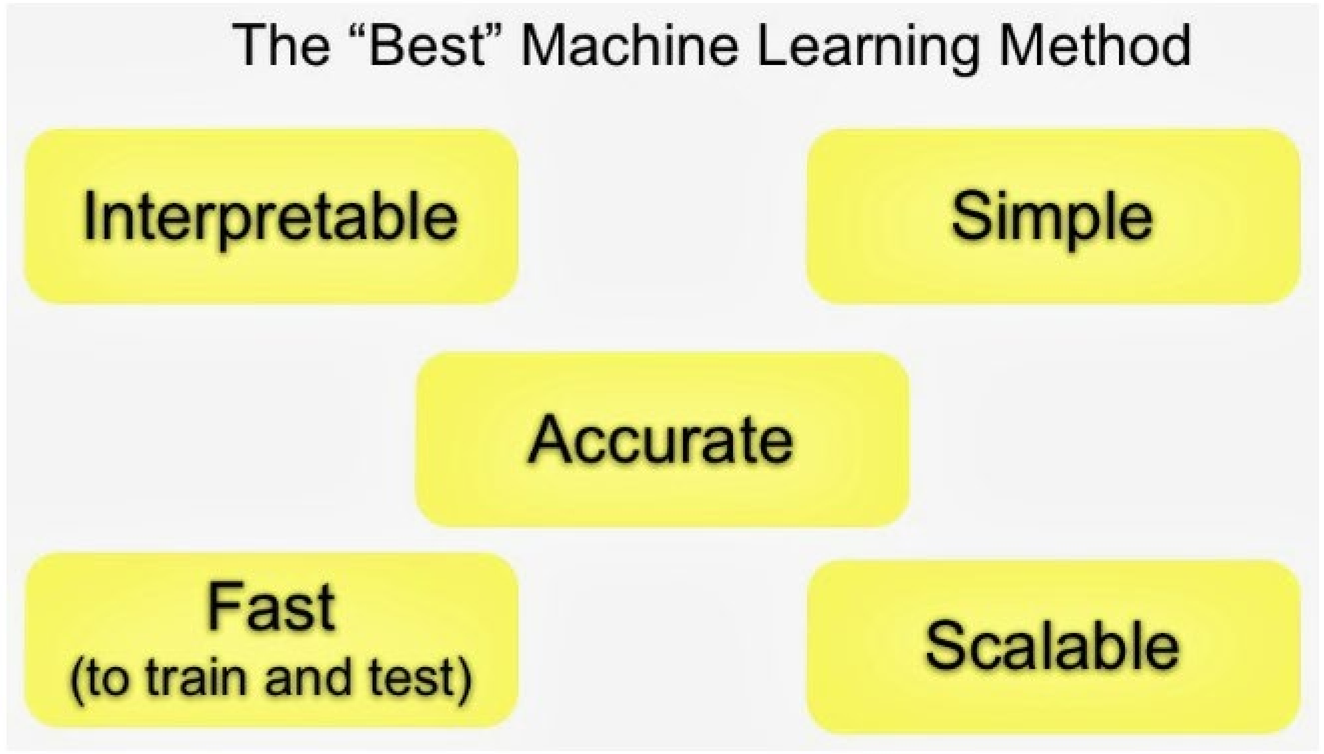
https://www.turing.com/kb/auc-roc-curves-and-their-usage-for-classification-in-python

Classifier Evaluation – ROC Curves (2)

- At **A** always predicting positive
- At **B** never predicting positive
- **C** is an example of a committed γ value
- **Area Under Curve (AOC)** can summarize ROC with a scalar where higher is better and 1 is maximum



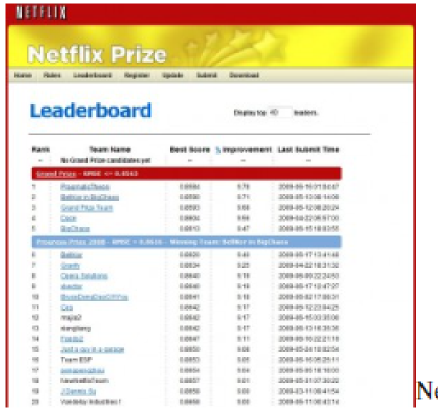
It’s Not All About Accuracy



http://radar.oreilly.com/2013/09/gaining-access-to-the-best-machine-learning-methods.html

Netflix Never Used Its $1 Million Algorithm Due To Engineering Costs

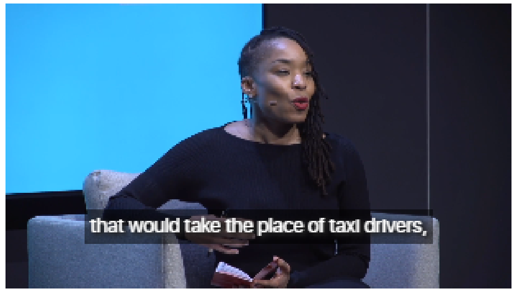
Netflix awarded a $1 million prize to a developer team in 2009 for an algorithm that increased the accuracy of the company's recommendation engine by 10 percent. But it doesn't use the million-dollar code, and has no plans to implement it in the future, Netflix announced on its blog Friday. The post goes on to explain why: [...]



Netflix awarded a $1 million prize to a developer team in 2009 for an algorithm that increased the accuracy of the company's recommendation engine by 10 percent. But it doesn't use the million-dollar code, and has no plans to implement it in the future, Netflix announced on its blog Friday. The post goes on to explain why: a combination of too much engineering effort for the results, and a shift from movie recommendations to the "next level" of personalization caused by the transition of the business from mailed DVDs to video streaming.

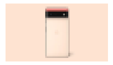
Netflix notes that it does still use two algorithms from the team that won the first Progress Prize for an 8.43 percent improvement to the recommendation engine's root mean squared error (the full $1 million was awarded for a 10 percent improvement). But the increase in accuracy on the winning improvements "did not seem to justify the engineering effort needed to bring them into a production environment," the blog post said. By that time, the company had moved on anyway.

TRENDING NOW



Automation Anywhere Presents Mihir Shukla in Conversation with Jahna Berry

Most Popular



GEAR

9 Early Amazon Prime Day Deals on Google Hardware

MEDEA GIORDANO

CULTURE

Westworld Has Entered the New, Better Frontier of Sci-Fi

ANGELA WATERCUTTER

https://www.wired.com/2012/04/netflix-prize-costs/

Concluding Remarks

- Introduced **Fisher's LDA**, a **linear classifier**, and how to **evaluate** it
- Look at *"lda_example.py"* Python or corresponding Matlab script
- Check out additional notes uploaded on Canvas
- Questions?