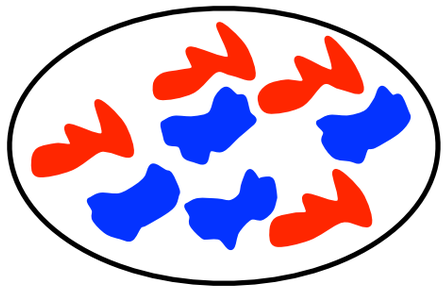
Elements of Statistical Learning



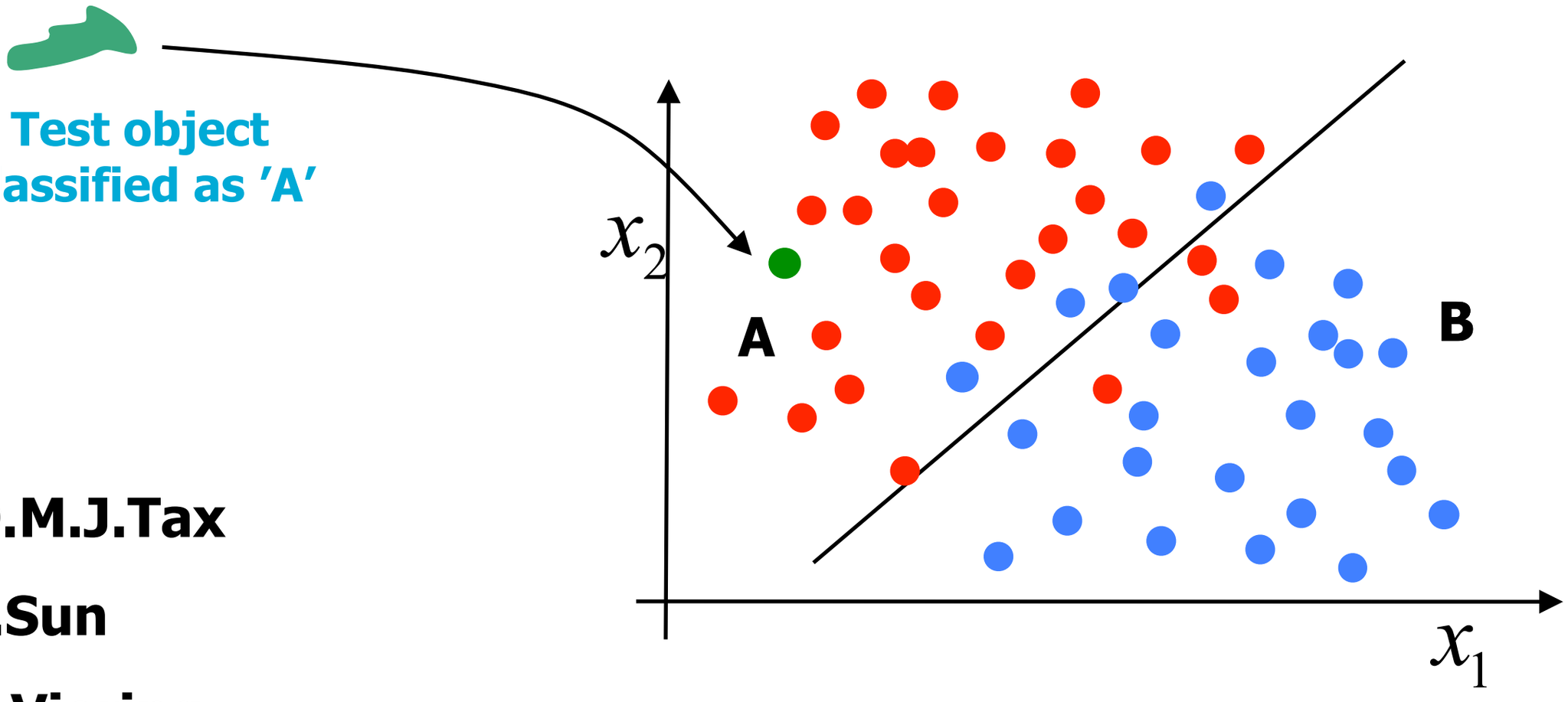
Sensor

Representation

Generalization

Feature Space

Classification

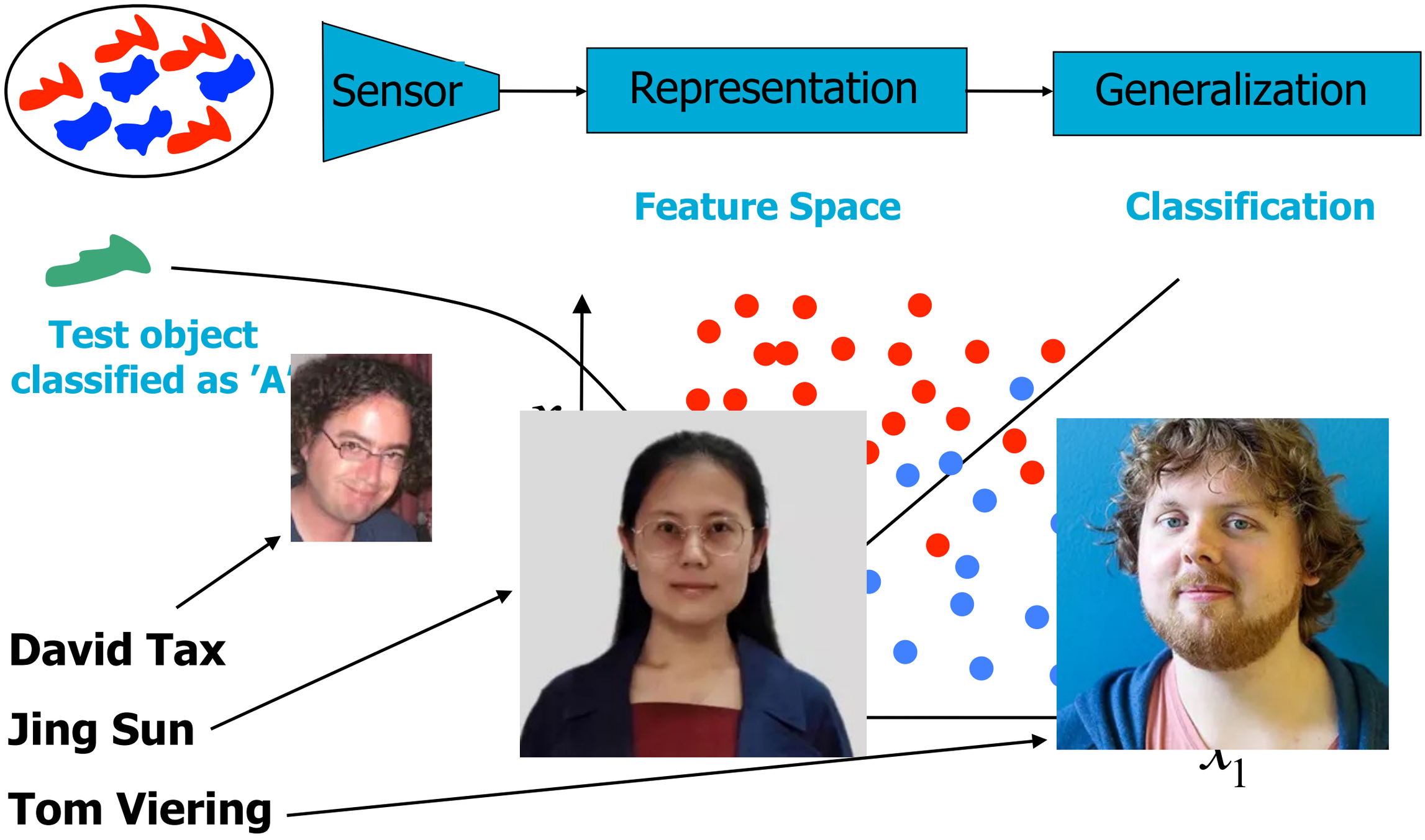


D.M.J.Tax

J.Sun

T.Viering

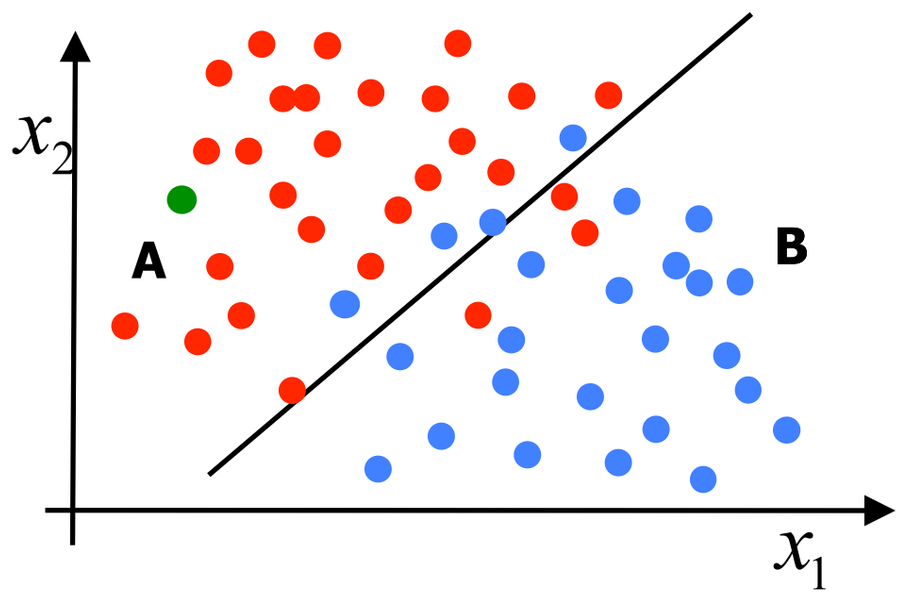
Elements of Statistical Learning



TU Delft

Learning

- Learning concept from data
- Learn to distinguish classes
- Learn to play a game
- Learn to accomplish a task
- Learn to generate samples



- Data: randomly? can we ask questions? (EXAMPLES/ORACLE)

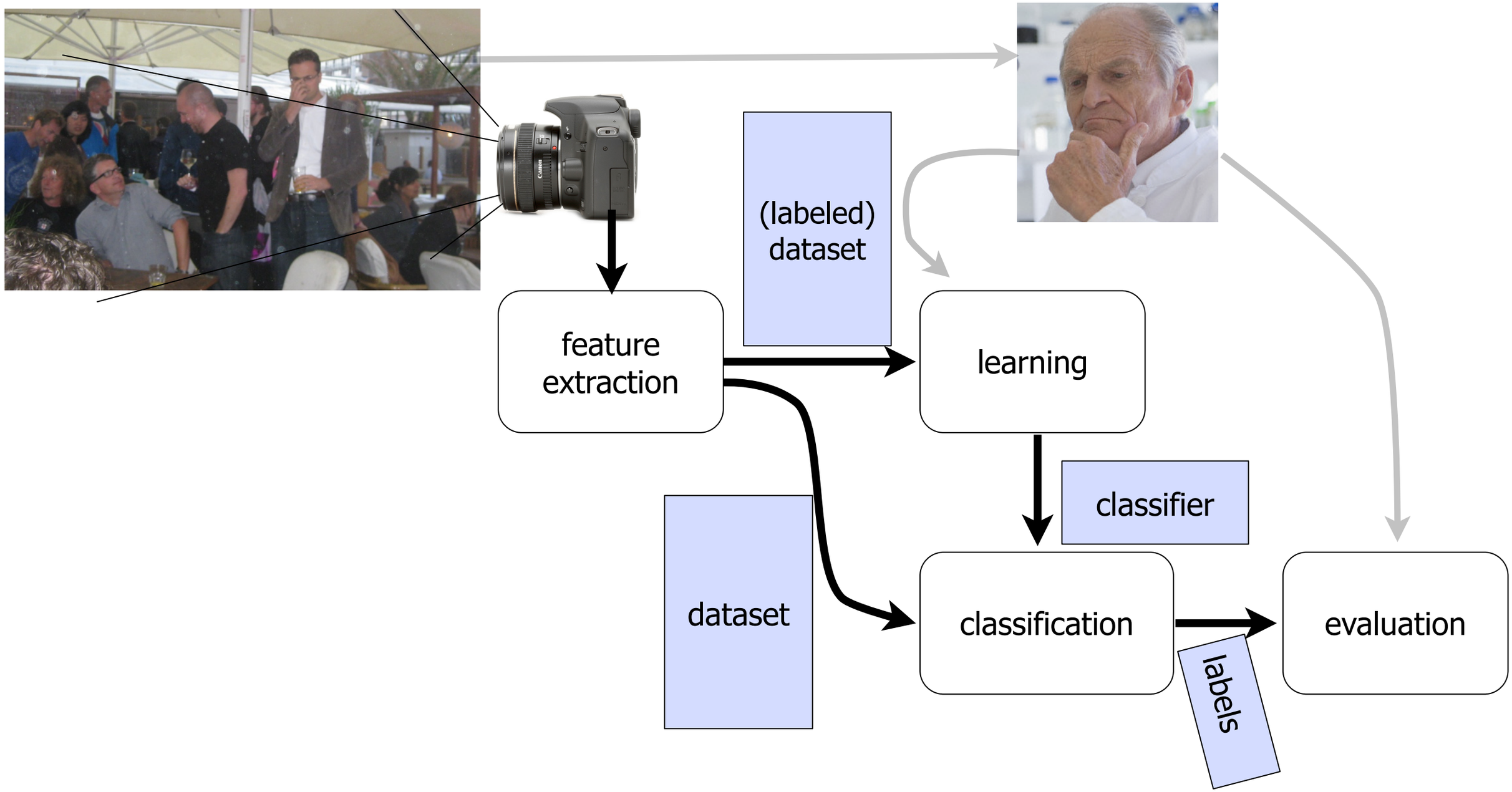
- Label: direct feedback? after the task is completed? depending on the action?

Statistical Learning

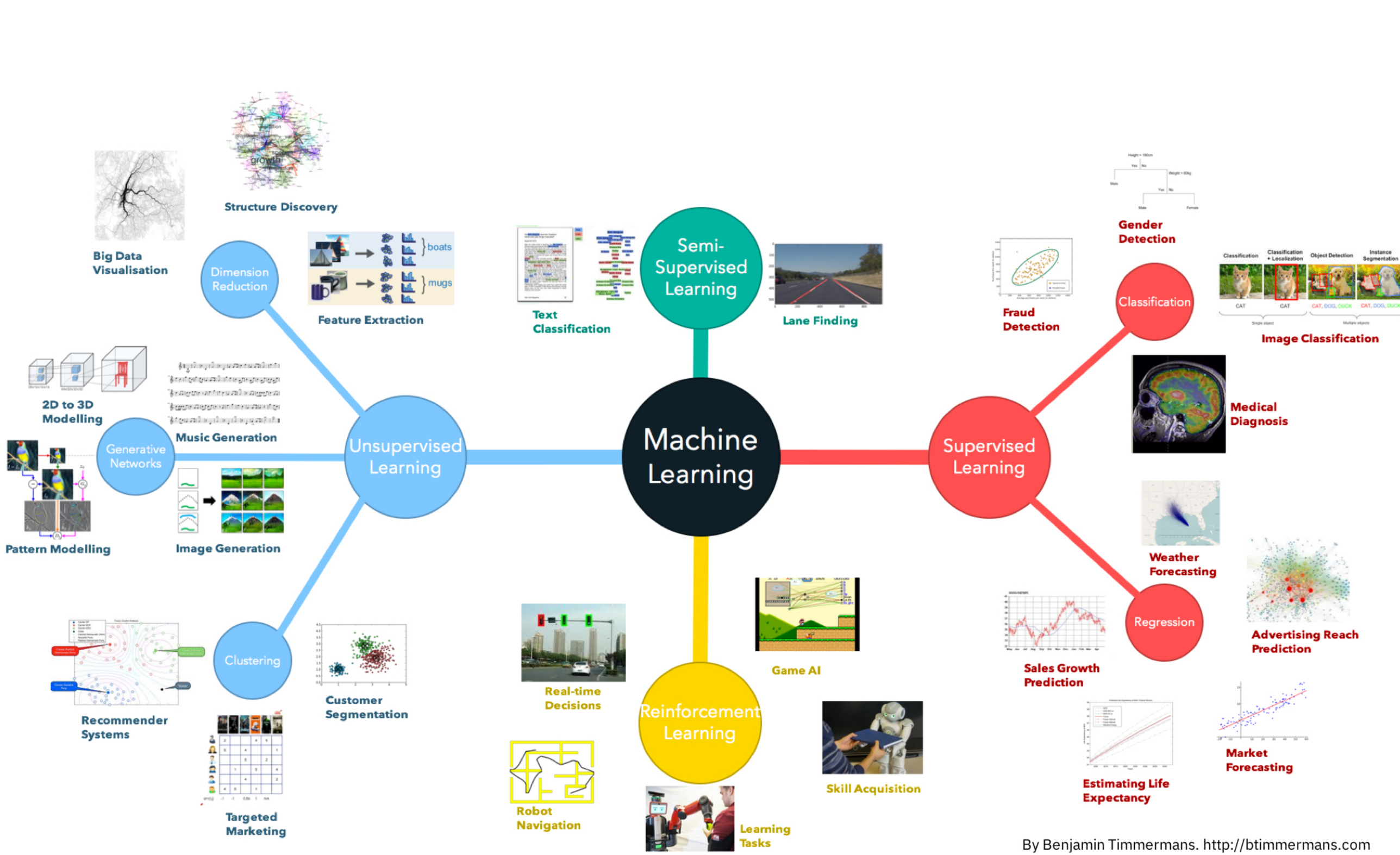
Apple iPhoto'09

- Everyday tasks are deceptively difficult
- You have to be able to recognise places, situations, objects, people
- To do this automatically, is the task of pattern recognition: **learning from examples**

Pattern recognition pipeline



- Check with independent test data how well it works!



By Benjamin Timmermans. http://btimmermans.com

Machine Learning

- Prior knowledge: CSE2510 Machine Learning and DSAIT4005 Machine and Deep Learning:
  Linear algebra, basic probability theory and statistics

- Material/literature:
  Different books/chapters per lecture

- This lecture: Section 2.1-2.6 of
  Pattern recognition, by Sergios Theodoridis, Konstantinos Koutroumbas (2009)

Schedule of the course

- Week 1: Basic Machine Learning: classification
- Week 2: Computational Learning Theory
- Week 3: Regression
- Week 4: Feature Reduction
- Week 5: Clustering
- Week 6: Practical ML and fairness
- Week 7: Q&A
- Week 8,9: no lecture, you work on the final project

- Week 9: Deadline final project
- Week 10: Exam!

Lectures, exercises, project and Exam

- Lectures are not mandatory
- (Non-mandatory) exercises in Python
- Two hours per week, TAs present to answer questions, two hours for yourself

- Start with a practice project
- Final project

- Exams are on-campus
- Final grade is a combination of exam and final project (70-30%)

More on project(s)

- Projects are done in groups of 5
- Groups are pre-defined, see Brightspace. If not assigned to group: come to me!
- Start with a **practice** project: solve a given classification problem
- Just to get you started
- Description in the Exercise-manual (Brightspace)

- The **final** project counts for a grade
- Find your own problem!
- Discuss with us (before December) if it is suitable

Generative AI/chatGPT/…

- Please don’t use these Large Language Models to answer an exercise question
- You don’t learn anything from that
- We’re NOT going to explain the output of chatGPT to you

- We are not here to teach you to use chatGPT,
  we are here to teach you Machine Learning, and how to apply it responsibly

This week

- Introduction, administrative stuff
- Learning from examples, some definitions
- General formulation of ML
- How do known methods fit the formulation
- Learning curves, feature curves
- Leftovers from Classification:
  - ROC and AUC, precision-recall curves
  - Platt scaling and Multi-class classifiers
  - Calibrated classifiers
  - Missing values
  - Feature scaling

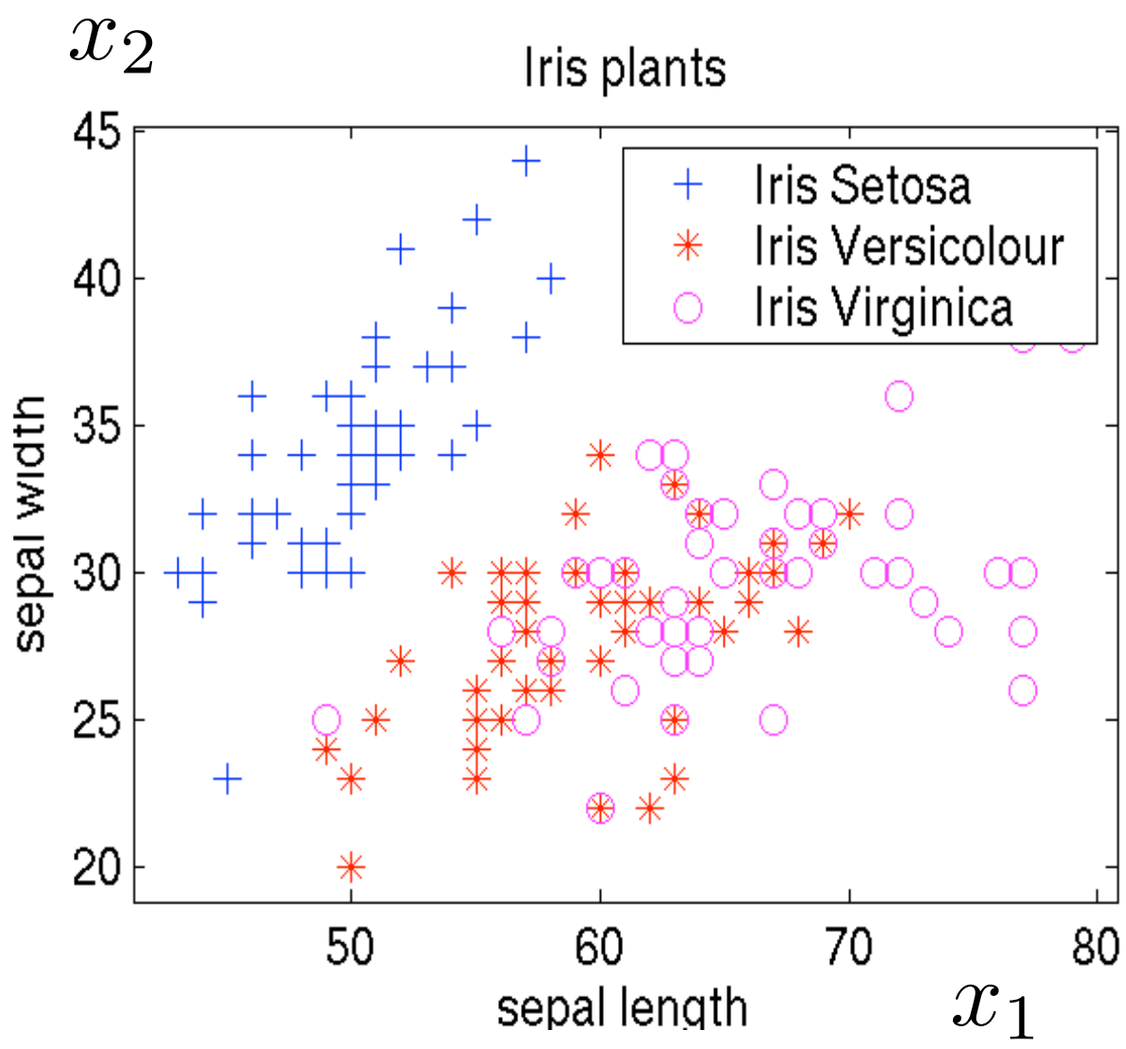
Objects in feature space

- We can interpret the measurements as a vector in a vector space:

$$\mathbf{x} = (x_1, x_2, ..., x_p)$$

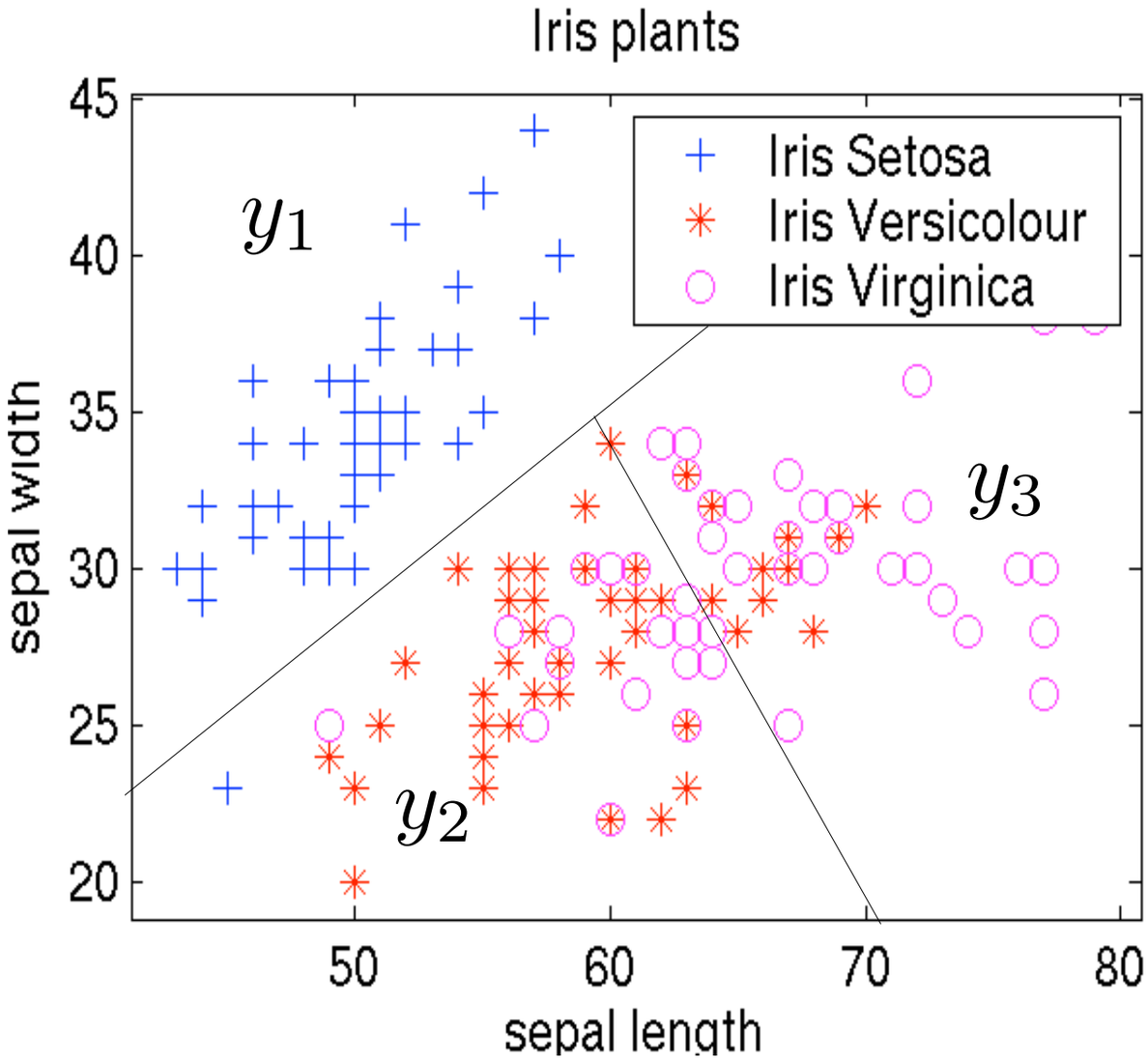
- This originates, in principle, from a probability density over the whole feature space

$$p(\mathbf{x}, y)$$

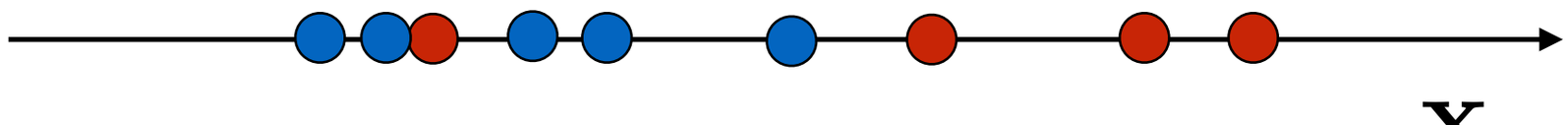


Classification

- Given labeled data: $\mathbf{x}$
- Assign to each object a class label y
- In effect splits the feature space in separate regions

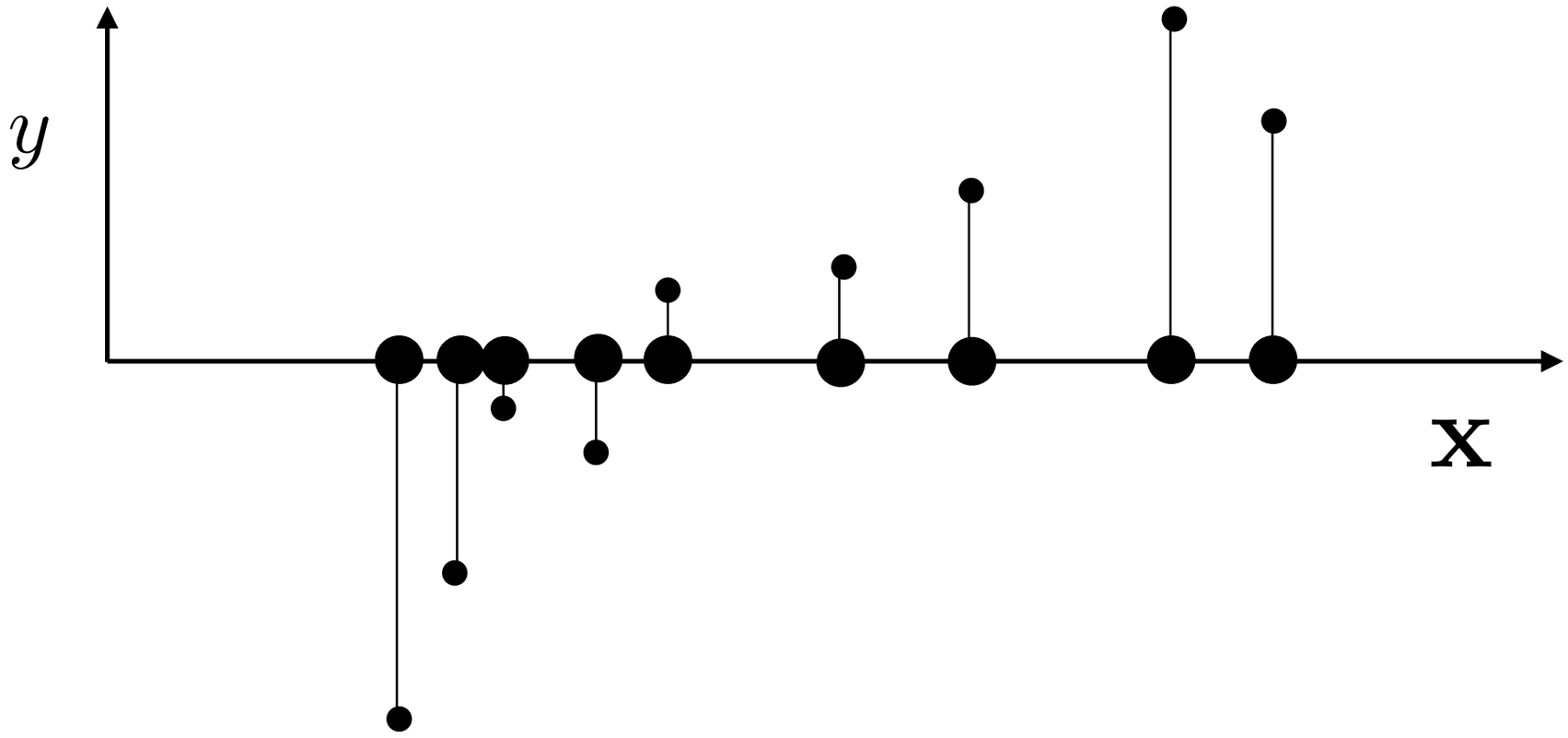


Also for regression, clustering, ...



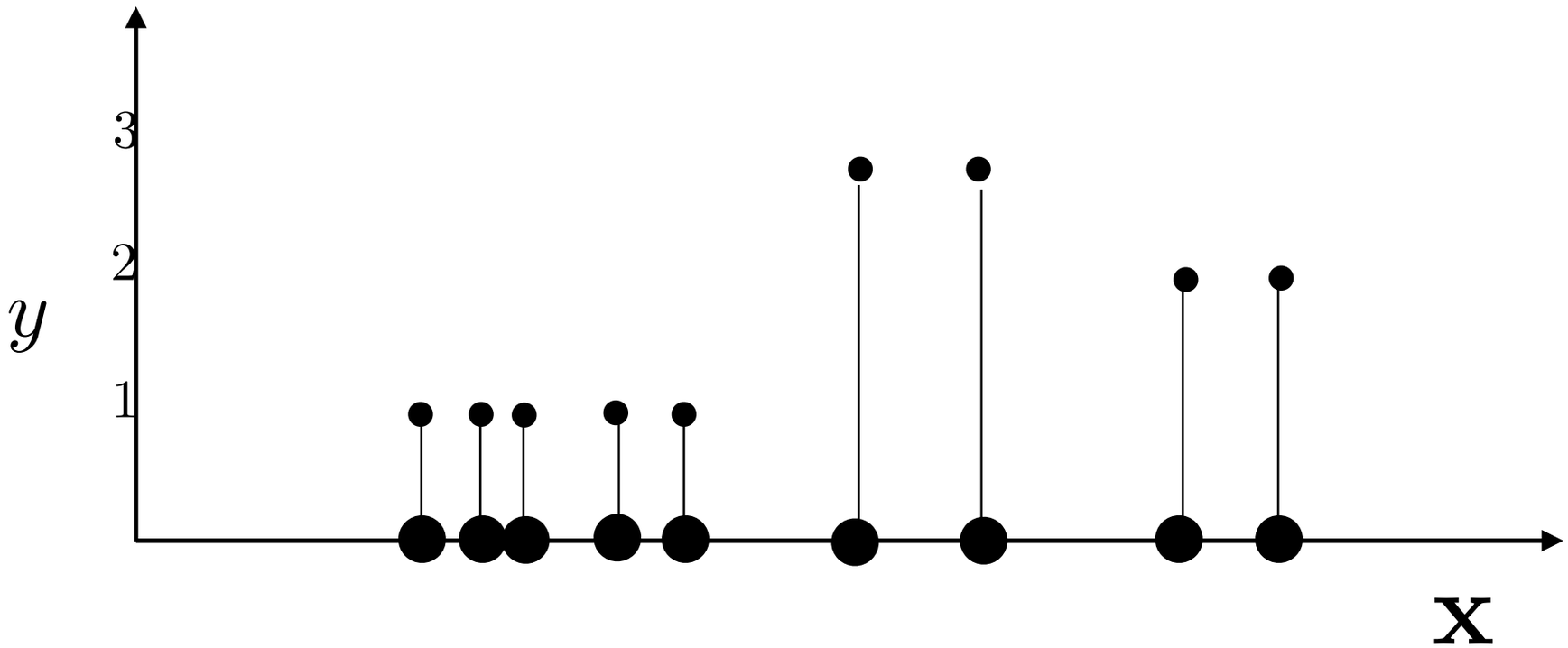
- Instead of class labels, real values: regression
- Instead of supervised labels, no labels: clustering

Also for regression, clustering, ...



- Instead of class labels, real values: regression
- Instead of supervised labels, no labels: clustering

Also for regression, clustering, ...



- Instead of class labels, real values: regression
- Instead of supervised labels, no labels: clustering

General formulation

- (Almost) all machine learning can be reduce to:

- Define a model family $f(\mathbf{x};\theta)$ with parameters θ
- Define a loss $l(\hat{y}, y)$ and a regularizer $\Omega(\theta)$

- Given a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, minimize the loss

$$L(\theta) = \sum_{i=1}^{N} l(f(\mathbf{x}_i;\theta), y_i) + \Omega(\theta)$$

and obtain the optimal $\hat{\theta}$

What classifiers do you know?

- Quadratic discriminant classifier
- Linear discriminant classifier
- Nearest mean classifier
- Parzen classifier
- kNN classifier
- Logistic classifier
- Perceptron classifier
- Neural networks
- Decision trees
- Support Vector Classifier
- …

Quadratic Discriminant Classifier

- How does the model look like?
- How does the loss look like?

Quadratic Discriminant Classifier

- How does the model look like?
  - Plug-in Bayes classifier:

$$f(\mathbf{x}) = \log p(y_1|\mathbf{x}) - \log p(y_2|\mathbf{x})$$

  - Gaussian per class:

$$\log(\hat{p}(y_i|\mathbf{x})) = -\frac{p}{2}\log(2\pi) - \frac{1}{2}\log(\det \Sigma_i)$$
$$-\frac{1}{2}(\mathbf{x}-\mu_i)^T\Sigma_i^{-1}(\mathbf{x}-\mu_i) + \log p(y_i) - \log p(\mathbf{x})$$

- How does the loss look like?
  - Maximum (log-)likelihood:

$$L = \prod_{i=1}^{N_1} p(\mathbf{x}_i^{(1)}|y_1) \prod_{i=1}^{N_2} p(\mathbf{x}_i^{(2)}|y_2)$$

Maximum likelihood

- Function to maximize:

$$\ln(L) = \sum_{i=1}^{N_1} \ln p(\mathbf{x}_i^{(1)}|y_1) + \sum_{i=1}^{N_2} \ln p(\mathbf{x}_i^{(2)}|y_2)$$

- Using Gaussian distribution

$$\hat{p}(\mathbf{x}|y) = \frac{1}{\sqrt{2\pi^p \det(\hat{\Sigma}_y)}} \exp\left(-\frac{1}{2}(\mathbf{x}-\hat{\mu}_y)^T \hat{\Sigma}_y^{-1}(\mathbf{x}-\hat{\mu}_y)\right)$$

- What are the parameters to optimize?

Maximum likelihood for mean 1

- Consider the optimization of μ_1:

$$L(\mu_1) = \sum_{i=1}^{N_1} \ln p(\mathbf{x}_i^{(1)} | \mu_1, \Sigma_1)$$

- Find maximum: $$\frac{\partial L(\mu_1)}{\partial \mu_1} = 0$$

- Solution: $$\hat{\mu}_1 = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i^{(1)}$$

(Similarly for mean 2)

Formulation for LDA, nearest mean?

- The model functions are basically the same
- The shape of the covariance matrices are restricted

Formulation for Parzen, kNN?

- How does the model look like?

$$f(\mathbf{x}) = p(\mathbf{x}|y_1)p(y_1) - p(\mathbf{x}|y_2)p(y_2)$$

$$p(\mathbf{x}|y) = \frac{1}{N_y}\sum_{i=1}^{N_y} K(\mathbf{x}|\mathbf{x}_i, h\mathbf{I})$$

$$p(\mathbf{x}|y) = \frac{k}{N_y}\frac{1}{V(\mathbf{x})}$$

- How does the loss look like?
  Hmm… what are the free parameters?
  What if you optimize?

$$L = \prod_{i=1}^{N_1} p(\mathbf{x}_i^{(1)}|y_1) \prod_{i=1}^{N_2} p(\mathbf{x}_i^{(2)}|y_2)$$

Logistic classifier

- The function model is $\ln\left(\frac{p(y_1|\mathbf{x})}{p(y_2|\mathbf{x})}\right) = \beta_0 + \beta^T\mathbf{x}$ or

$$p(y_2|\mathbf{x}) = \frac{1}{1+\exp(\beta_0+\beta^T\mathbf{x})}$$

- The loss is again the (log-)likelihood

$$\ln(L) = \sum_{i=1}^{N_1} \ln p(\mathbf{x}_i^{(1)}|y_1) + \sum_{i=1}^{N_2} \ln p(\mathbf{x}_i^{(2)}|y_2)$$

- How to optimize the loss?
  Gradient descent:

$$\beta_{new} = \beta_{old} + \eta\frac{\partial \ln(L)}{\partial\beta}$$

Optimizing the logistic

- Use Bayes theorem to convert $p(\mathbf{x}|y_i)$ into $p(y_i|\mathbf{x})$

$$\ln(L) = \sum_{i=1}^{N_1} \ln p(y_1|\mathbf{x}_i^{(1)}) + \sum_{i=1}^{N_2} \ln p(y_2|\mathbf{x}_i^{(2)}) + K$$

- Fill in the logistic function, and obtain:

$$\ln(L') = \sum_{i=1}^{n_1} (\beta_0 + \beta^T \mathbf{x}_i^{(1)})_{n_1+n_2} - \sum_{i=1} \ln(1 + \exp(\beta_0 + \beta^T \mathbf{x}_i))$$

Derivative of the ln(L)

- Take the derivative of ln(L) with respect to the parameters:

$$\frac{\partial \ln(L)}{\partial \beta_0} = N_1 - \sum_{i=1}^{N_1+N_2} p(y_1|\mathbf{x}_i)$$

$$\frac{\partial \ln(L)}{\partial \beta_j} = \sum_{i=1}^{N_1} (\mathbf{x}_i^{(1)})_j - \sum_{i=1}^{N_1+N_2} p(y_1|\mathbf{x}_i)(\mathbf{x}_i)_j$$

- Take initial values $\beta_0 = 0,\ \beta = \mathbf{0}$
- Keep iterating

$$\beta_{new} = \beta_{old} + \eta \frac{\partial \ln(L)}{\partial \beta}$$

till convergence

Perceptron, neural networks

- Perceptron: $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$

$$L = \sum_{i=1}^{N}(-y_i f(\mathbf{x}_i))_+$$

- Neural networks

$$f(\mathbf{x}) = g_L(...g_3(\mathbf{W}_2 g_2(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + b_2) + ...)$$

$$L = \sum_{i=1}^{N}(f(\mathbf{x}_i) - y_i)^2$$

or

$$L = -\sum_{i=1}^{N}(y_i \log(f(\mathbf{x}_i)) + (1 - y_i)\log(1 - f(\mathbf{x}_i)))$$

Other models?

Now for regression, clustering, …?

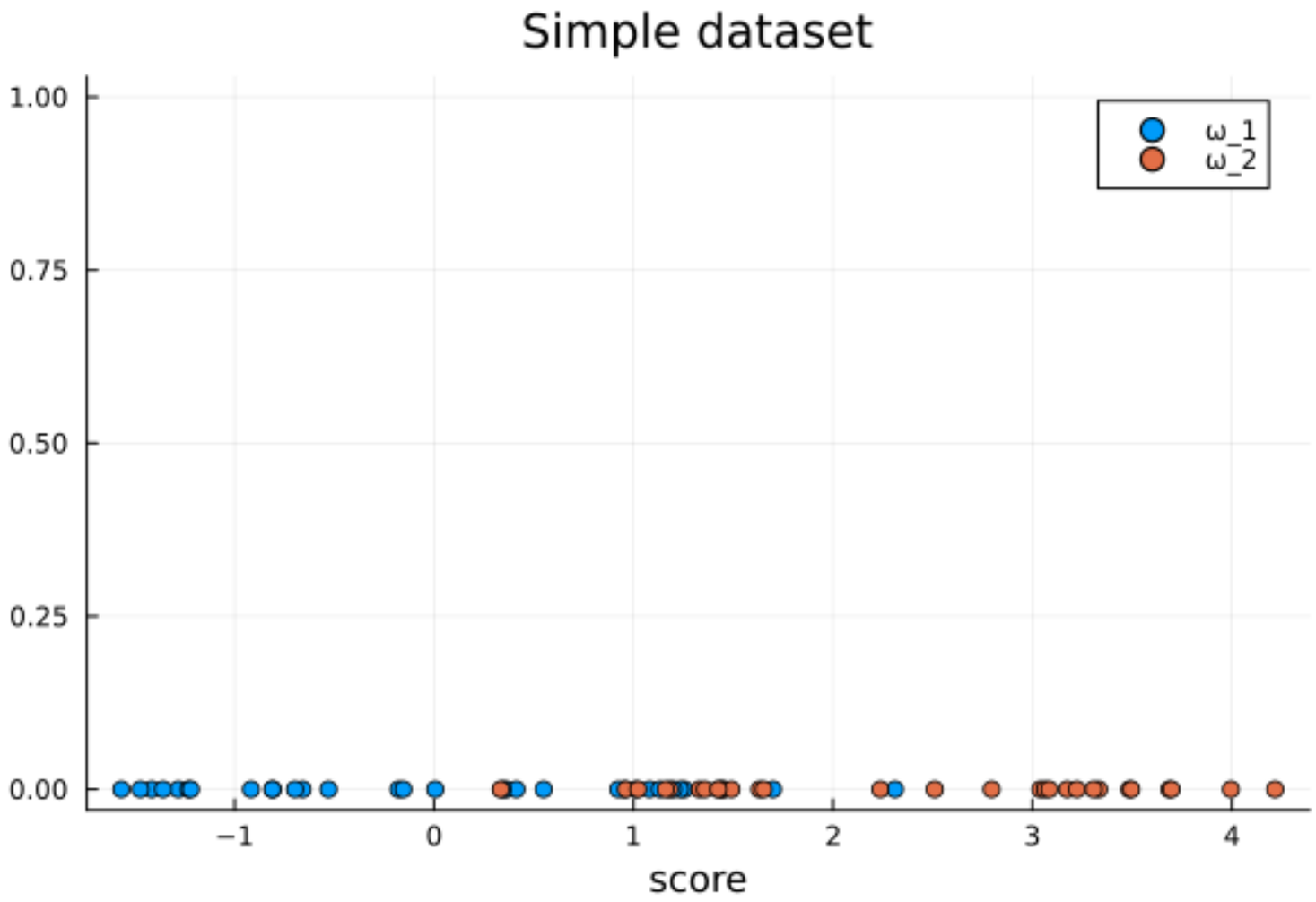
- Coming weeks we'll invent models f(**x**) and loss functions L for other problems, like regression, clustering and feature reduction

Left-over issues on classification

- Two-class classification
  - ROC curve, operating points,
  - Area under the ROC curve
  - Precision-recall curve
- Multi-class classification
- Platt scaling
- Missing values
- Sources of variation

Two-class classification

- Maybe the most basic ML problem(?)



- Given these outputs, how good is the model?

Two-class classification

- Often the classes are also 'asymmetric', in the sense that one class may be more 'important' than the other
- Postive/negative class:
  - Medical domain: + disease, - healthy
  - Retrieval: + relevant, - non-relevant
- Requires different treatment of the different errors
- Standard: $\varepsilon = \varepsilon_1 p(\omega_1) + \varepsilon_2 p(\omega_2)$
- Separate:

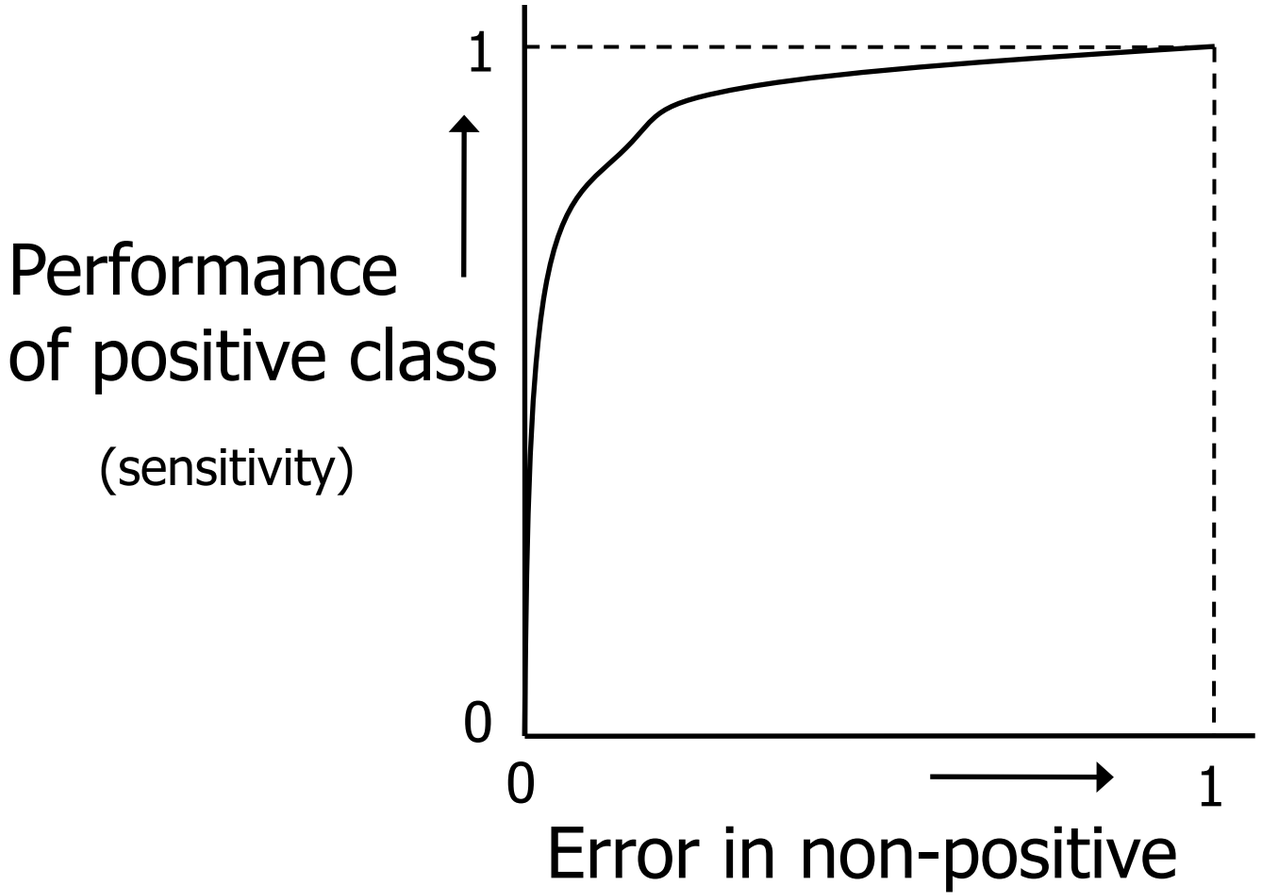
$$\text{spec.} = \text{true negative rate} = \frac{TN}{N}$$

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

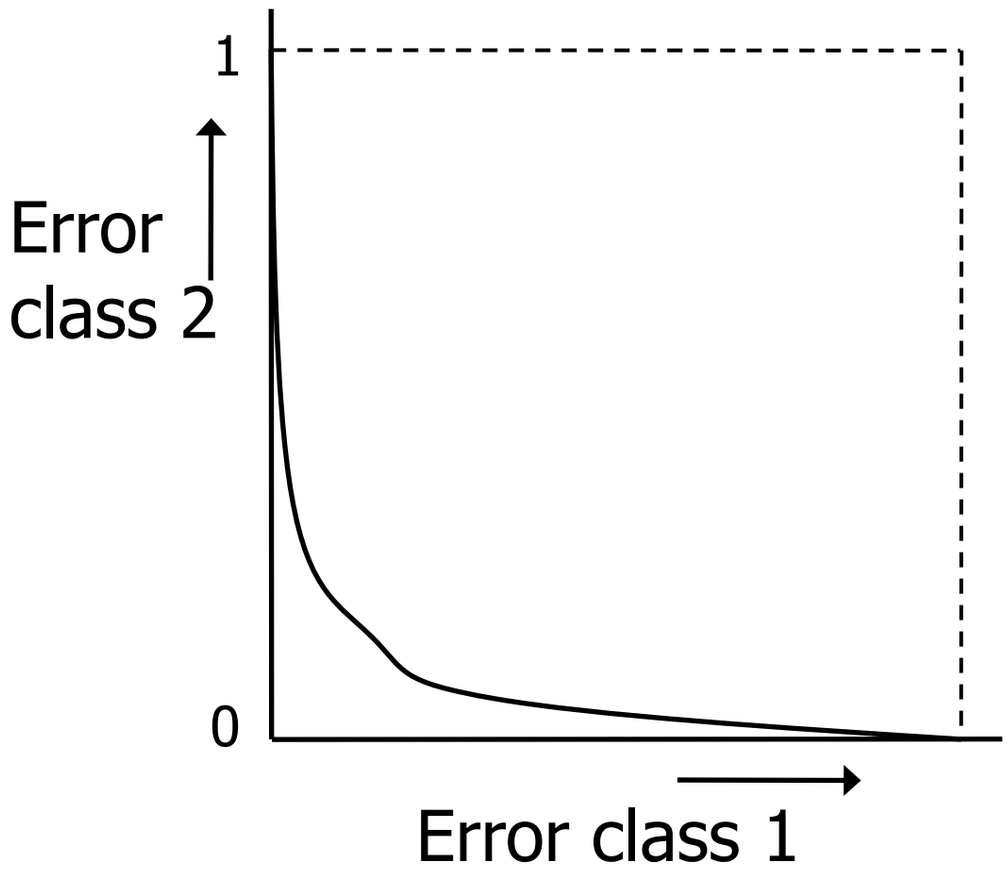
$$\text{sensitivity} = \text{recall} = \text{true positive rate} = \frac{TP}{P}$$

ROC Analysis

ROC: Receiver-Operator Characteristic (from communication theory)



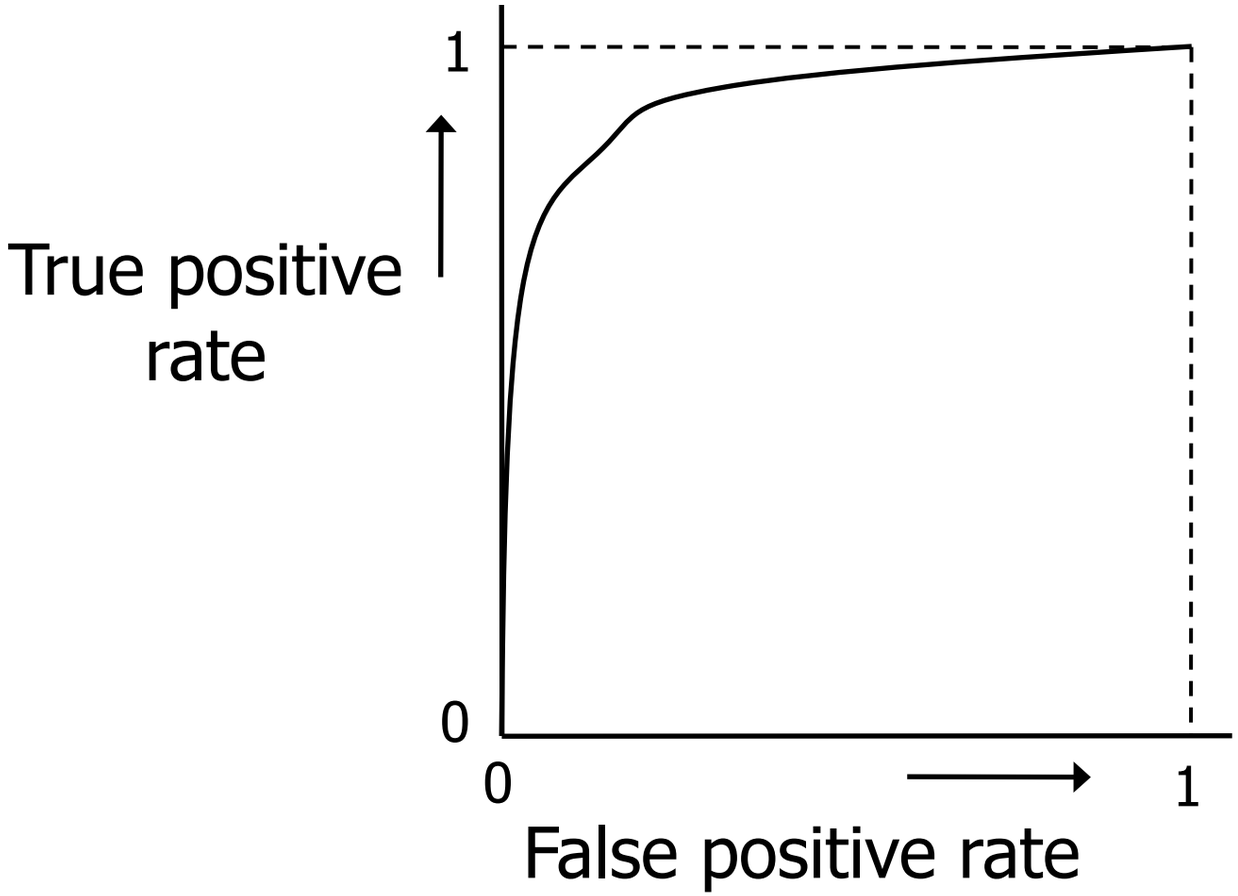
Medical diagnostics
Database retrieval



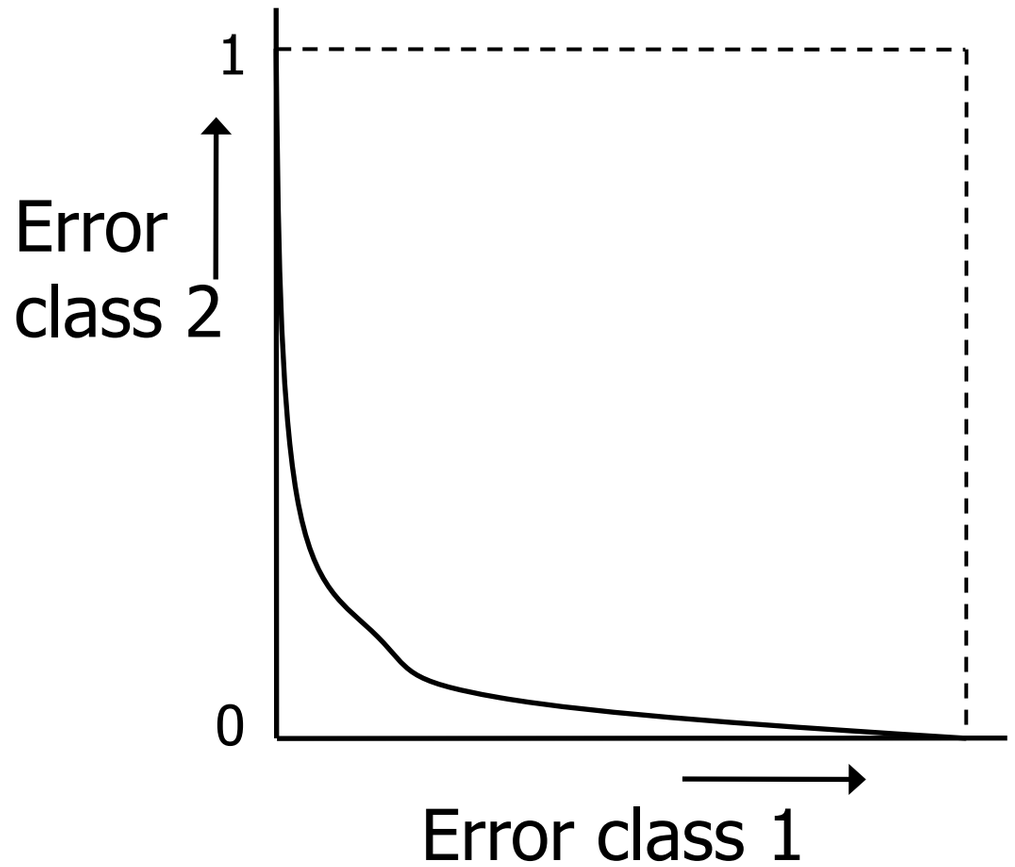
2-class pattern recognition

ROC Analysis

ROC: Receiver-Operator Characteristic (from communication theory)



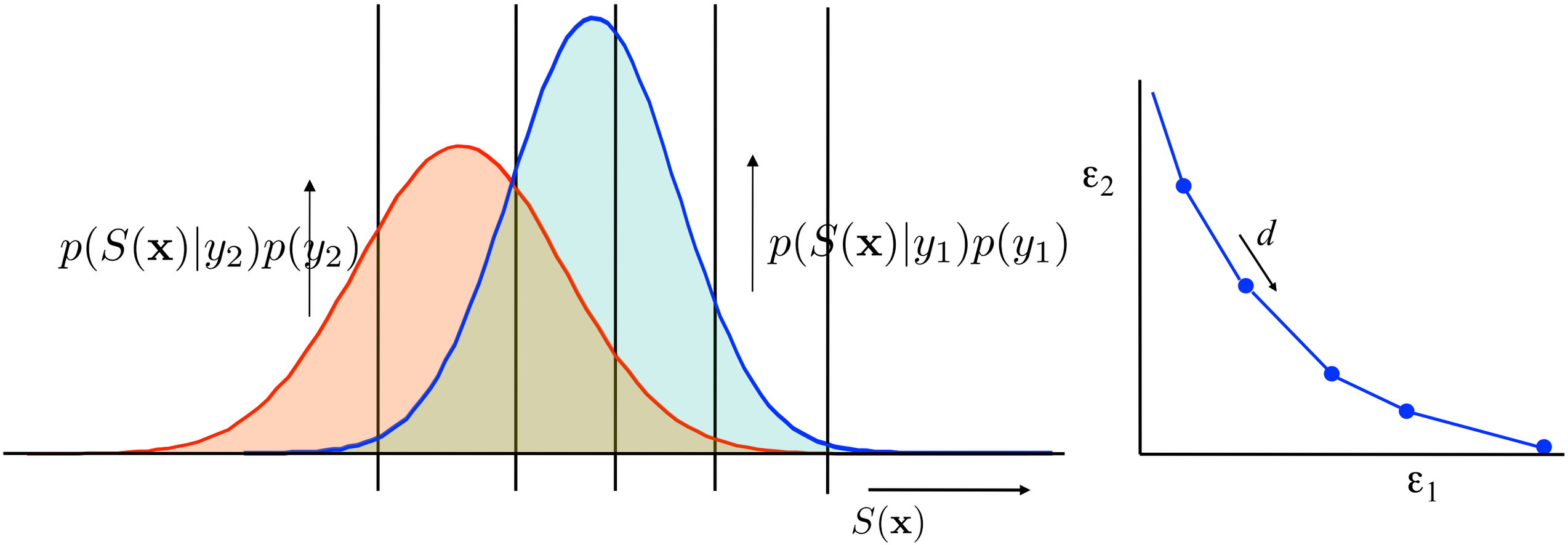
Medical diagnostics
Database retrieval



2-class pattern recognition

ROC Curve

- Curve is obtained by varying classifier threshold d



Example to compute ROC curve

- Assume I trained a classifier, and testing it on a test set:
- Four objects of class 1:

$$\hat{p}(y_1|\mathbf{x}_1) = 0.3$$
$$\hat{p}(y_1|\mathbf{x}_2) = 0.8$$
$$\hat{p}(y_1|\mathbf{x}_3) = 0.9$$
$$\hat{p}(y_1|\mathbf{x}_4) = 0.55$$

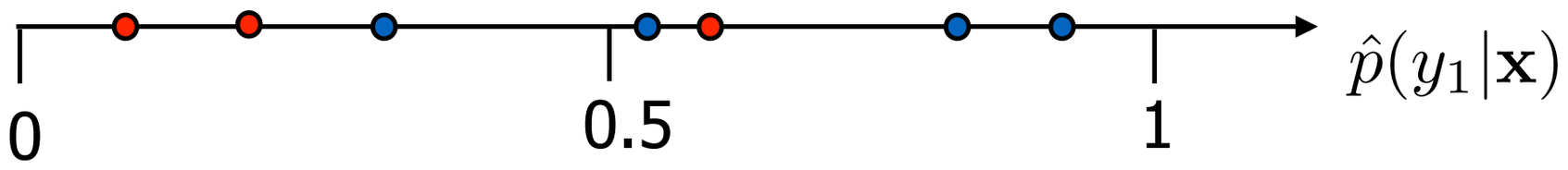
- Three objects of class 2:

$$\hat{p}(y_1|\mathbf{x}_5) = 0.2$$
$$\hat{p}(y_1|\mathbf{x}_6) = 0.1$$
$$\hat{p}(y_1|\mathbf{x}_7) = 0.6$$

- How does the ROC curve look like? ε_2 vs. ε_1

Example to compute ROC curve

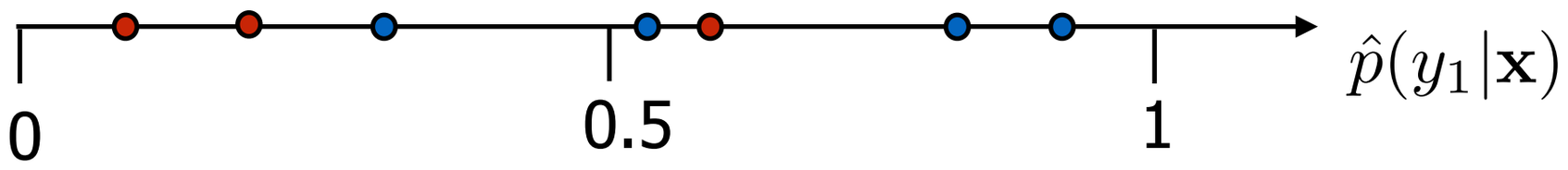
- Find out how obj's are classified for different thresholds



- For example, what are the errors for $d = 0.5$?

Example to compute ROC curve

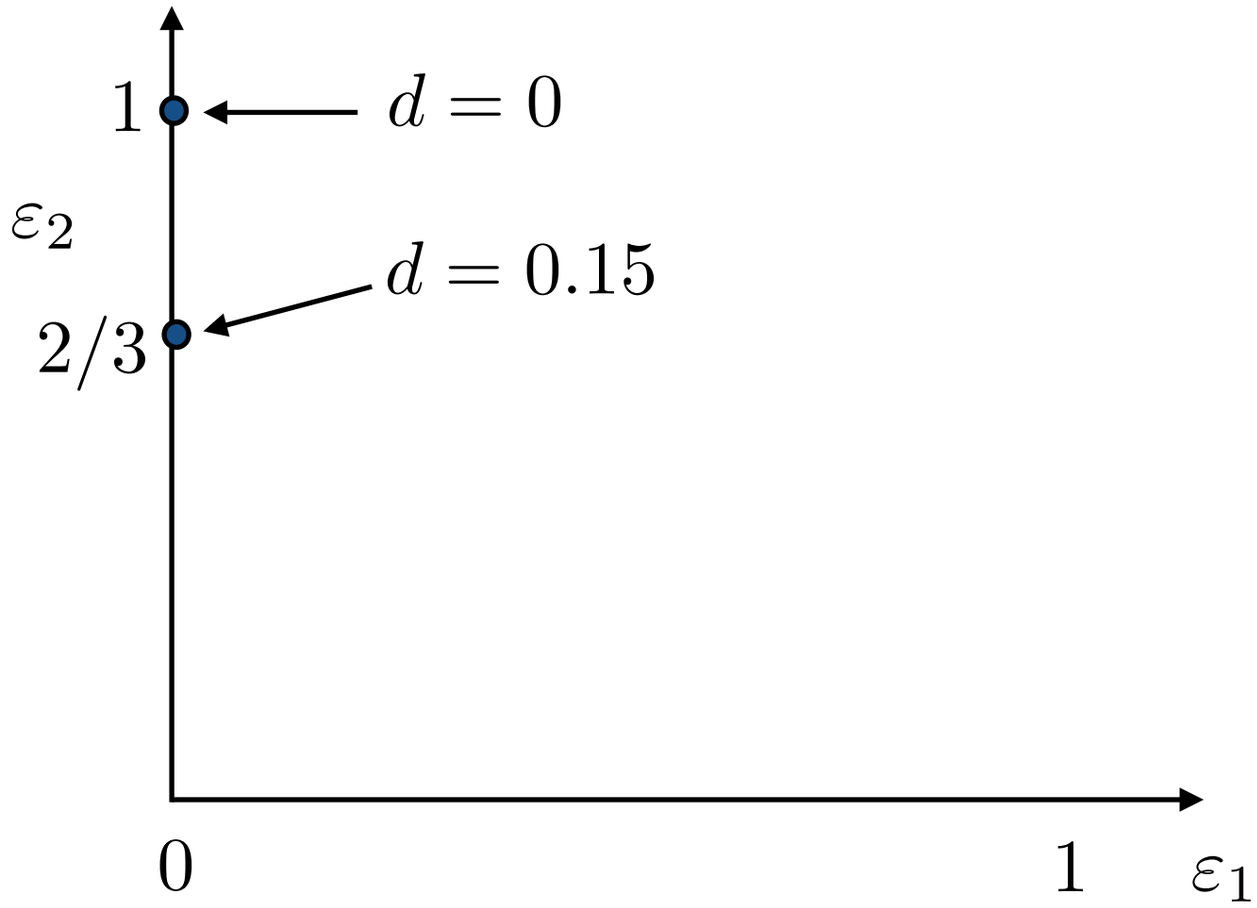
- Find out how obj's are classified for different thresholds



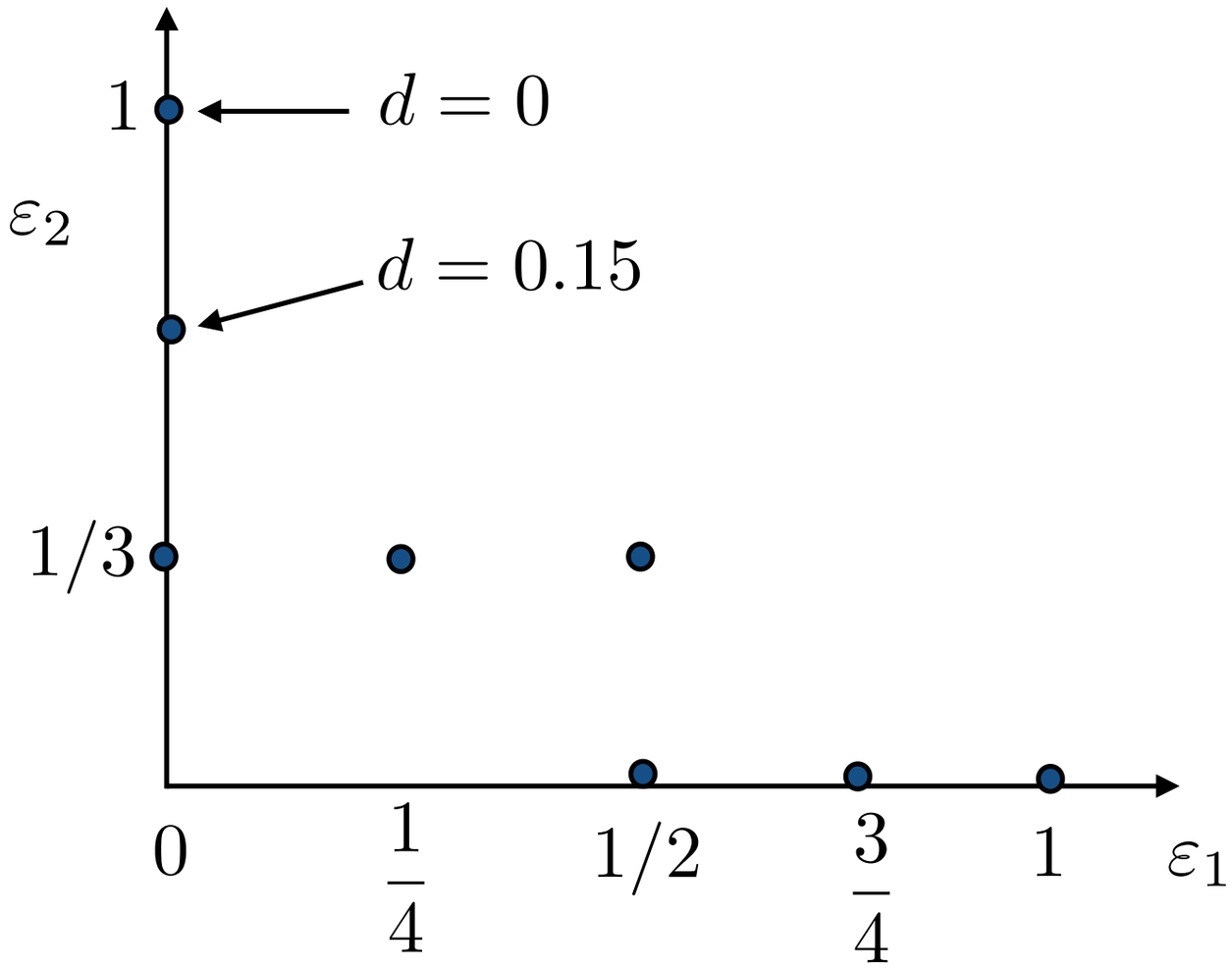
- For example, what are the errors for $d = 0.5$?

$$\varepsilon_1 = 1/4$$
$$\varepsilon_2 = 1/3$$

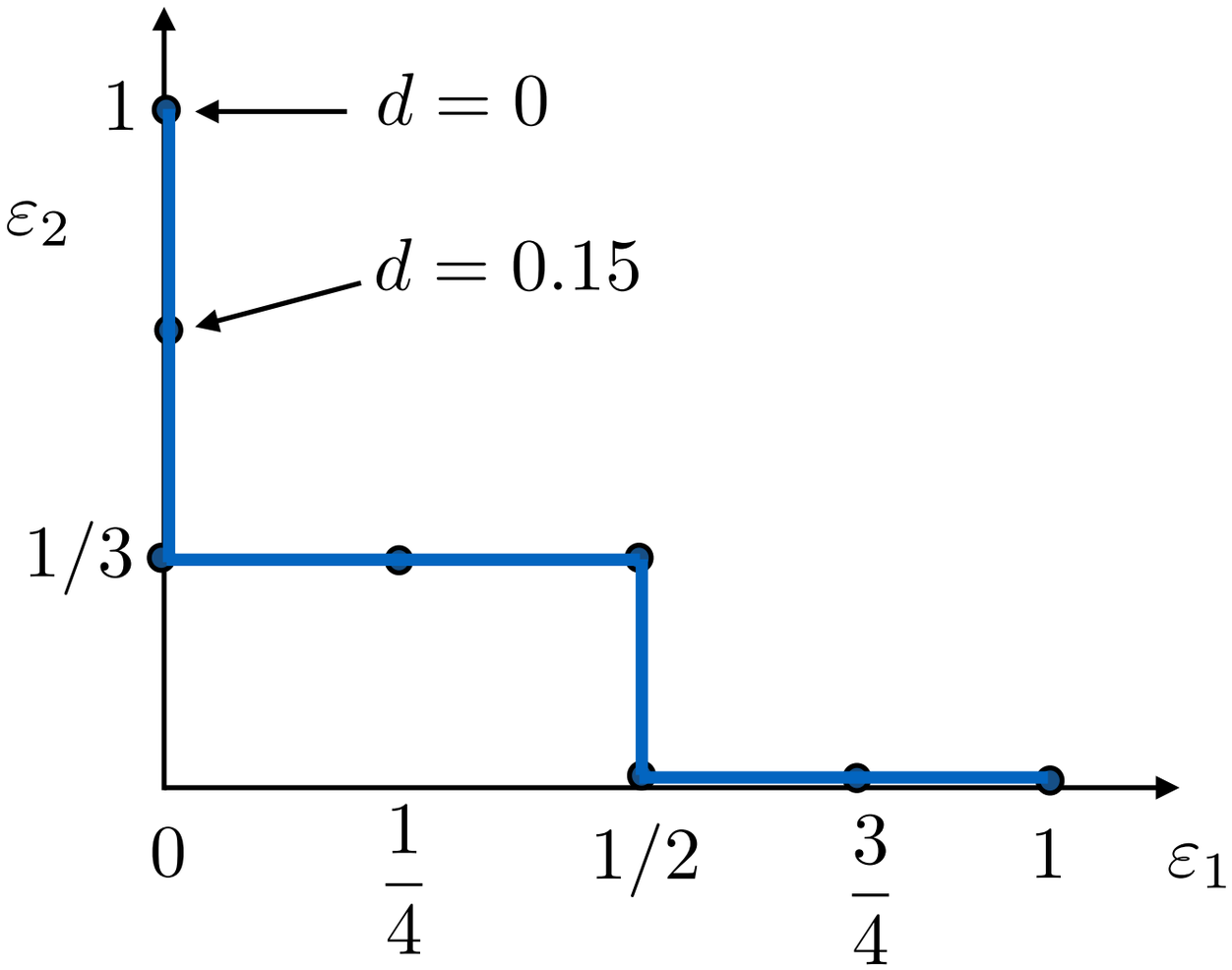
Example to compute ROC curve



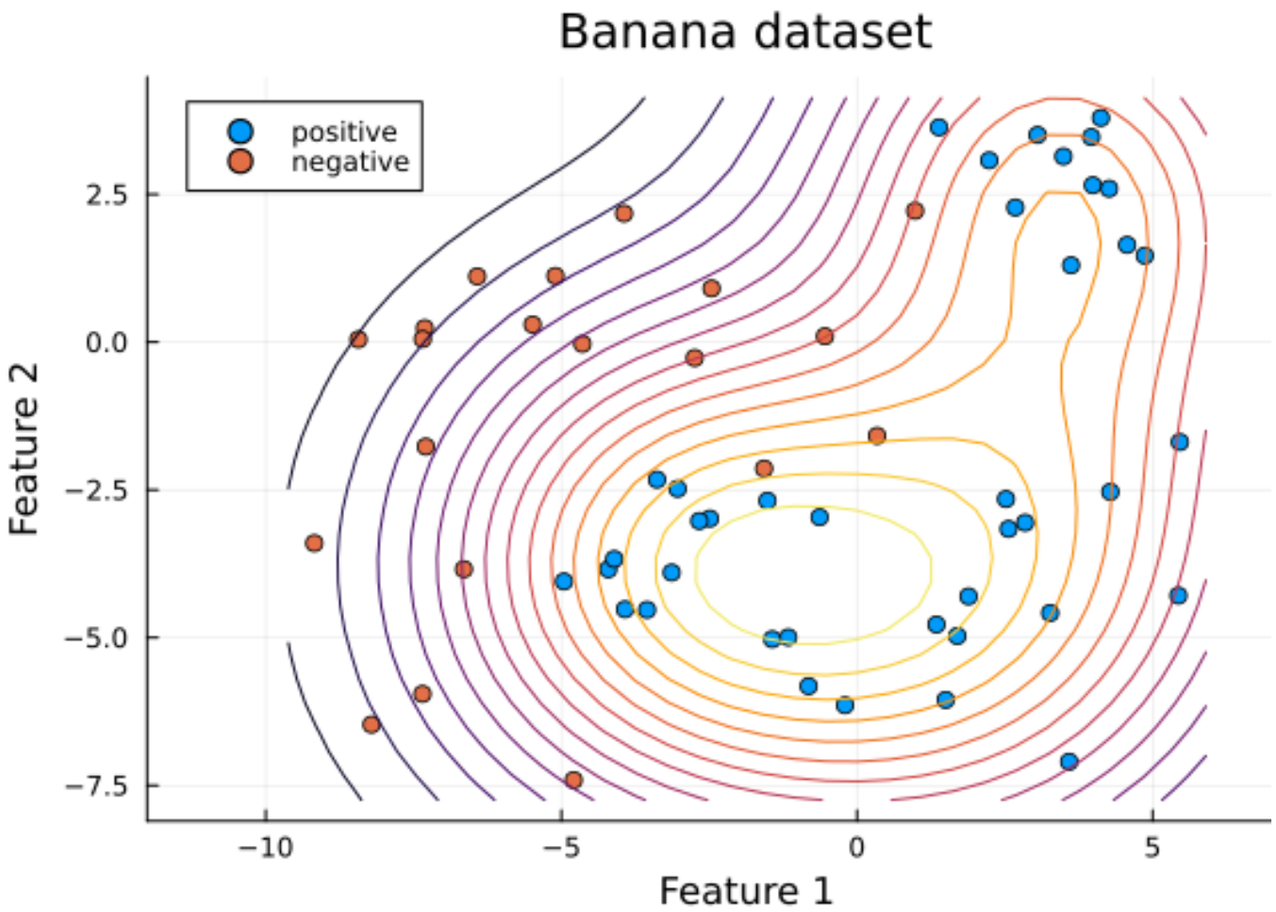
Example to compute ROC curve

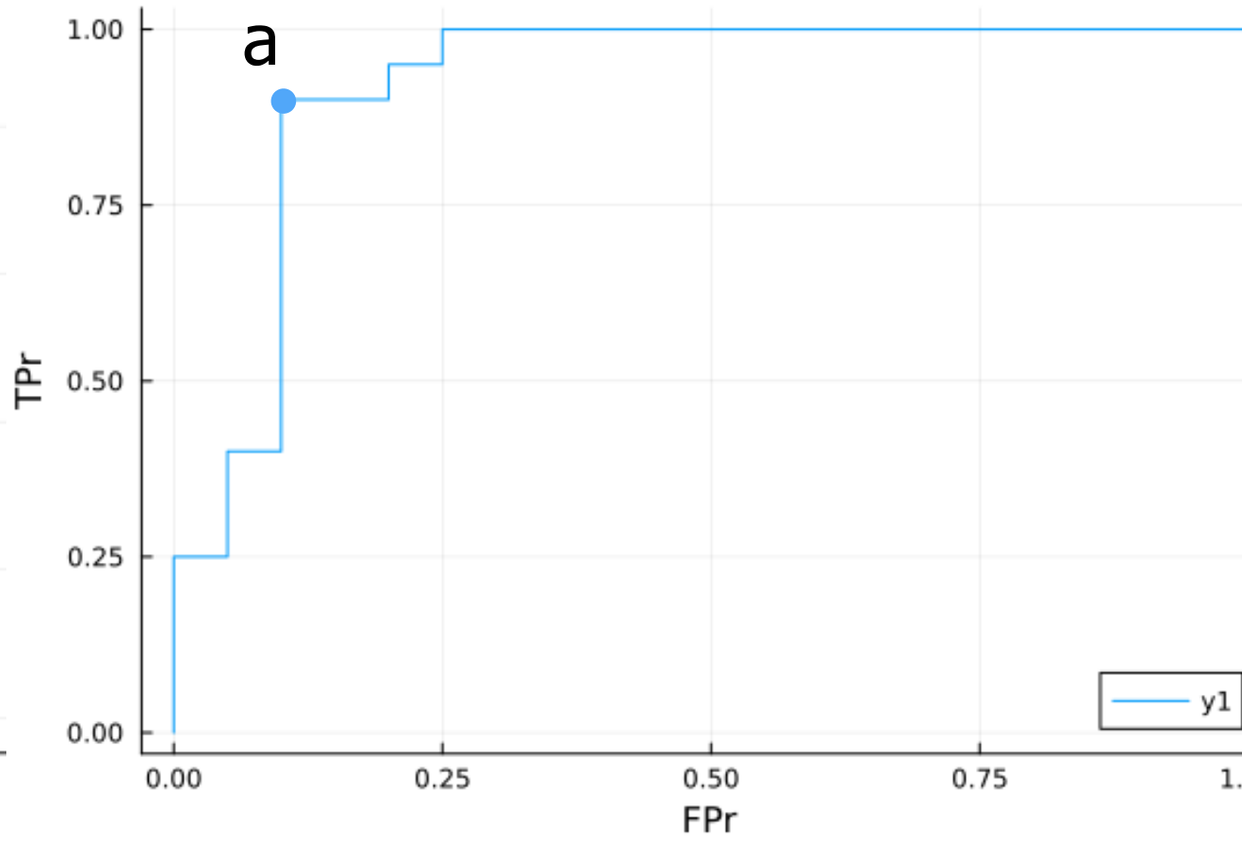


Example to compute ROC curve



Operating points

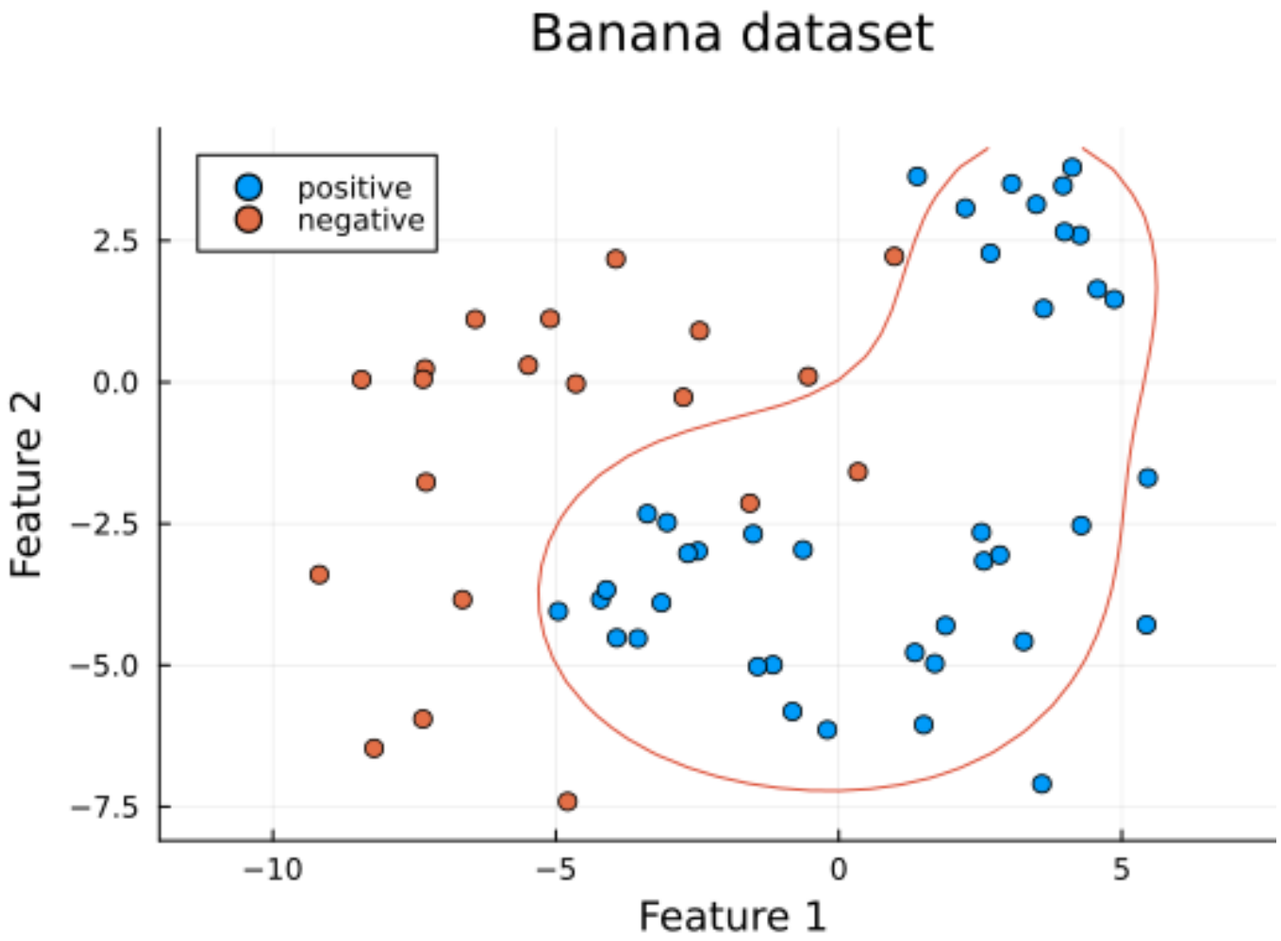


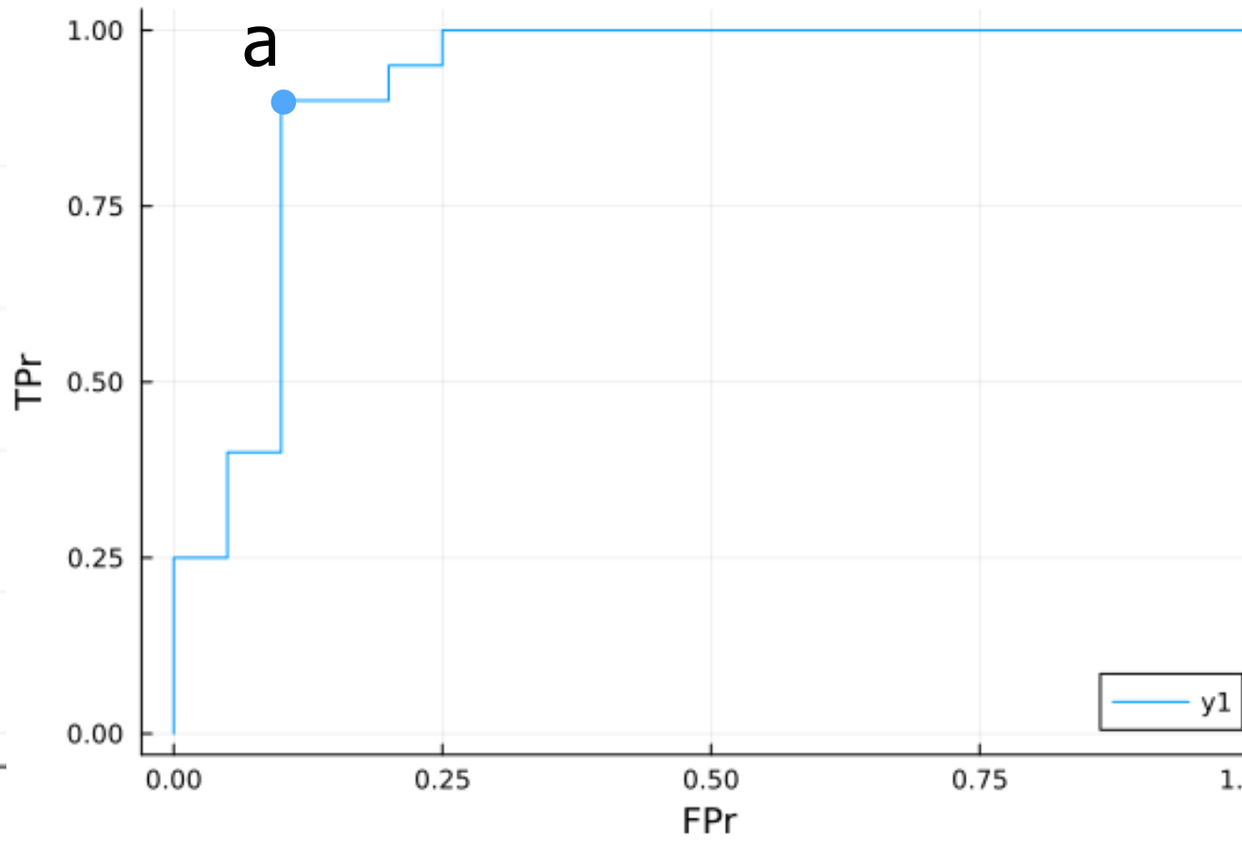


- Given for this dataset, where is operating point a?

TU Delft

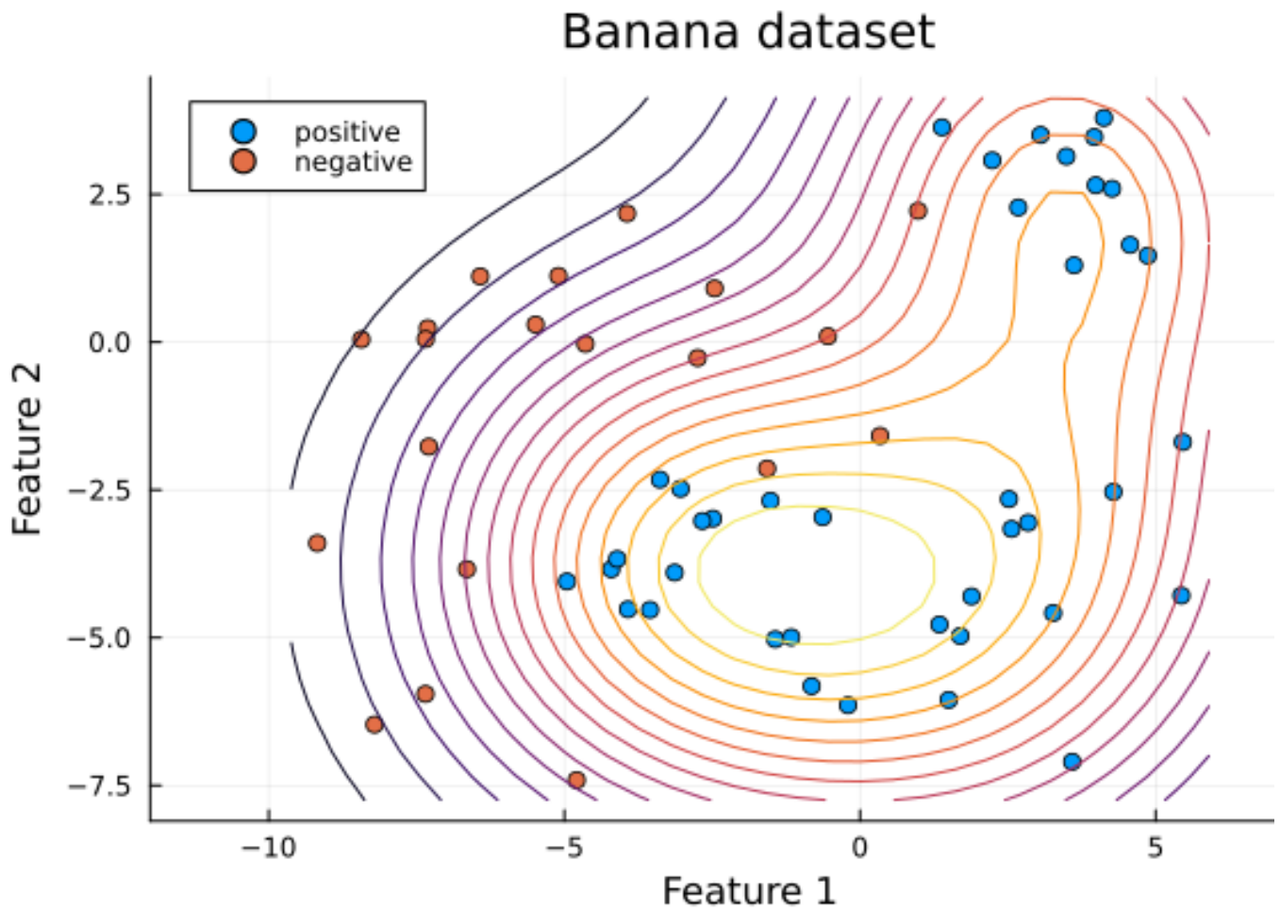
Operating points

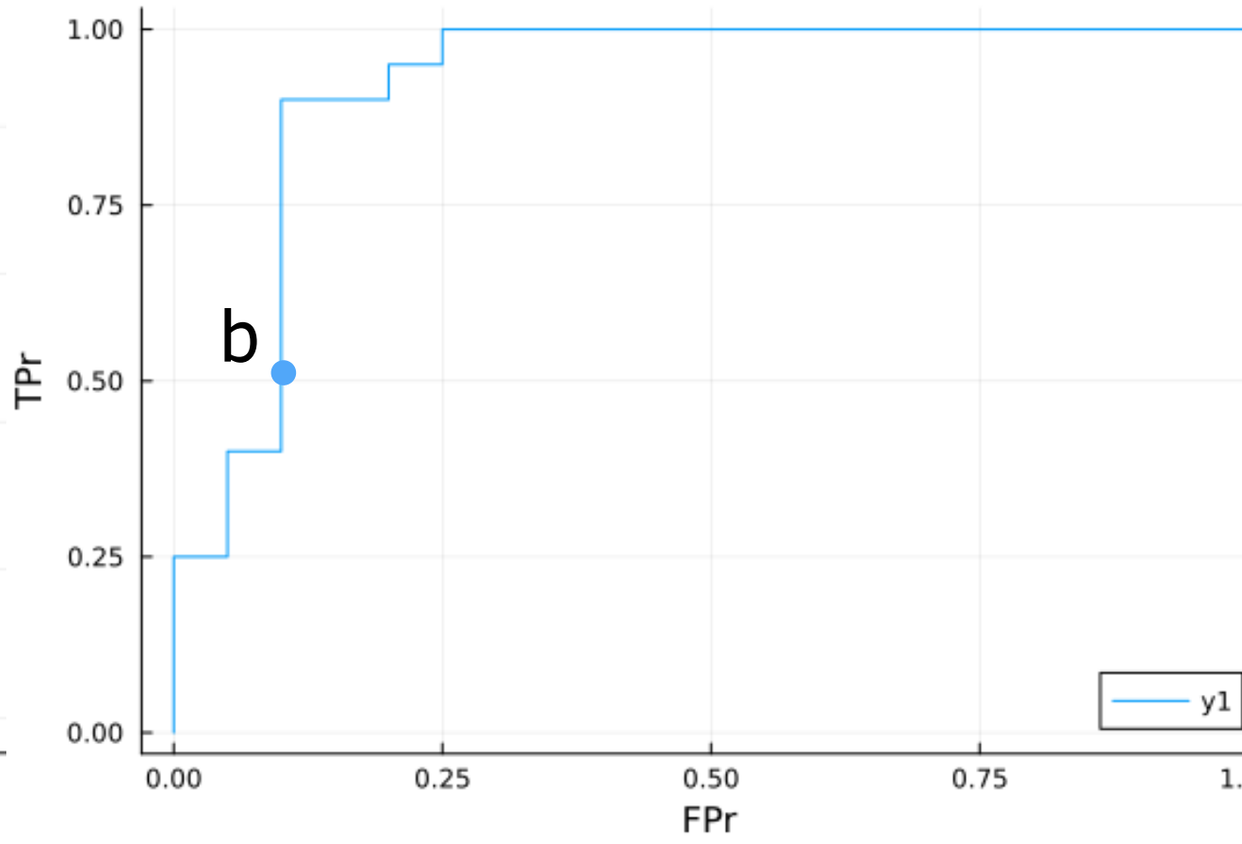




- Given for this dataset, where is operating point a?

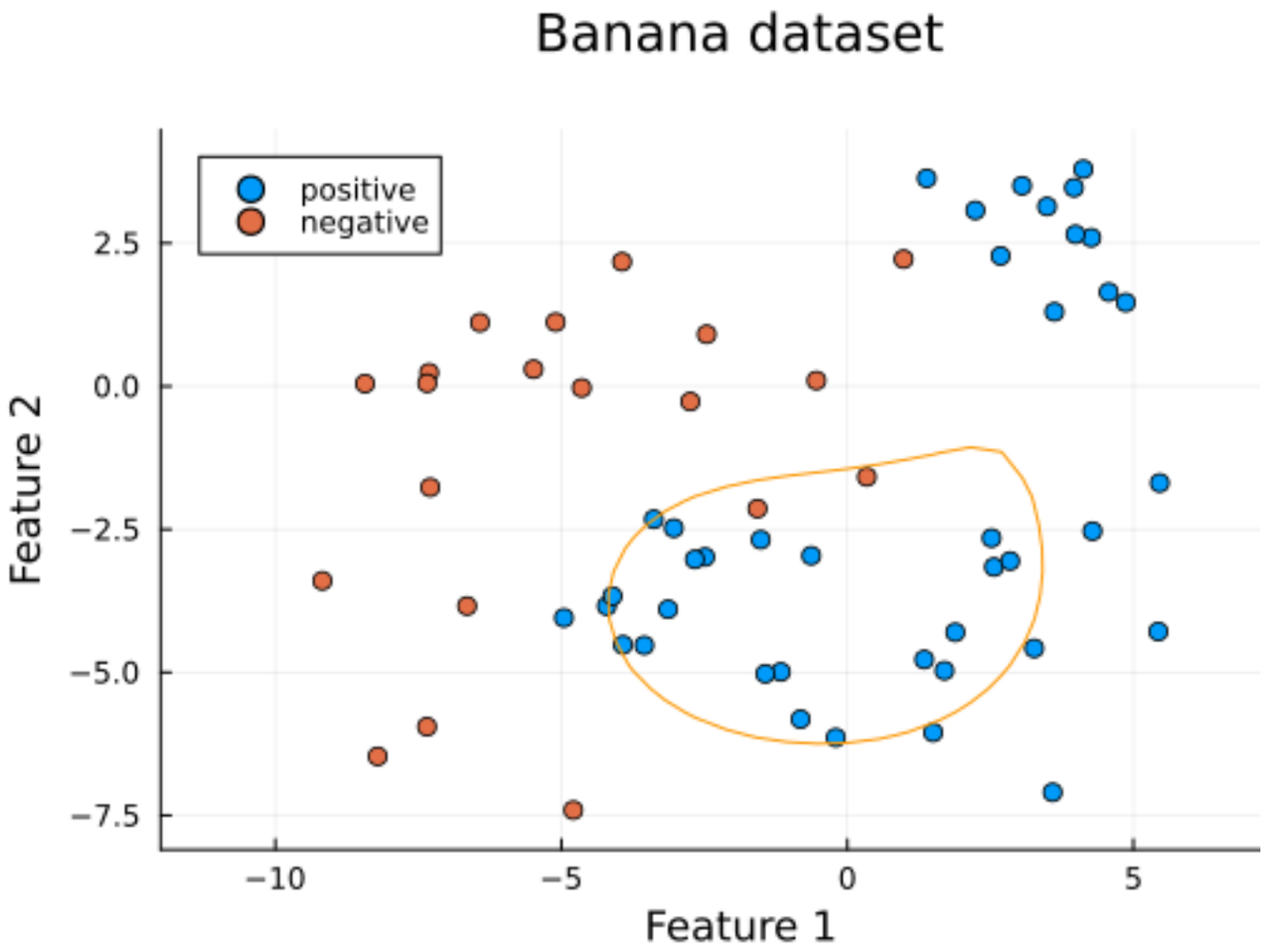
Operating points

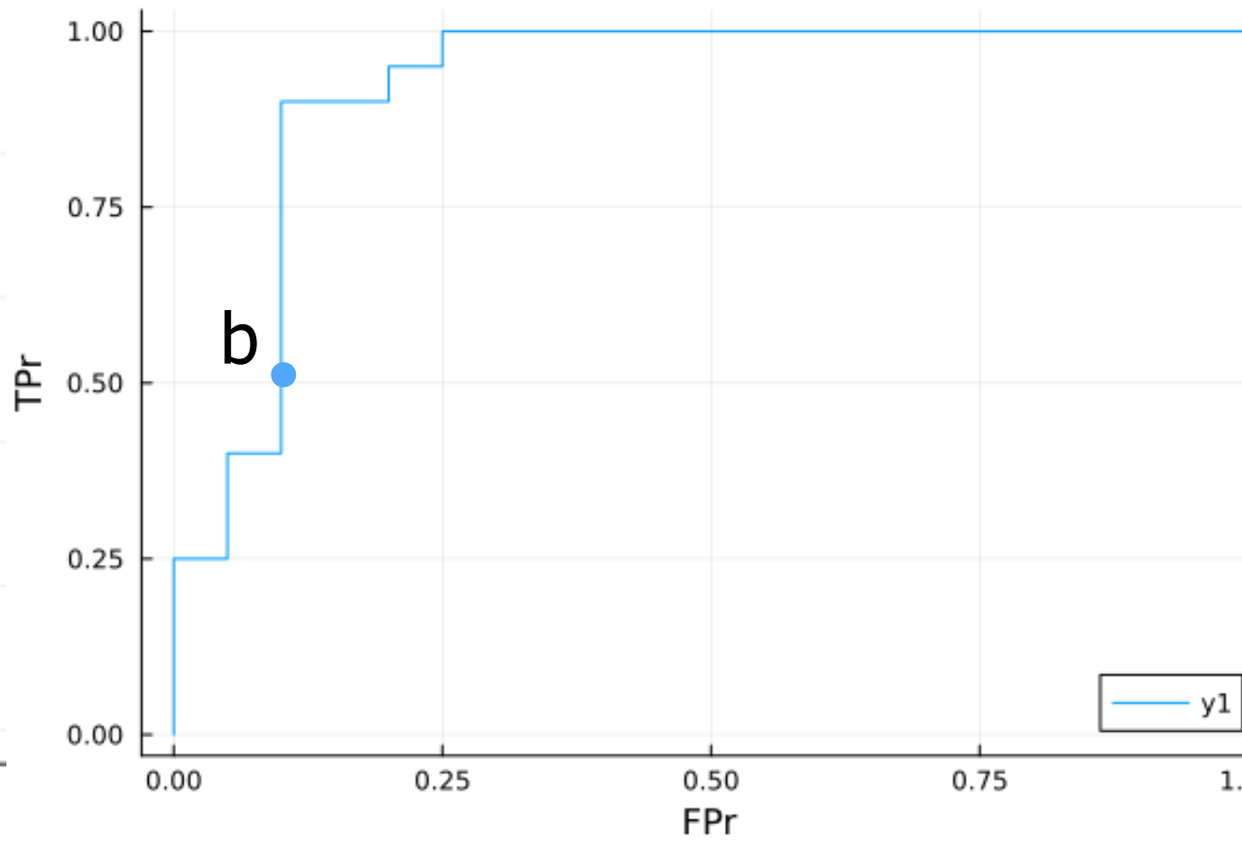




- Given for this dataset, where is operating point b?

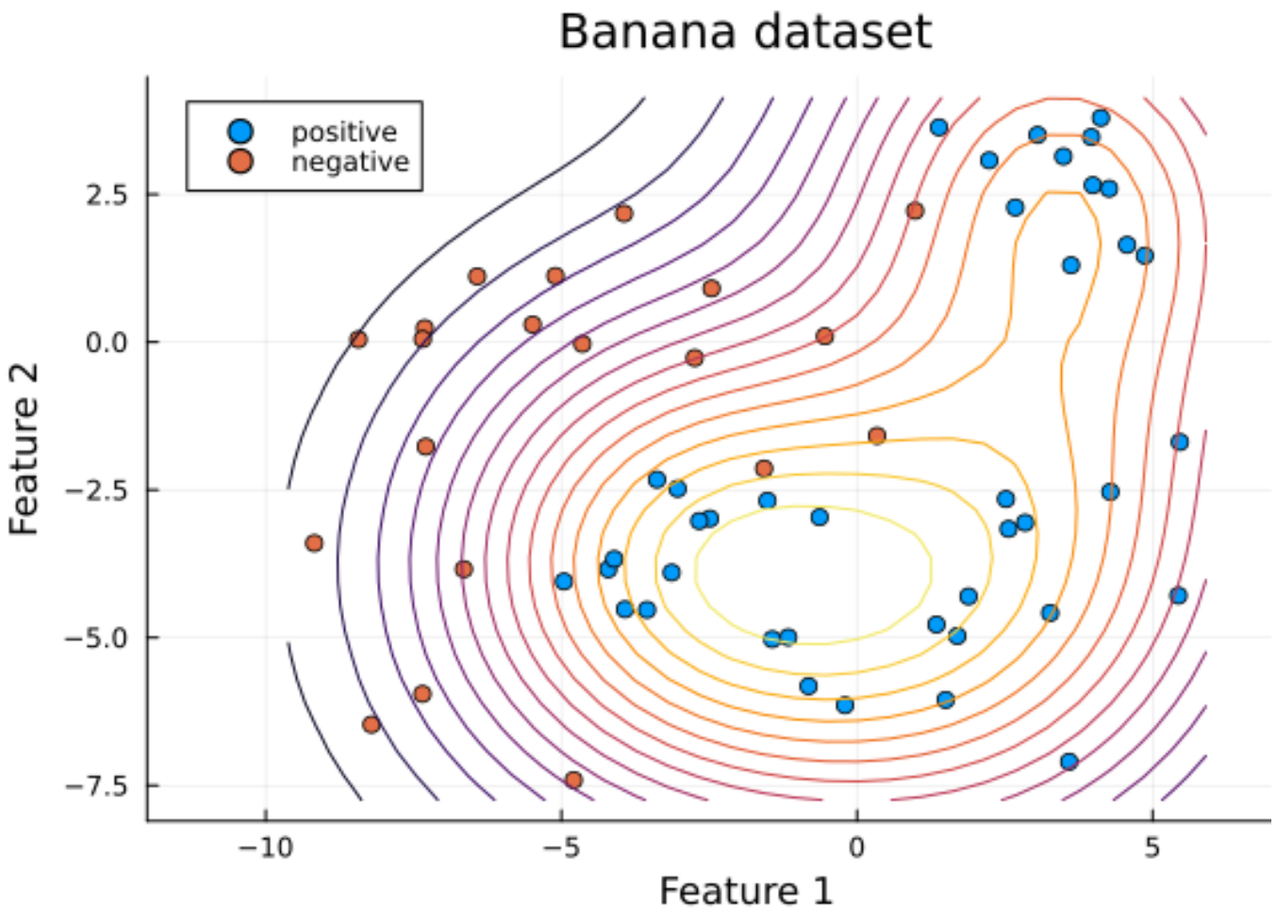
Operating points

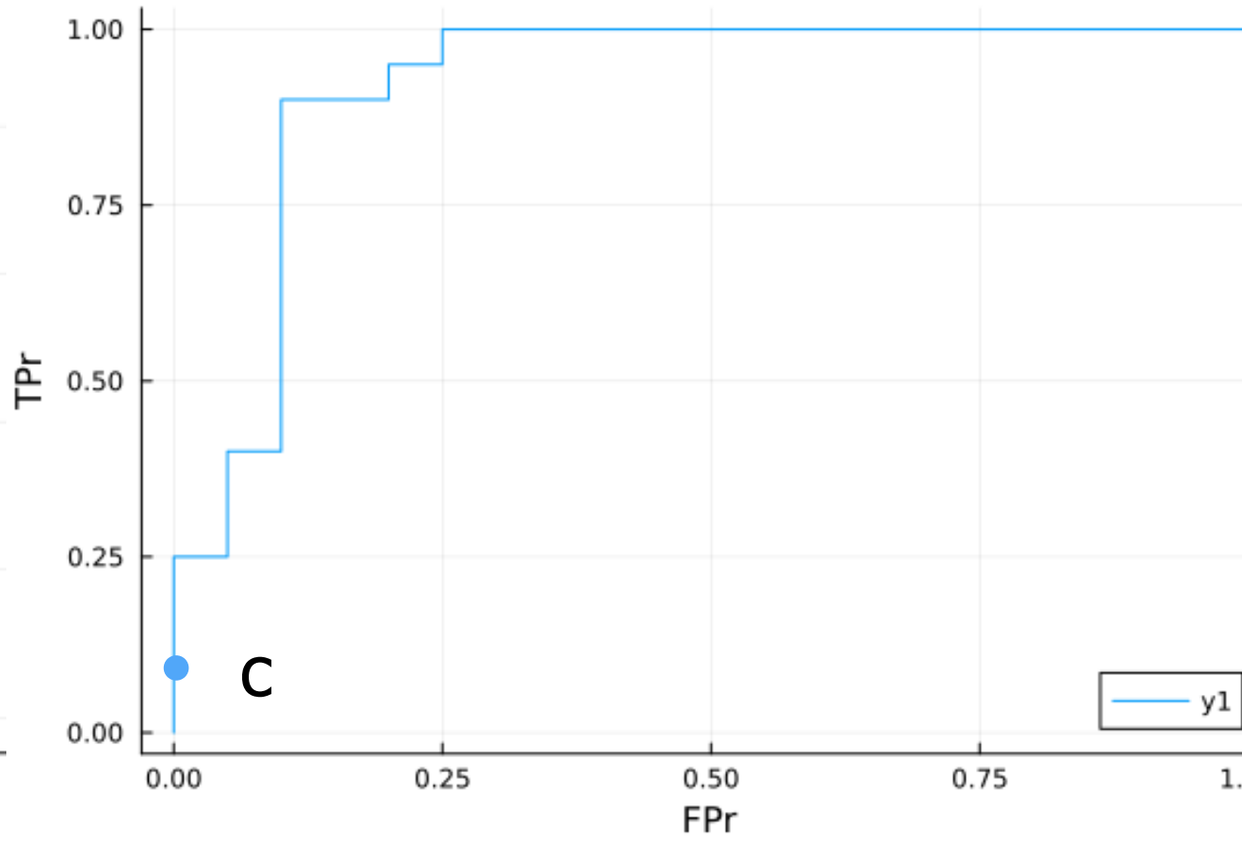




- Given for this dataset, where is operating point b?

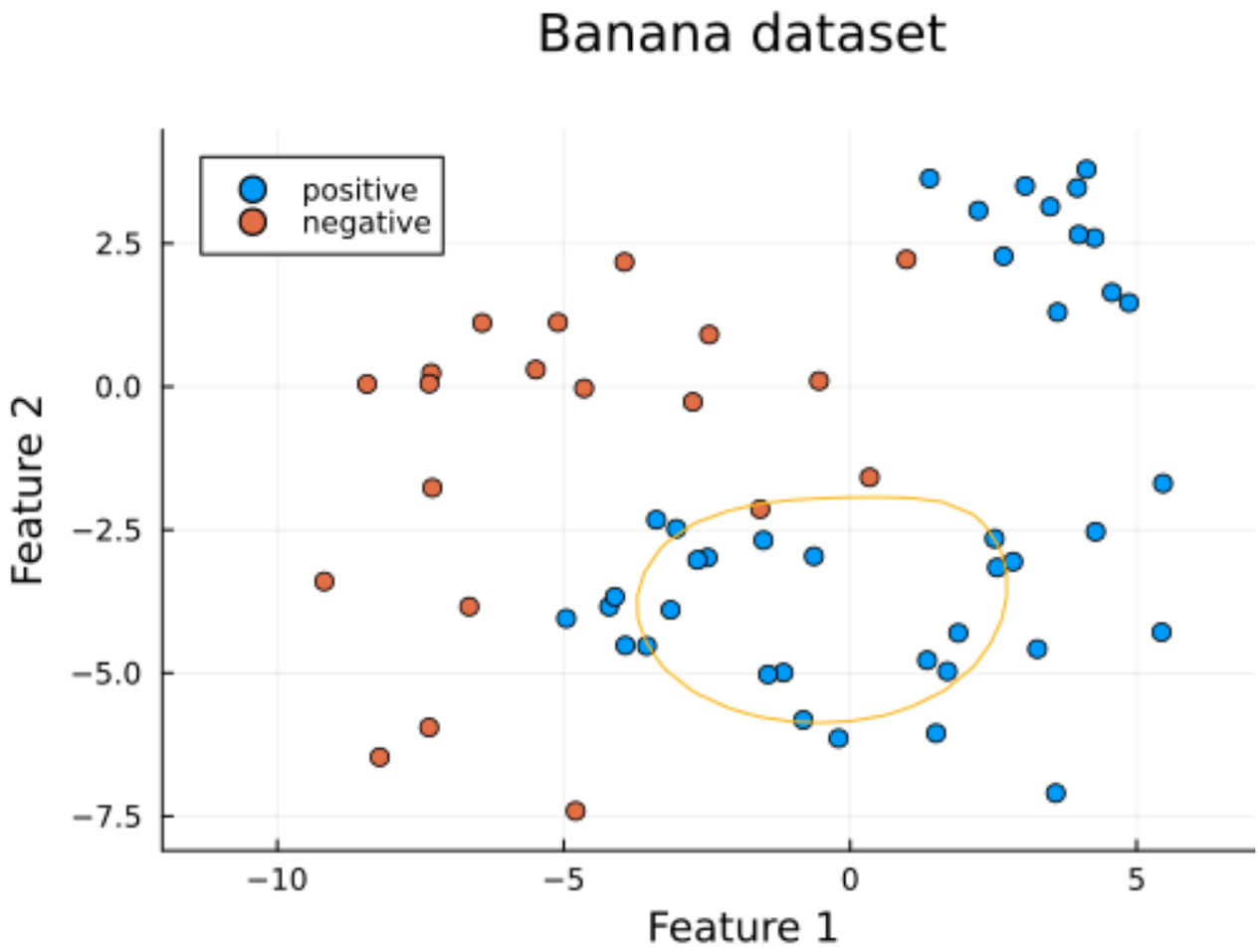
Operating points

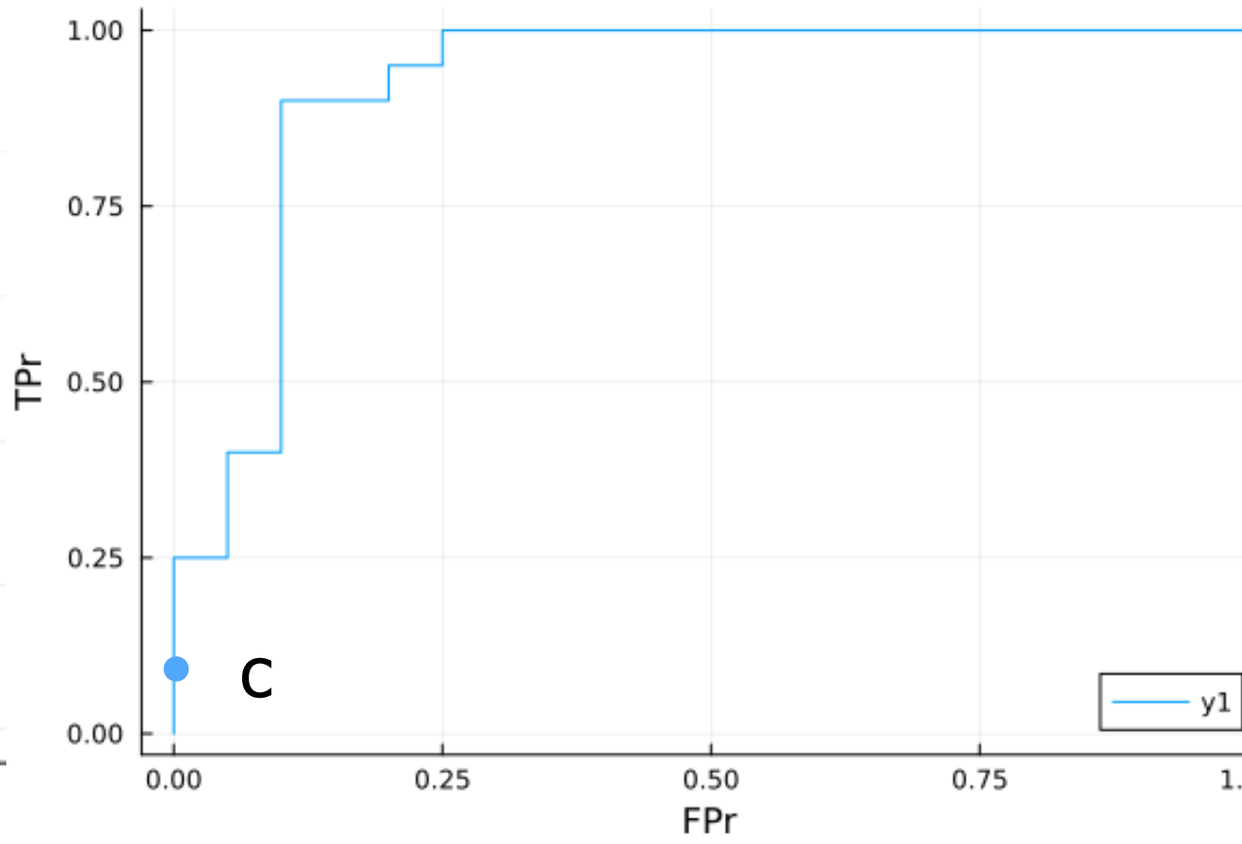




- Given for this dataset, where is operating point c?

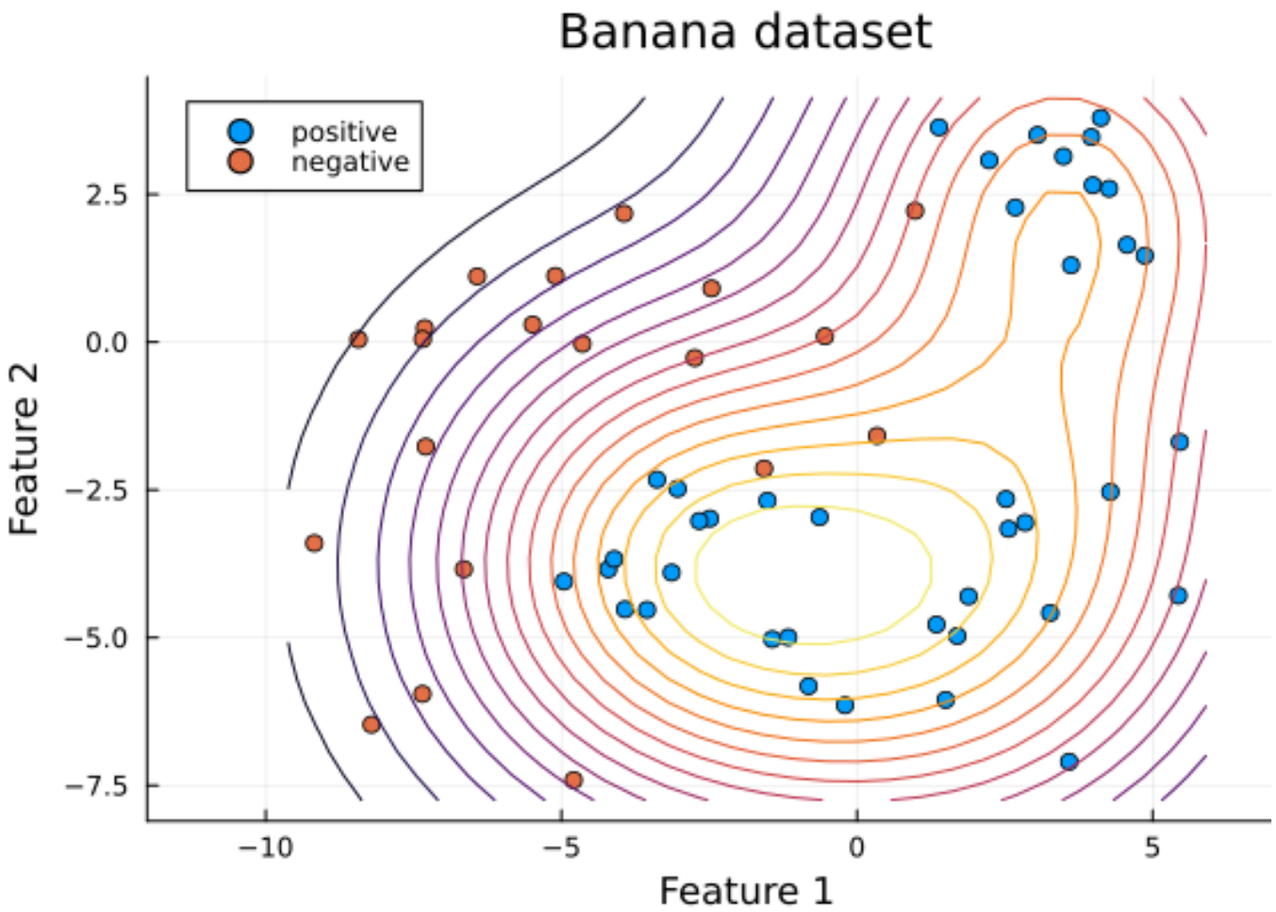
Operating points

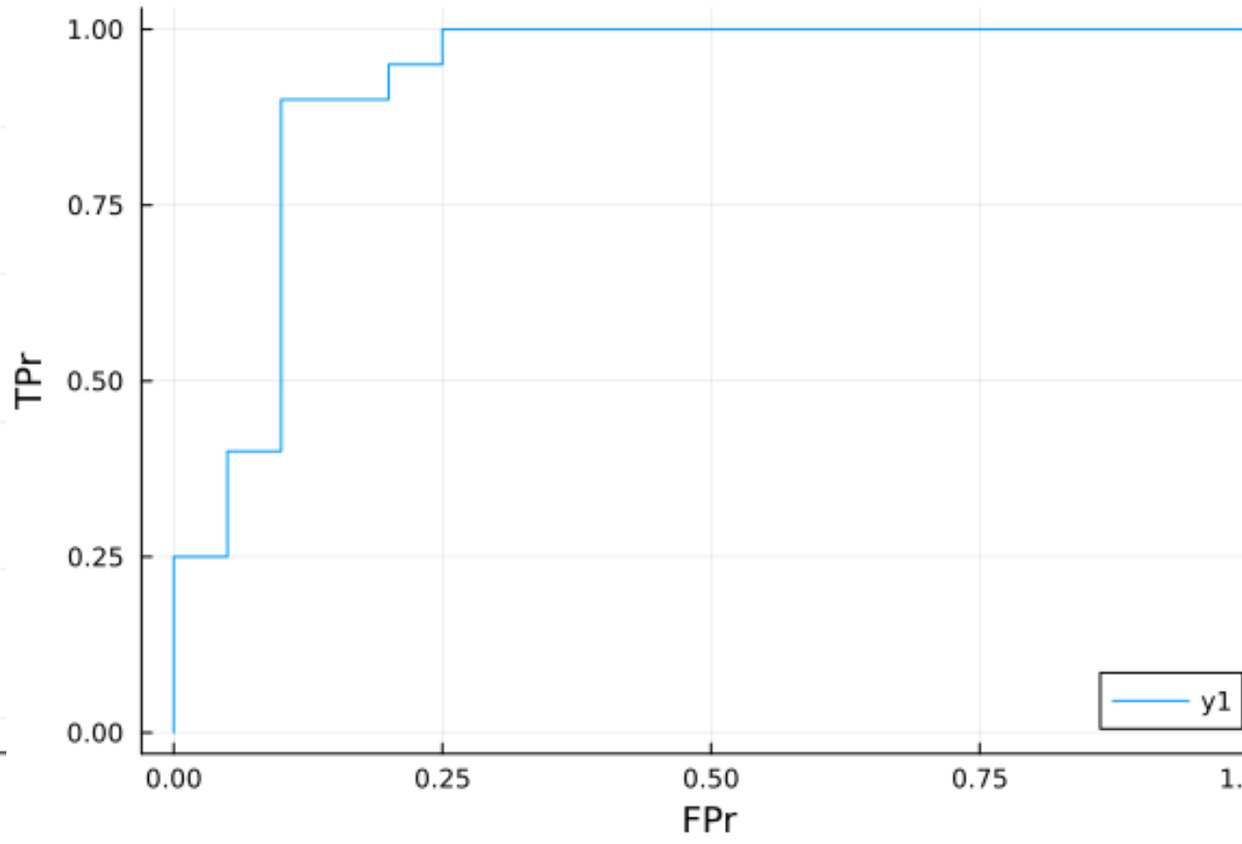




- Given for this dataset, where is operating point c?

Area under the ROC curve

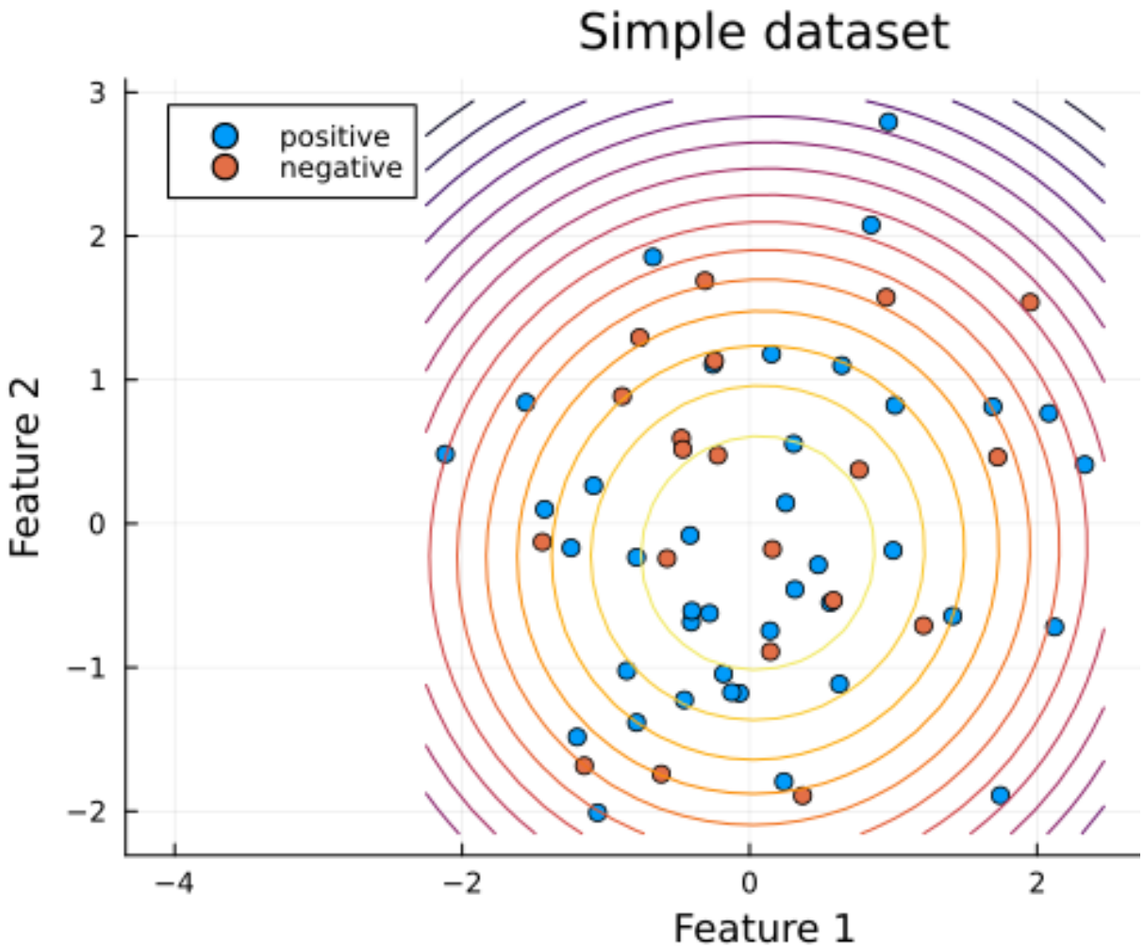


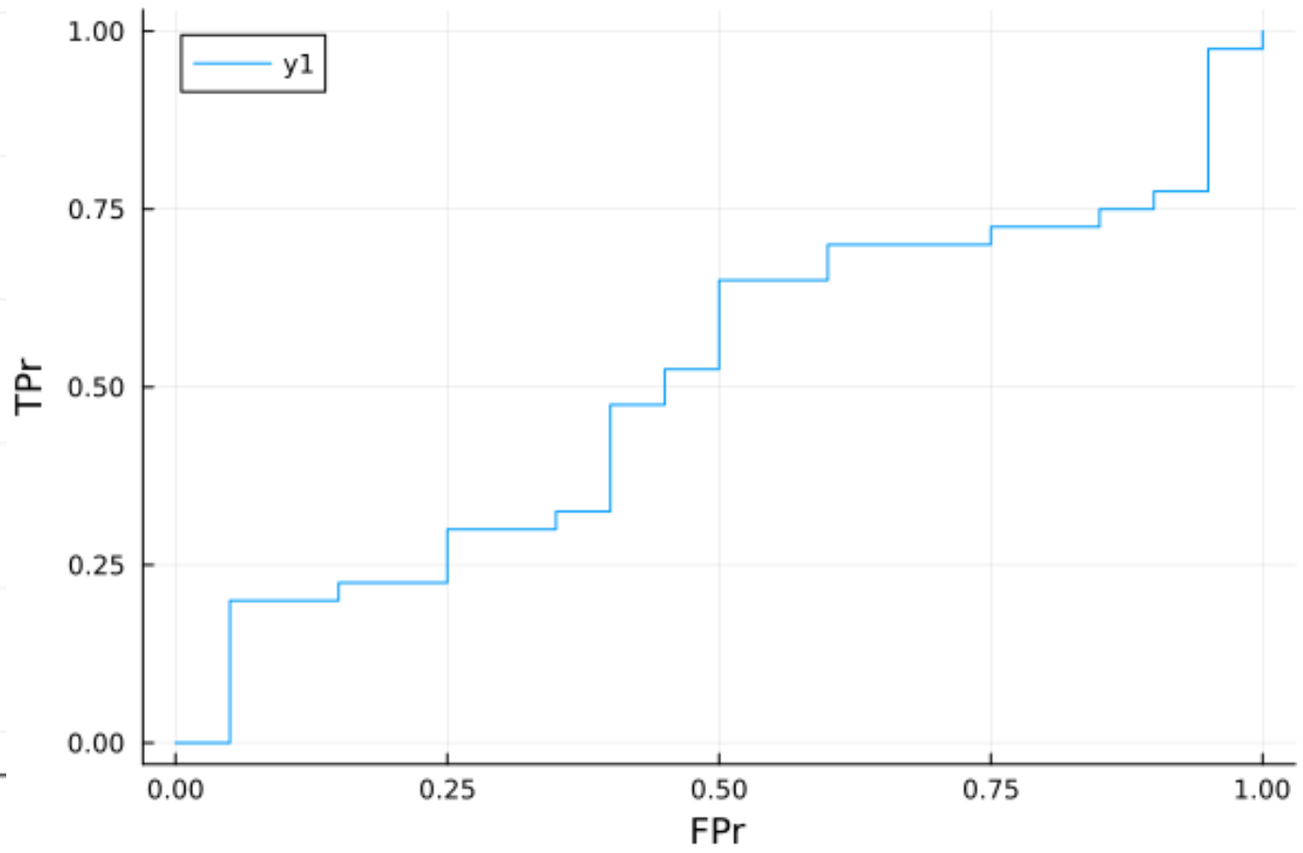


- Area under the ROC curve, AUC or AUROC, standard performance measure
- Not sensitive to class imbalance in test set

TU Delft

Area under the ROC curve

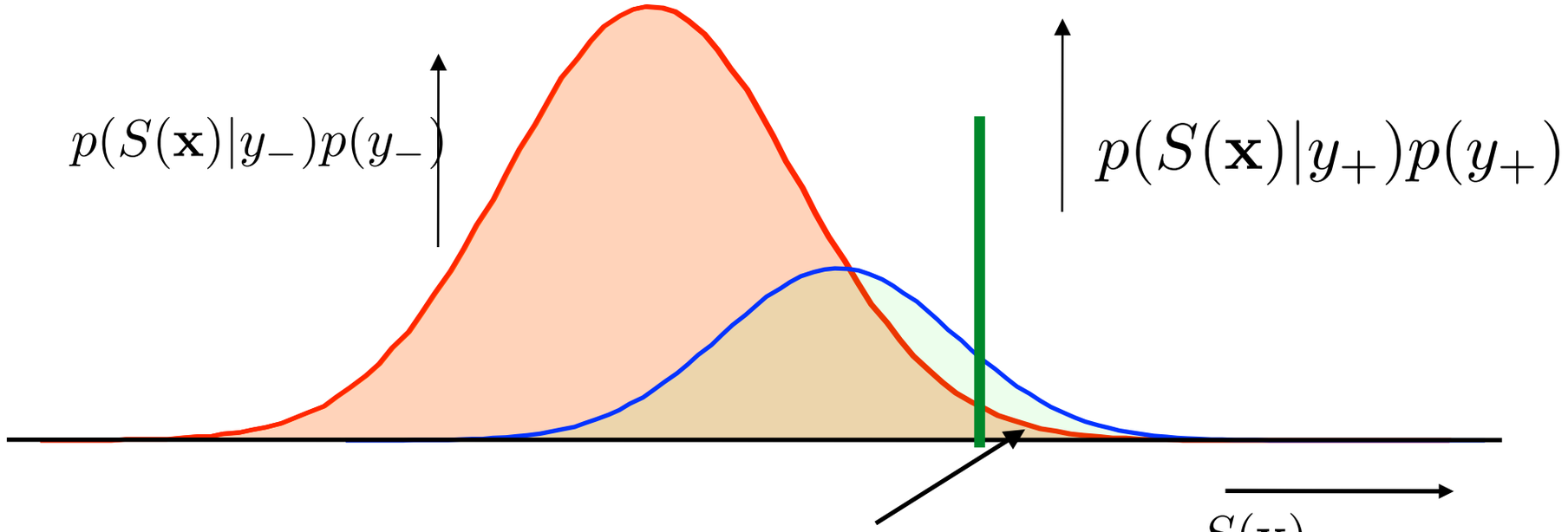




- For fully overlapping dataset, and very poorly performing classifiers, the Area under the ROC curve becomes 0.5

Precision-recall Curve

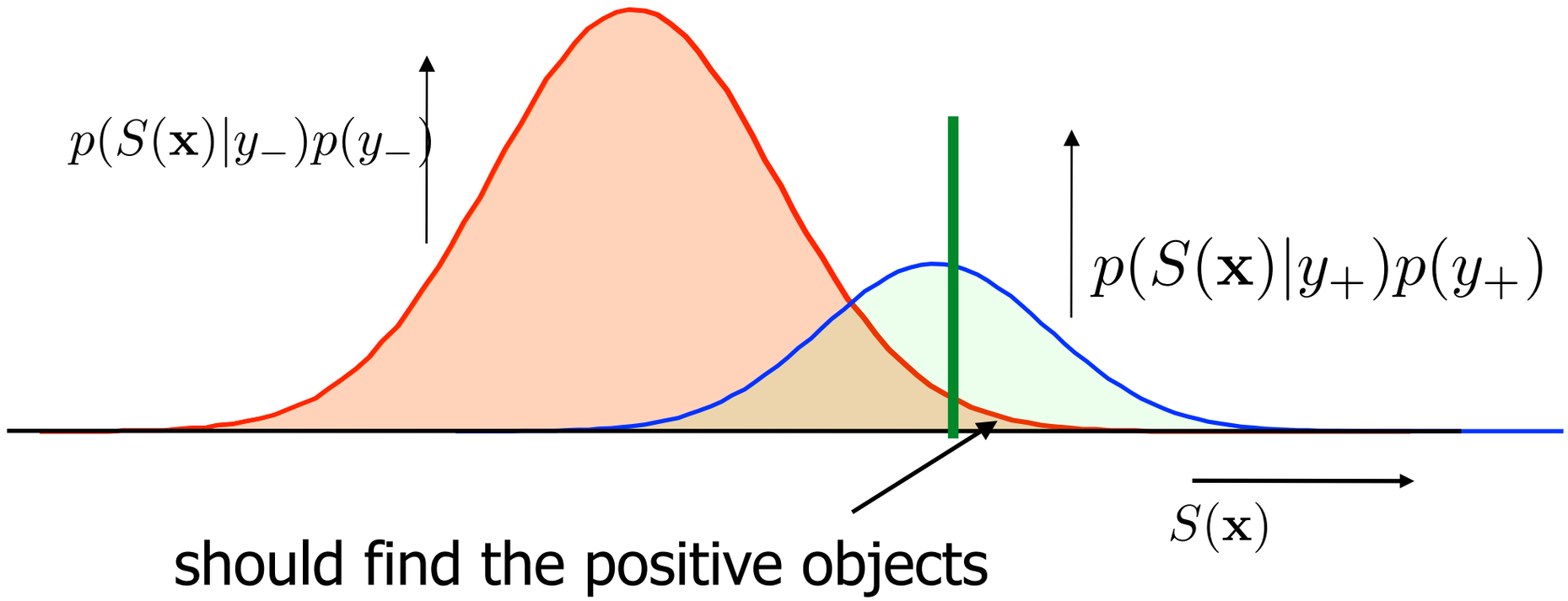
- Focus more on the positive class: encourage that if a classifiers an object is positive, it certainly is positive
- Used in retrieval tasks
- Not so sensitive to (size of) negative class



should find the positive objects

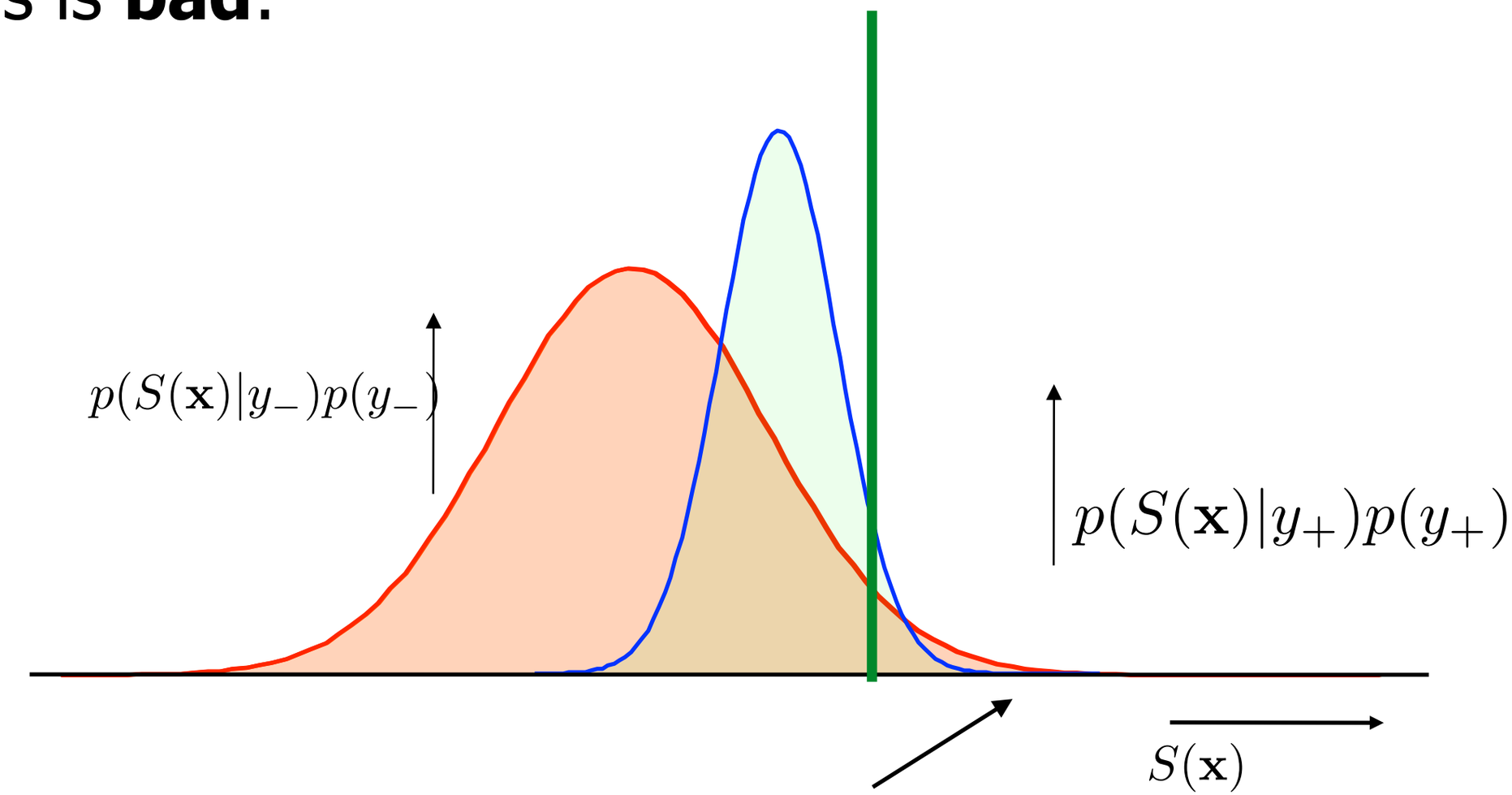
Precision-recall Curve

- Focus more on the positive class: encourage that if a classifiers an object is positive, it certainly is positive
- This is very **good**:



Precision-recall Curve

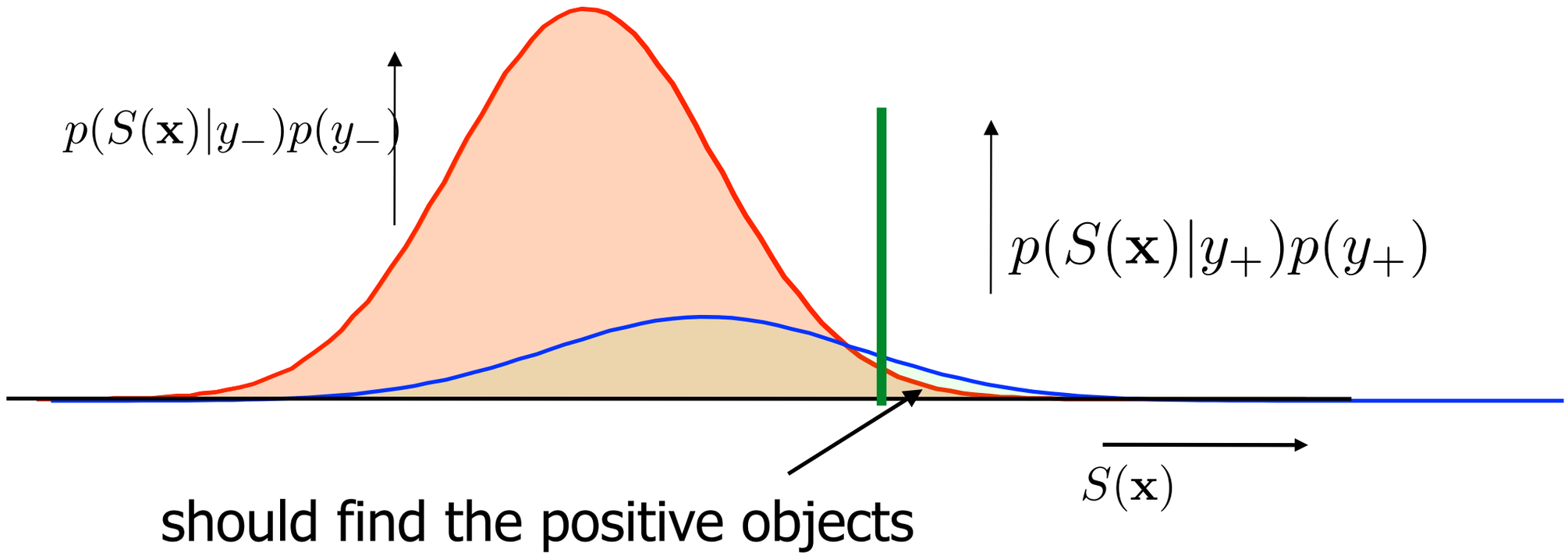
- Focus more on the positive class: encourage that if a classifiers an object is positive, it certainly is positive
- This is **bad**:



negative objects with higher score than positives!

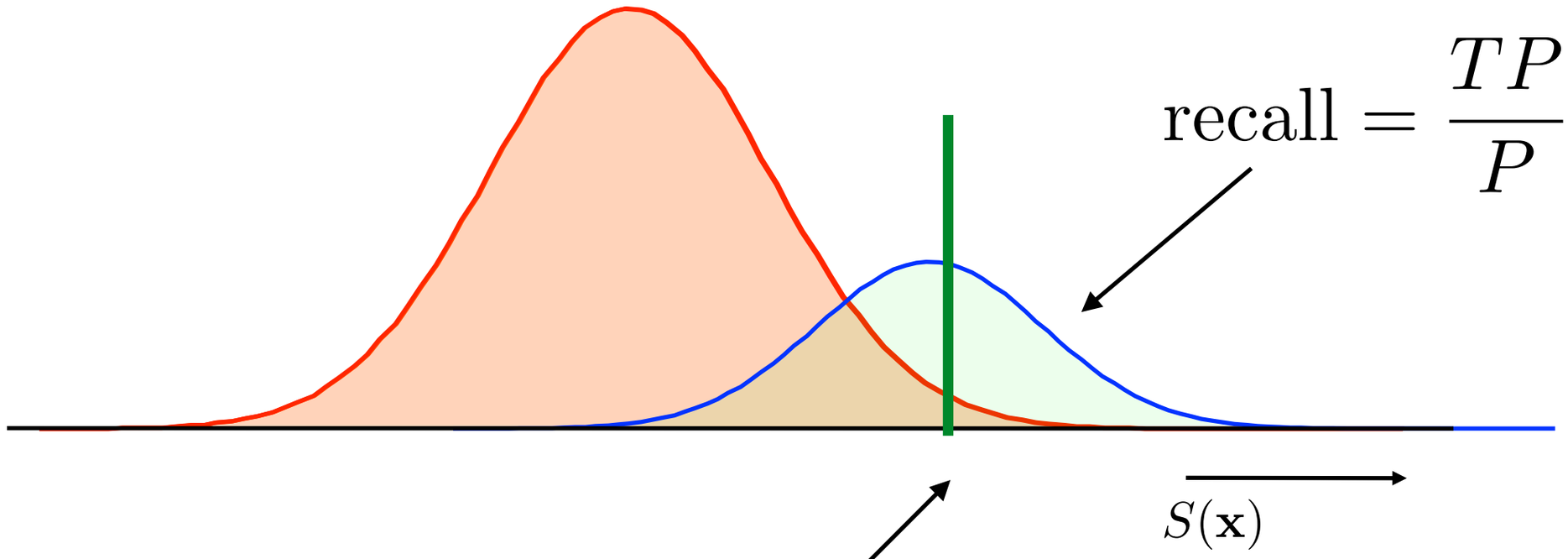
Precision-recall Curve

- Focus more on the positive class: encourage that if a classifiers an object is positive, it certainly is positive
- This is also ok:



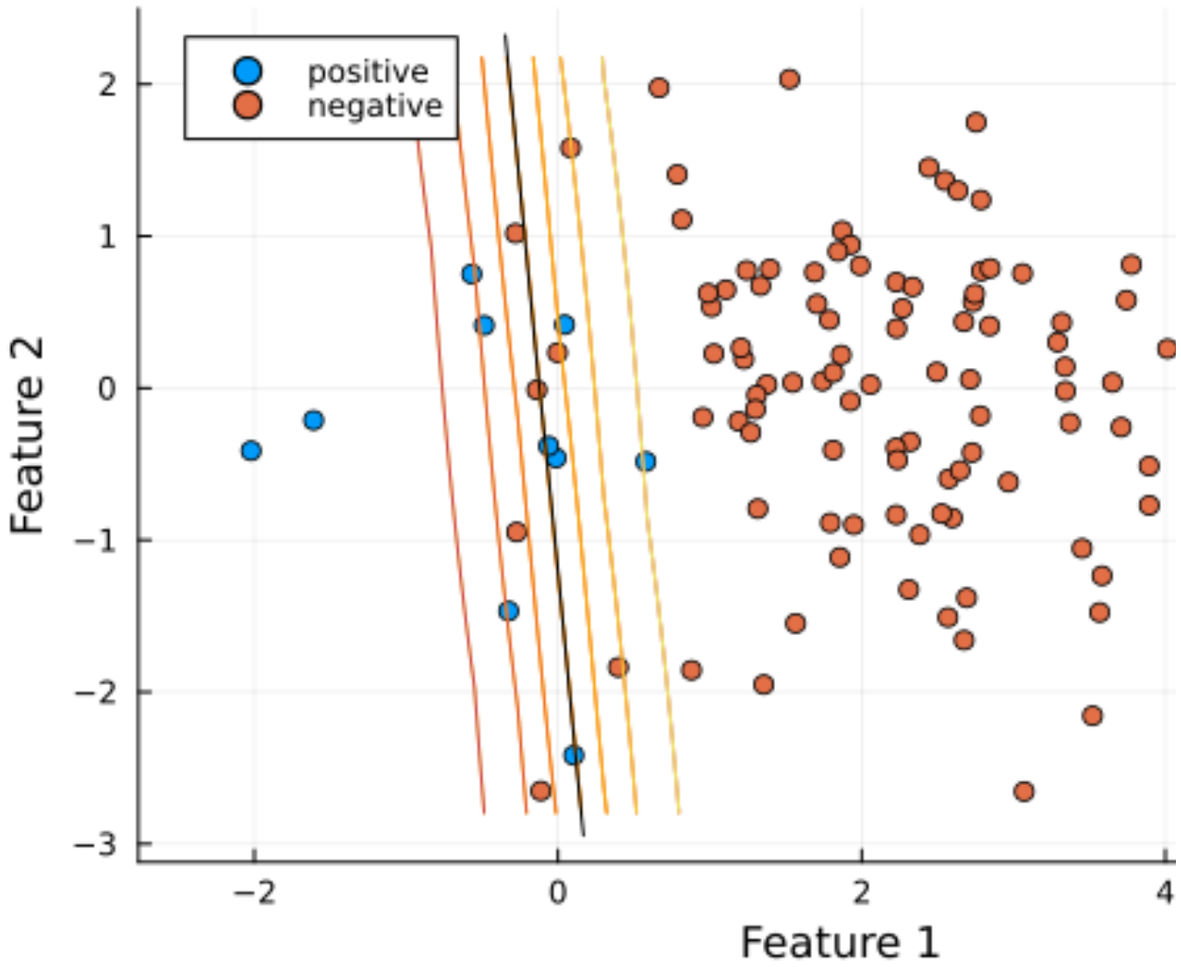
Precision-recall Curve

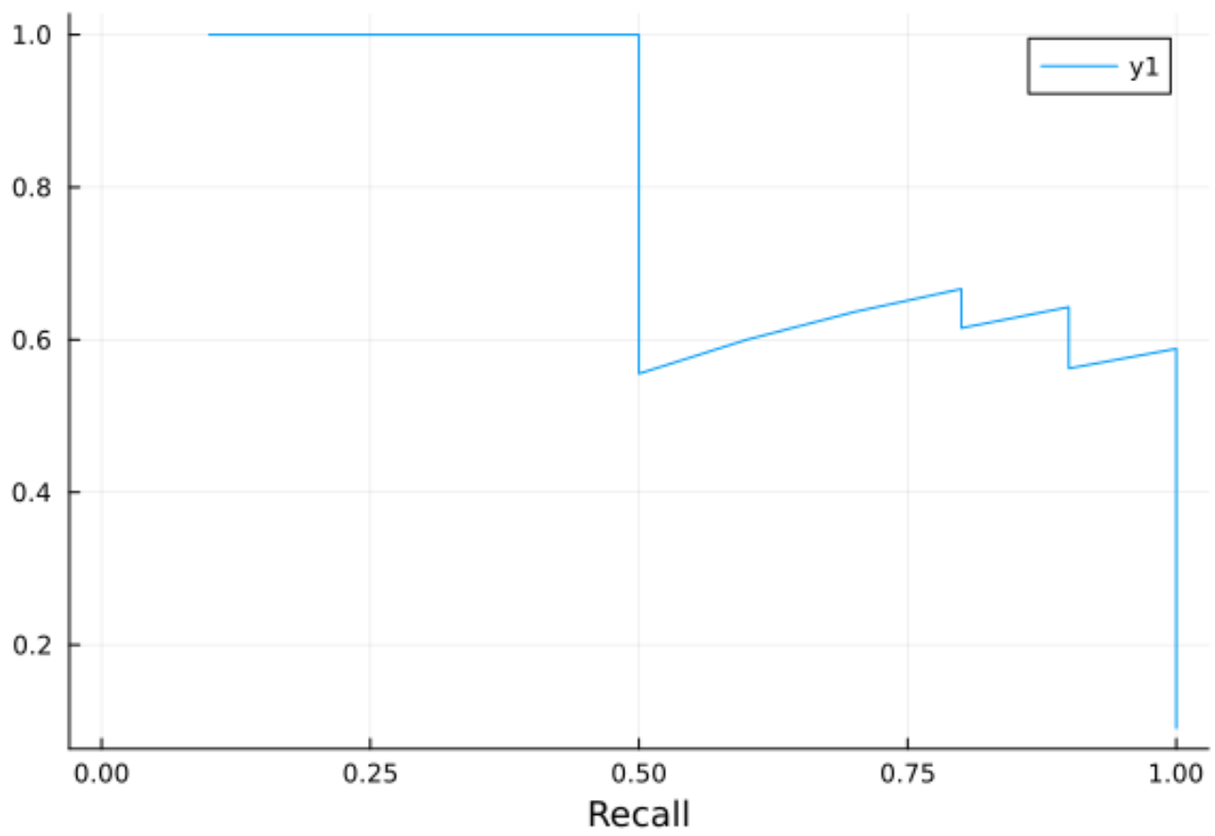
- Focus more on the positive class: encourage that if a classifiers an object is positive, it certainly is positive
- This is very good:



Precision-recall curve

- Precision-recall is a non-monotonic curve:



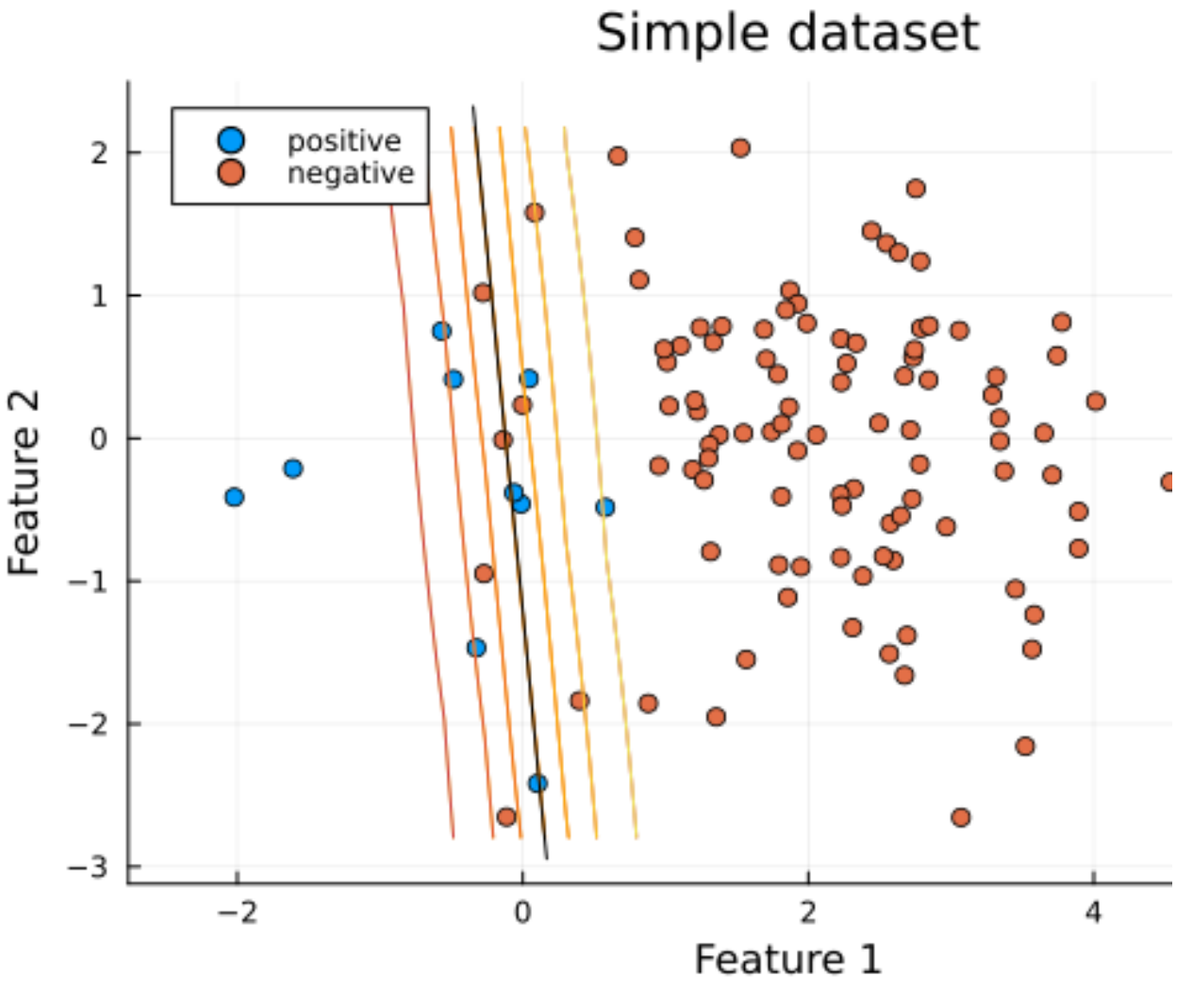


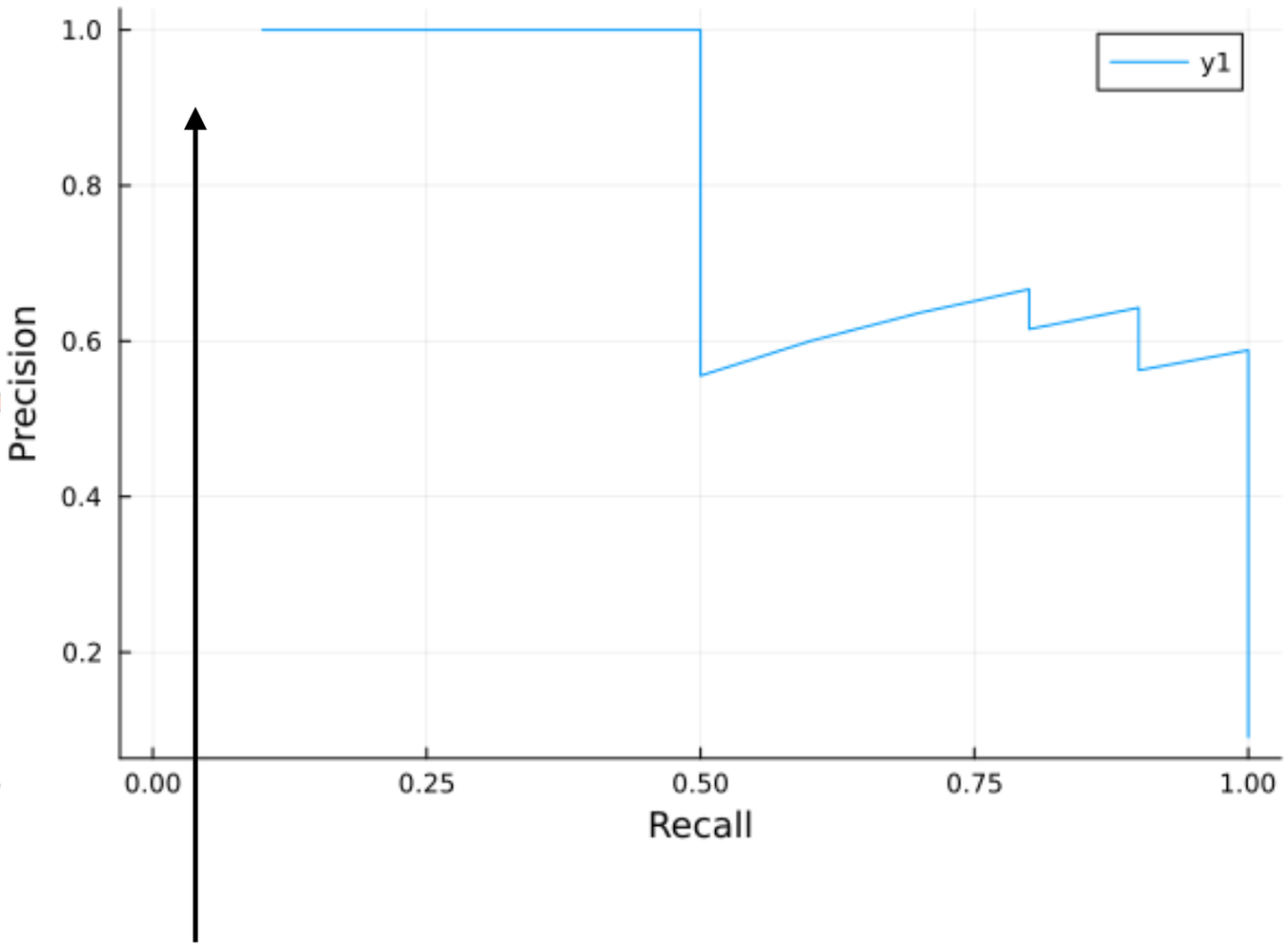
$$n_+ = 10$$

$$n_- = 100$$

Precision-recall curve

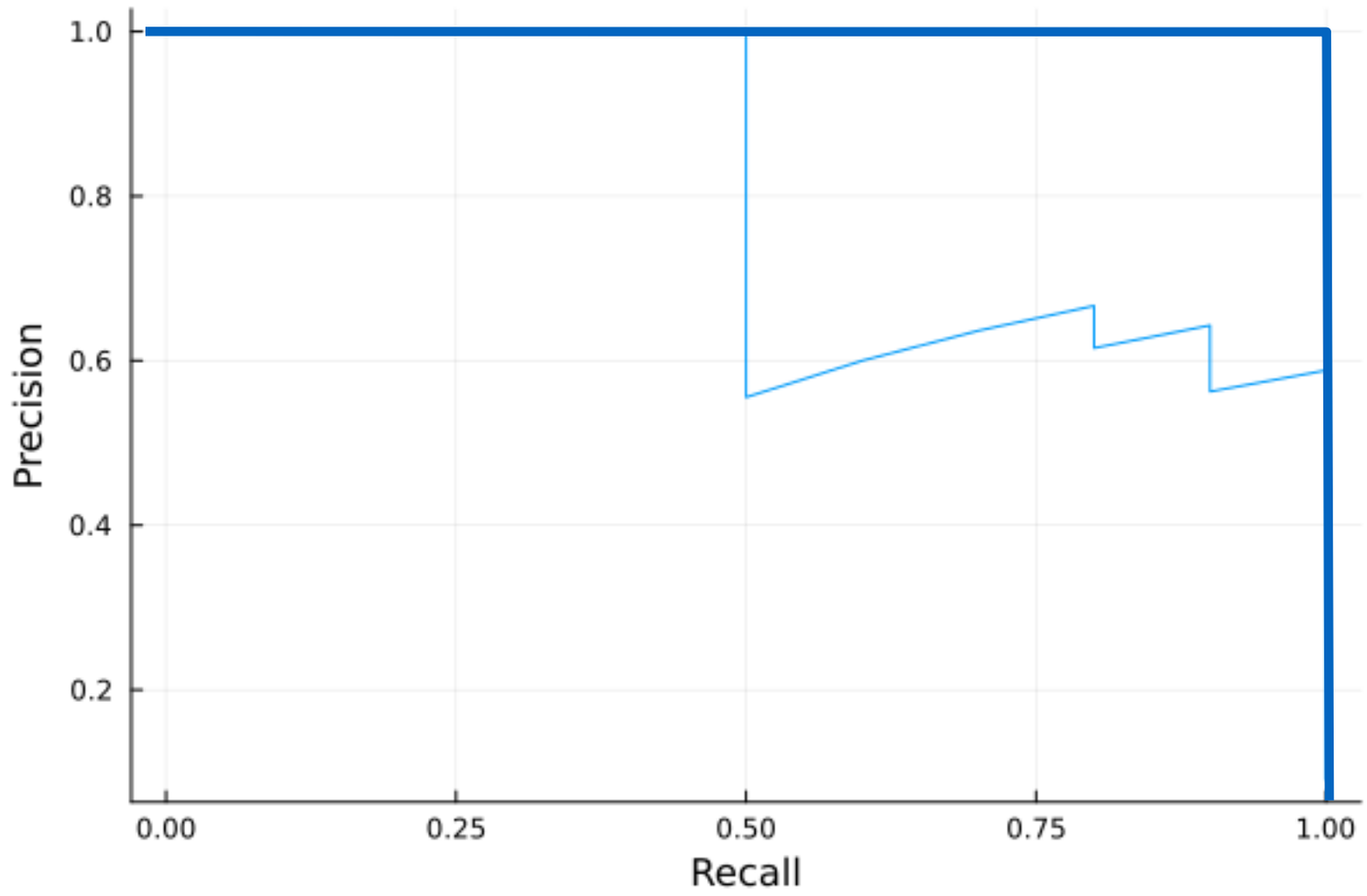
- Precision-recall is a non-monotonic curve:





what is happening here?

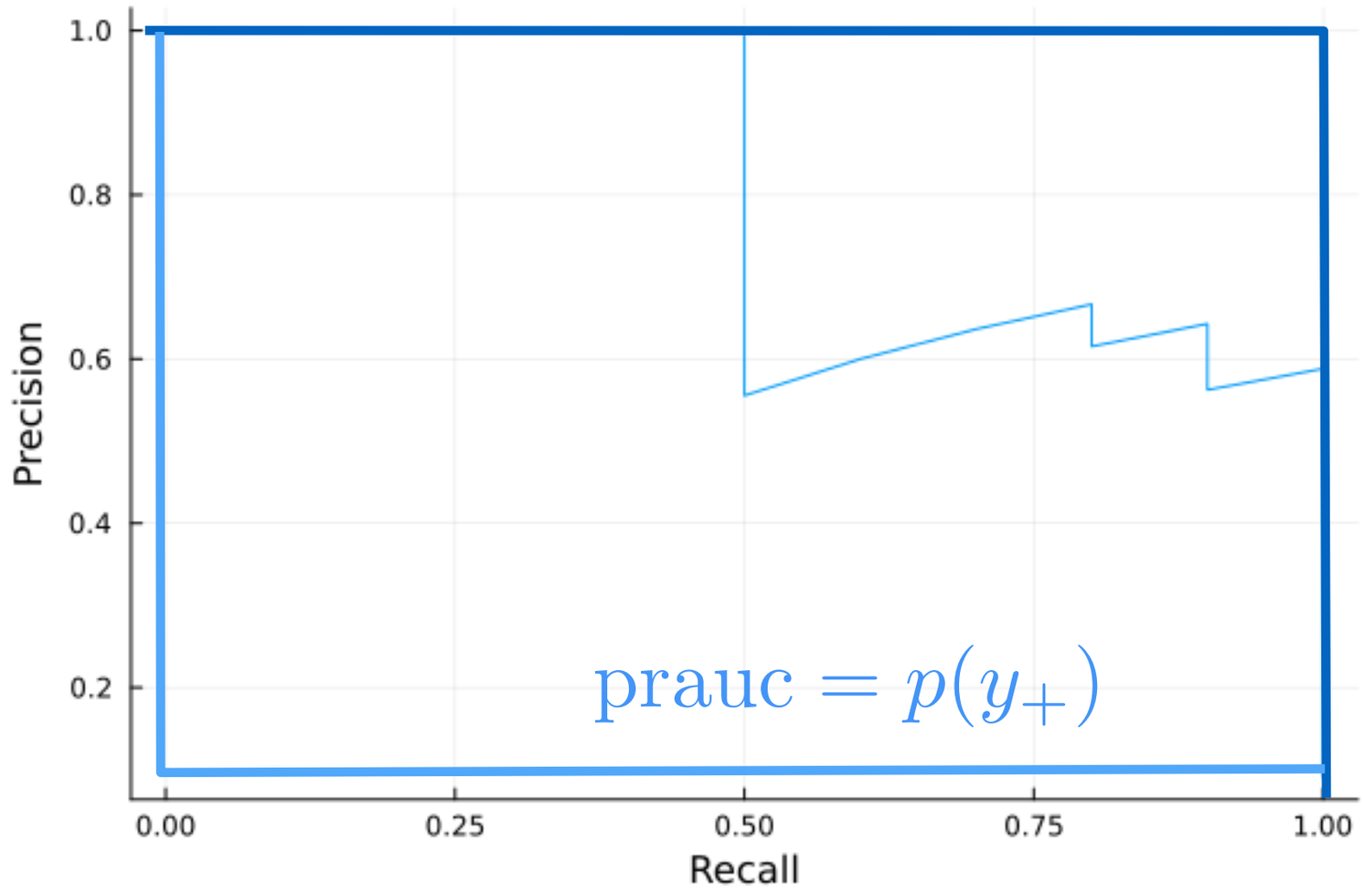
Area under Precision-recall curve



$\text{prauc} = 1$

- Summary of PR curve: Area under PR curve: PRAUC
- For perfectly seperable data, PRAUC = 1

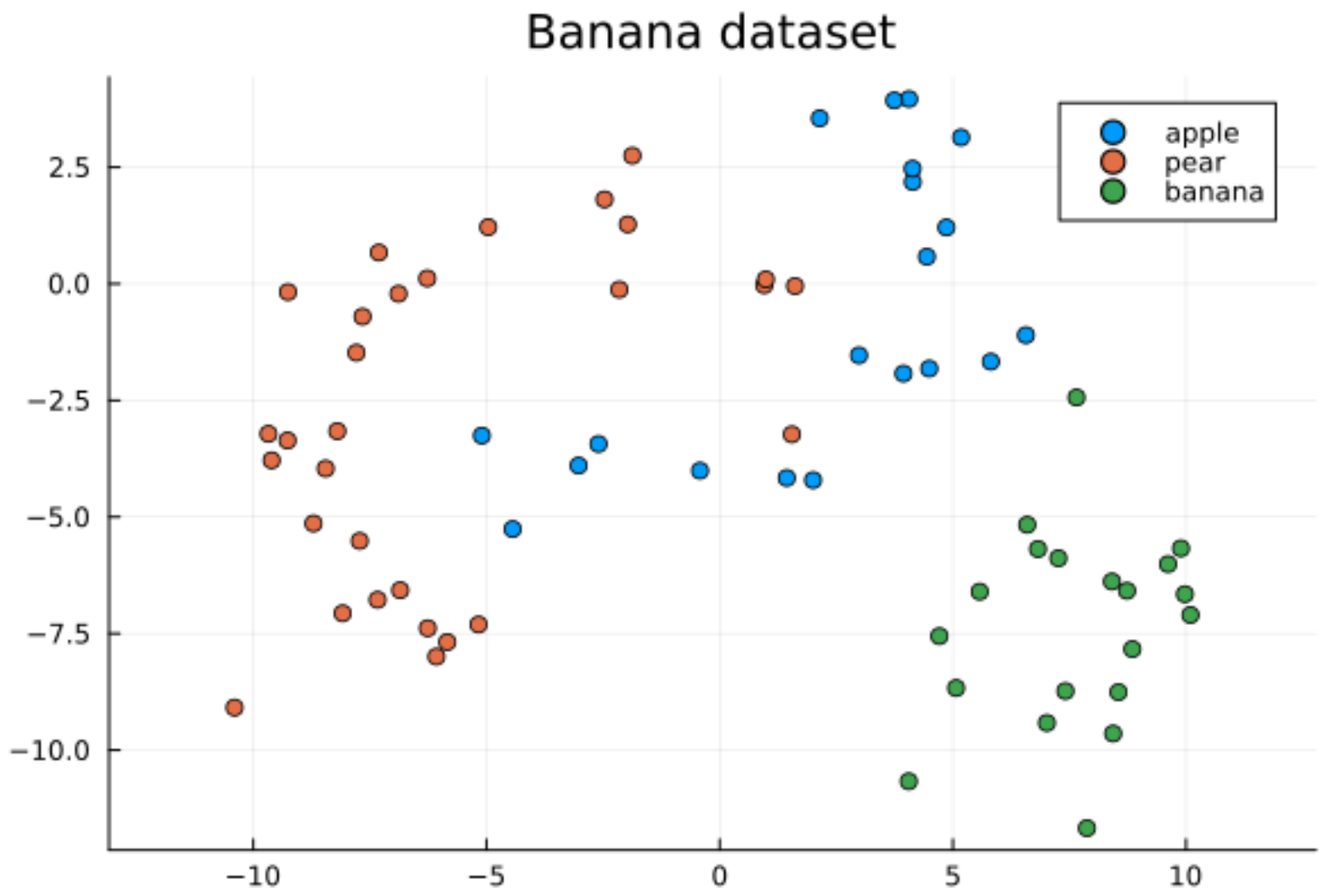
Area under Precision-recall curve



$\mathrm{prauc} = 1$

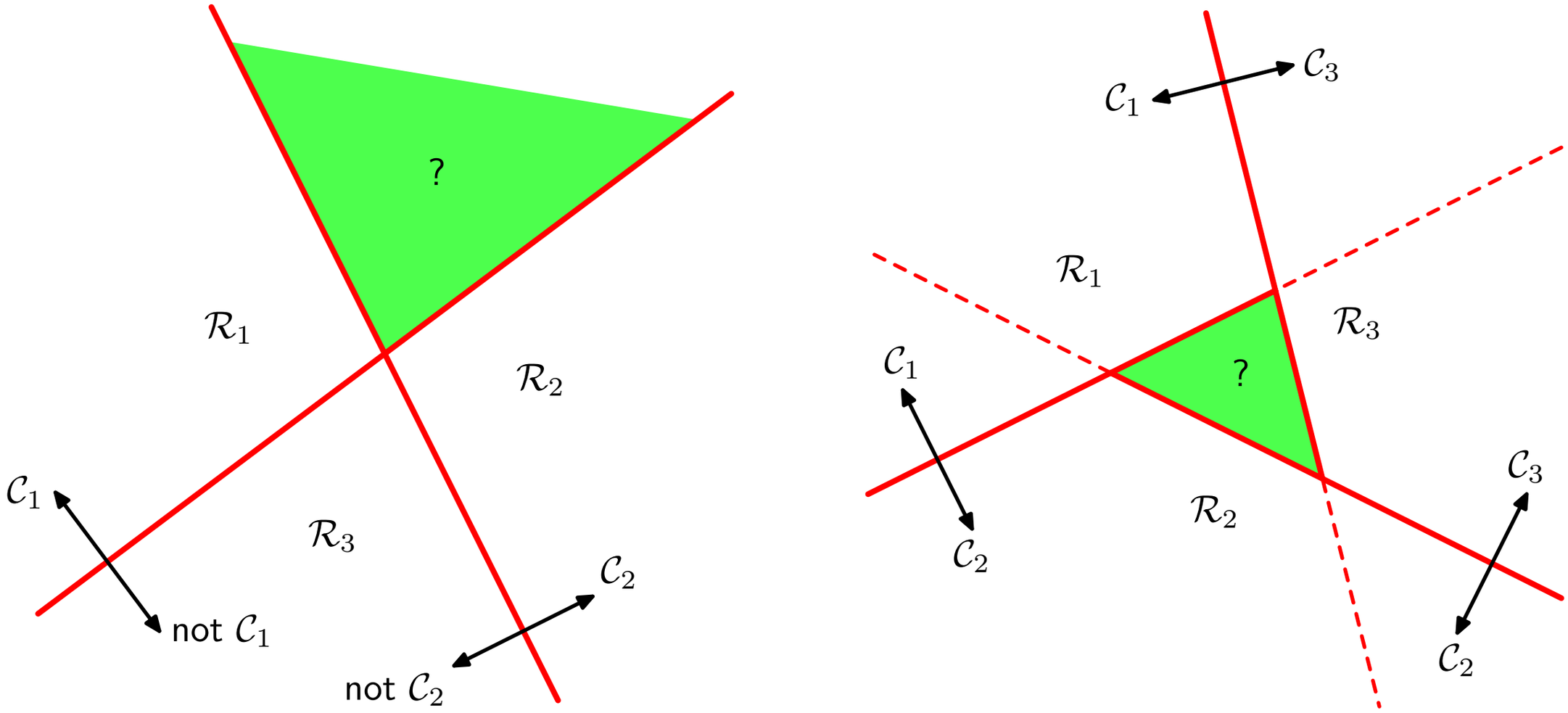
- Summary of PR curve: Area under PR curve: PRAUC
- For perfectly seperable data, PRAUC = 1
- For fully overlapping classes, PRAUC = $p(n_+)$

Multi-class classification



- No problem for generative models (why?)
- How to do it for Support Vector classifier? Perceptron? Fisher classifier?

From two-class to multi-class: voting



(Figure 4.1 Bishop)

- (left) One-vs.-rest, and (right) one-vs-one classifiers
- Both cause ambiguous regions

From two-class to multi-class

- In generative models we compare posterior probabilities:

$$\hat{y} = \underset{y}{\operatorname{argmax}}\ \hat{p}(y|\mathbf{x})$$

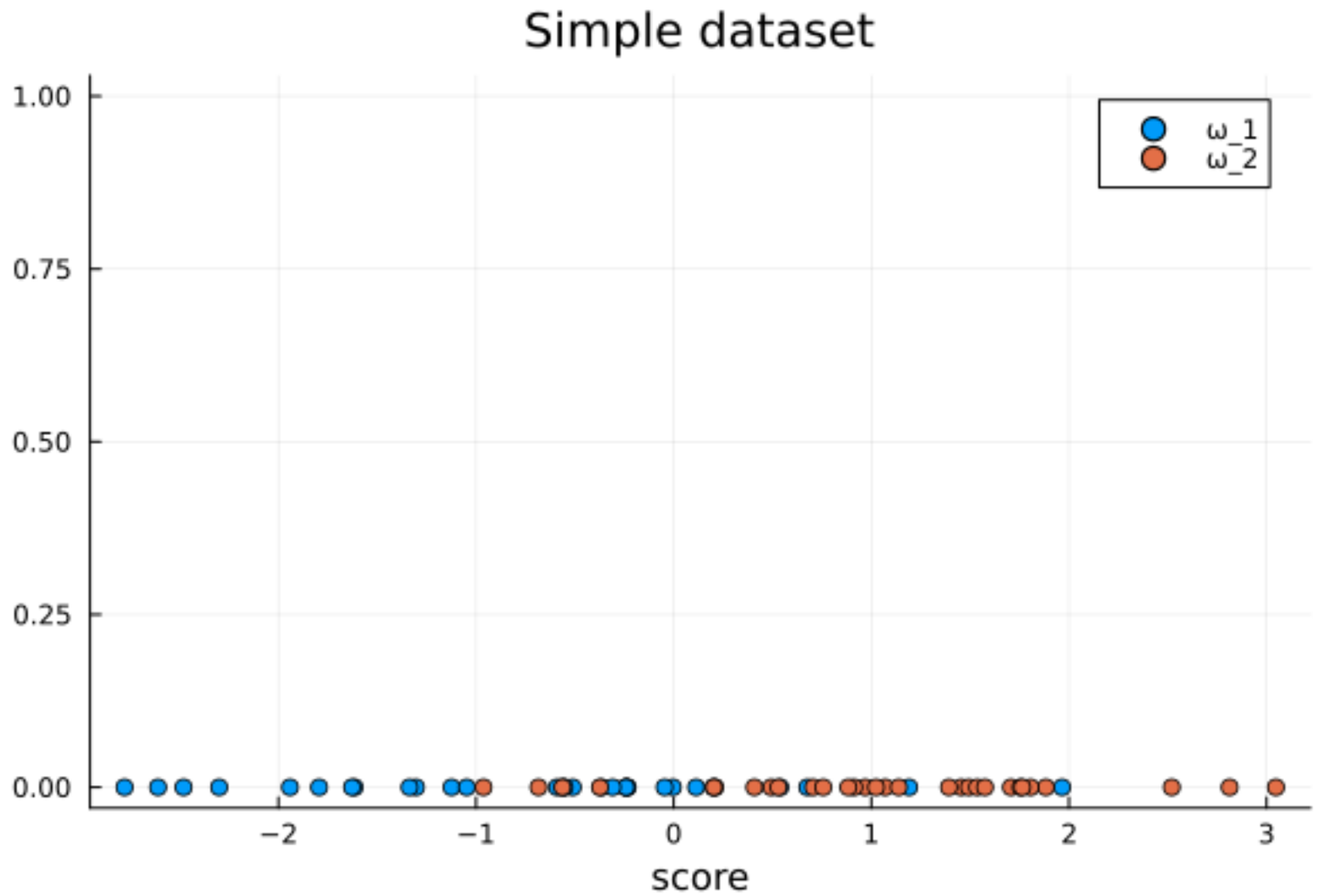
- But for some classifiers we only get a score (for each class):

$$s_k(\mathbf{x}),\ \ k = 1, .., K$$

- Can we convert

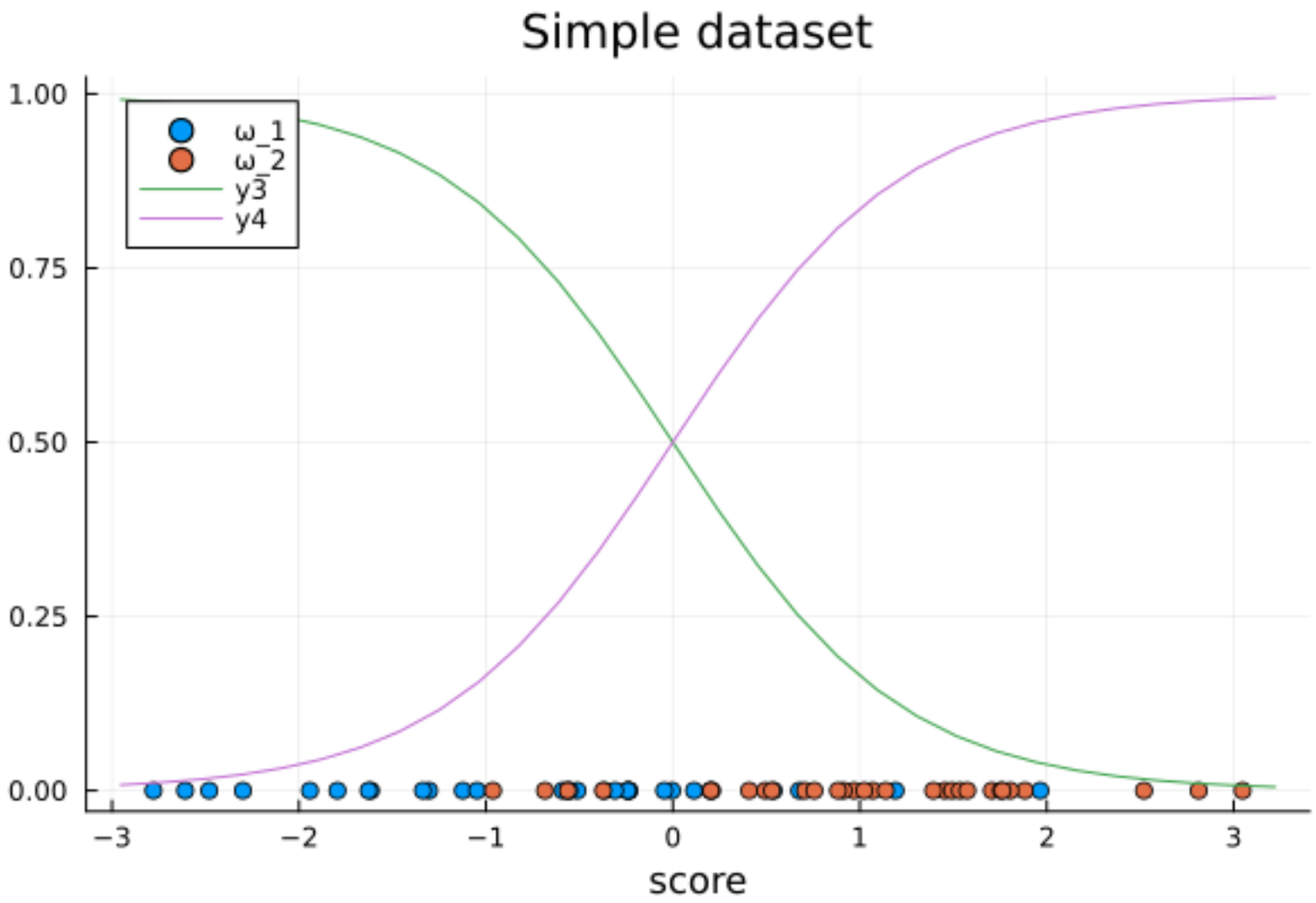
$$s_k(\mathbf{x}) \rightarrow \tilde{p}(y_k|\mathbf{x}),\ \ k = 1, .., K$$

‘Platt’ scaling



- Fit a logistic to the output score:
- $$s_k(\mathbf{x}) \rightarrow \tilde{p}(y_k|\mathbf{x}) = \frac{1}{1+\exp(-ws_k(\mathbf{x}))}$$

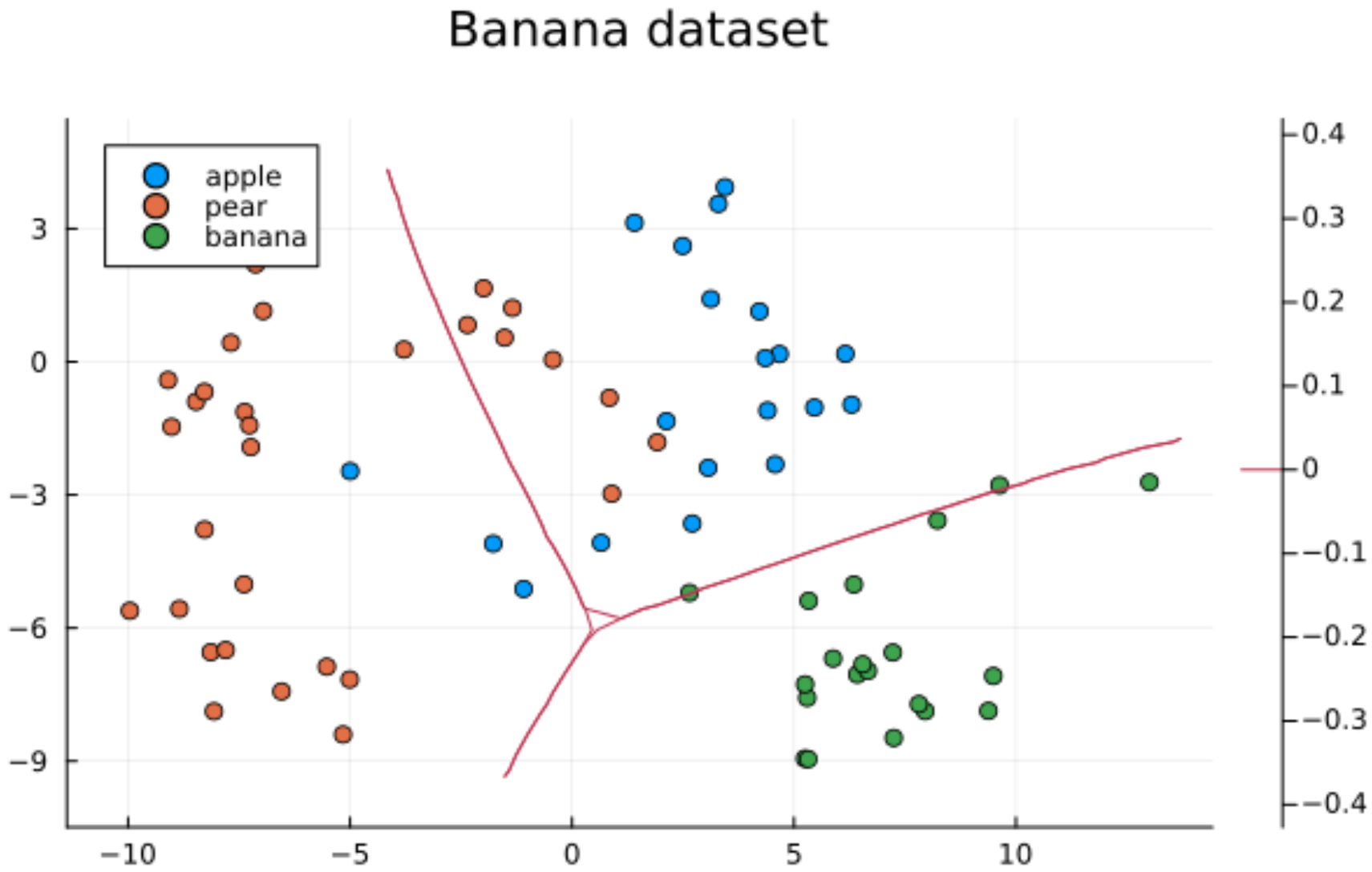
‘Platt’ scaling



- By fitting a logistic $s_k(\mathbf{x}) \rightarrow \tilde{p}(y_k|\mathbf{x}) = \dfrac{1}{1+\exp(-ws_k(\mathbf{x}))}$

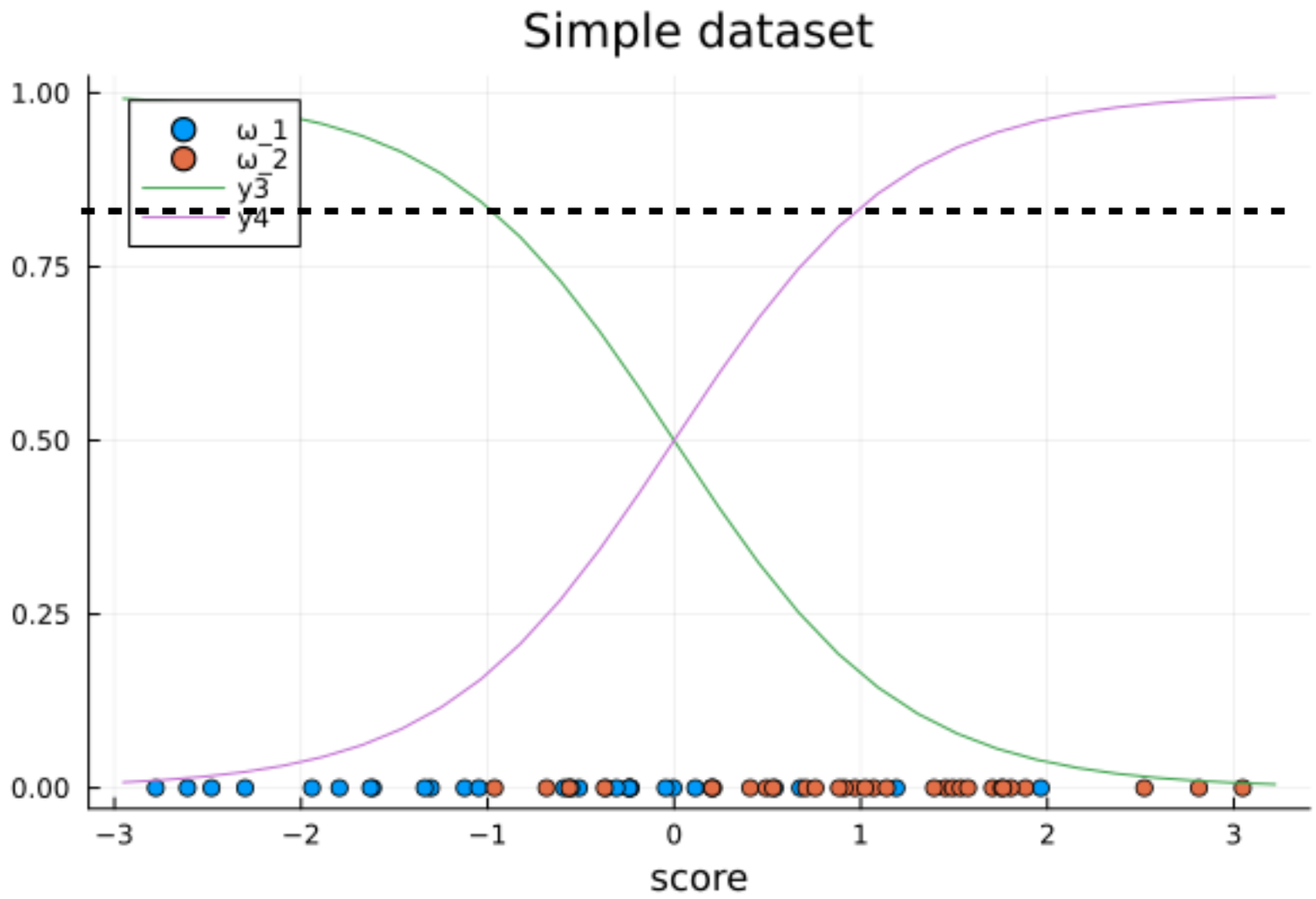
  without bias, the decision boundary does not change

Multi-class classification



- Use the estimated posteriors
- Assign to the highest estimated posterior

Classifier calibration



- Gives a **calibrated** classifier: you can trust the posterior probability:
  when $\tilde{p}(y_k|\mathbf{x}) = 0.8$ then in 80% object **x** came from y_k

Missing data

- Assume we have some medical records:

	age	height	weight
person A	34	1.80	70
person B	50	1.85	NaN
person C	71	1.60	65
person D	25	1.95	80

- What happened here?

Missing data

- Assume we have some medical records:

	age	height	weight
person A	34	1.80	70
person B	50	1.85	NaN
person C	71	1.60	65
person D	25	1.95	80

- Forgot to write down?

If independent of measured features:
MCAR: missing completely at random

If dependent on measured features:
MAR: missing at random

- Scales broke down due to weight?

MNAR: missing not at random

How to deal with missing values?

- Remove incomplete rows (data loss!)

- Impute (for MCAR)
  - Mean imputation (use known values)
  - Regression imputation (use other features)

- Use a model that can handle missing values (for MAR)
  - Decision trees, random forests
  - Probabilistic models (using the Expectation-Maximization procedure, see later)

Feature scaling

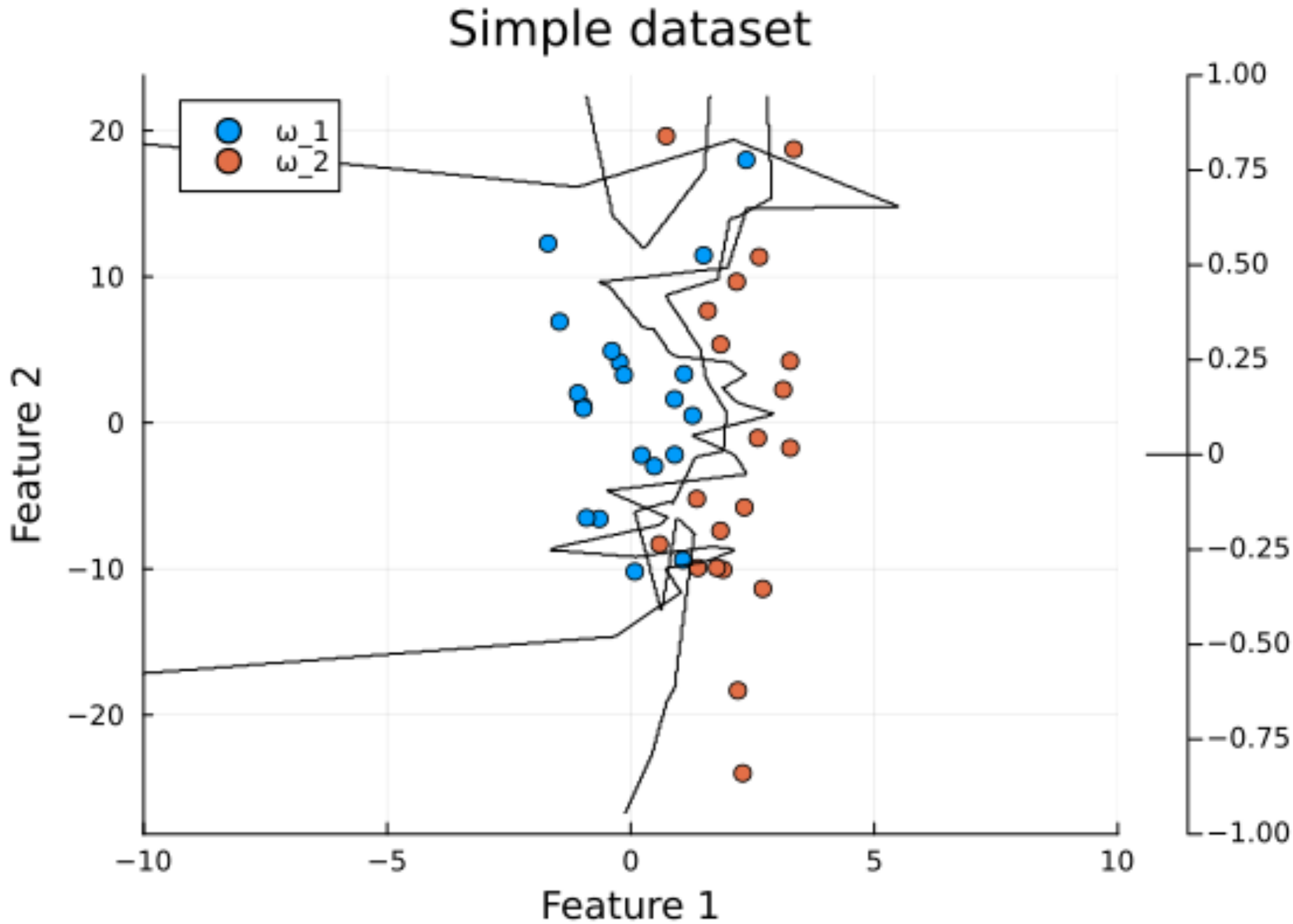
- Which classifiers will change (improve or deteriorate) when one or more features are rescaled?
  - Quadratic/Linear discriminant classifier
  - Nearest mean classifier
  - Parzen classifier
  - kNN classifier
  - Logistic classifier
  - Perceptron classifier
  - Neural networks
  - Decision trees
  - Support Vector Classifier
  - …

Feature scaling

- Which classifiers will change (improve or deteriorate) when one or more features are rescaled?
  - Quadratic/Linear discriminant classifier — No
  - Nearest mean classifier — Yes
  - Parzen classifier — Yes
  - kNN classifier — Yes
  - Logistic classifier — Yes and No
  - Perceptron classifier — Yes
  - Neural networks — No
  - Decision trees — No
  - Support Vector Classifier — Yes
  - …

Feature scaling

- 1-NN:



Sources of variation

- One last issue to consider:
  Assume you're training and testing a classifier on some dataset. You observe a very large spread in the test error, and you're not happy with it. How can you reduce the test error variance?

Sources of variation

- One last issue to consider:
  Assume you're training and testing a classifier on some dataset. You observe a very large spread in the test error, and you're not happy with it. How can you reduce the test error variance?
  - Get more (test-) data
  - Get a more simple model
  - Reduce the number of possible hyperparameters
  - Have a better optimizer
  - …

Conclusions

- Two-class classification problems are a very simpified Machine Learning problem
- In the real world, problems are way more complex
- Even for classification problems, need to take care of missing values, feature scaling, proper evaluation of the models, proper estimation of performance, …

- Next weeks: computational learning theory, regression, clustering, dimensionality reduction, fairness