Quantification Methods

Quantification: Predicting Class Frequencies via Supervised Learning

Alejandro Moreo, Fabrizio Sebastiani

ISTI-CNR, Pisa, Italy
{alejandro.moreo,fabrizio.sebastiani}@isti.cnr.it

ESSAI
Athens, Greece – July 22-26, 2024

Outline

Notation Recap

- $\mathcal{X} = \mathcal{R}^d$ input space

Notation Recap

- $\mathcal{X} = \mathcal{R}^d$ input space
- $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ label space; n the number of classes

Notation Recap

- $\mathcal{X} = \mathcal{R}^d$ input space
- $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ label space; n the number of classes
- $\mathbf{x} \in \mathcal{X}$ a data instance

Notation Recap

- $\mathcal{X} = \mathcal{R}^d$ input space
- $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ label space; n the number of classes
- $\mathbf{x} \in \mathcal{X}$ a data instance
- $L = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$ training data, sampled IID from distribution P
- $U = \{\mathbf{x}^{(i)}\}_{i=1}^{m'}$ test data, sampled IID from distribution Q

Notation Recap

- $\mathcal{X} = \mathcal{R}^d$ input space
- $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ label space; n the number of classes
- $\mathbf{x} \in \mathcal{X}$ a data instance
- $L = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$ training data, sampled IID from distribution P
- $U = \{\mathbf{x}^{(i)}\}_{i=1}^{m'}$ test data, sampled IID from distribution Q
- $p_\sigma(y)$ the true prevalence of y in sample σ (the "prior")
- $\hat{p}_\sigma^M(y)$ the prevalence of y in σ as estimated by method M

Notation Recap

- $\mathcal{X} = \mathcal{R}^d$ input space
- $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ label space; n the number of classes
- $\mathbf{x} \in \mathcal{X}$ a data instance
- $L = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$ training data, sampled IID from distribution P
- $U = \{\mathbf{x}^{(i)}\}_{i=1}^{m'}$ test data, sampled IID from distribution Q
- $p_\sigma(y)$ the true prevalence of y in sample σ (the "prior")
- $\hat{p}_\sigma^M(y)$ the prevalence of y in σ as estimated by method M
- $h : \mathcal{X} \to \mathcal{Y}$ a hard classifier
  - $h(\mathbf{x}) = \hat{y}$ a predicted label

Notation Recap

- $\mathcal{X} = \mathcal{R}^d$ input space
- $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ label space; n the number of classes
- $\mathbf{x} \in \mathcal{X}$ a data instance
- $L = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$ training data, sampled IID from distribution P
- $U = \{\mathbf{x}^{(i)}\}_{i=1}^{m'}$ test data, sampled IID from distribution Q
- $p_\sigma(y)$ the true prevalence of y in sample σ (the "prior")
- $\hat{p}_\sigma^M(y)$ the prevalence of y in σ as estimated by method M
- $h : \mathcal{X} \to \mathcal{Y}$ a hard classifier
  - $h(\mathbf{x}) = \hat{y}$ a predicted label
- $s : \mathcal{X} \to \Delta^{n-1}$ a soft classifier
  - $s_i(\mathbf{x}) = P(y_i|\mathbf{x})$ a posterior probability

Notation Recap

- $\mathcal{X} = \mathcal{R}^d$ input space
- $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ label space; n the number of classes
- $\mathbf{x} \in \mathcal{X}$ a data instance
- $L = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$ training data, sampled IID from distribution P
- $U = \{\mathbf{x}^{(i)}\}_{i=1}^{m'}$ test data, sampled IID from distribution Q
- $p_\sigma(y)$ the true prevalence of y in sample σ (the "prior")
- $\hat{p}_\sigma^M(y)$ the prevalence of y in σ as estimated by method M
- $h : \mathcal{X} \to \mathcal{Y}$ a hard classifier
  - $h(\mathbf{x}) = \hat{y}$ a predicted label
- $s : \mathcal{X} \to \Delta^{n-1}$ a soft classifier
  - $s_i(\mathbf{x}) = P(y_i|\mathbf{x})$ a posterior probability
- X, Y, $\hat{Y}$ random variables

Notation Recap (cont'd)

- In a binary setting, we might write $\mathcal{Y} = \{\ominus, \oplus\}$
- A binary classifier h is characterized by the contingency table:

		pred $\hat{y} = \ominus$	pred $\hat{y} = \oplus$
true	$y = \ominus$	TN	FP
	$y = \oplus$	FN	TP

- Useful values:

true positive rate

$$\text{tpr}_h = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

false positive rate

$$\text{fpr}_h = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

Classification VS Quantification

Classification

- Given a labeled training set, learn a classifier

$$h : \mathcal{X} \rightarrow \mathcal{Y}$$

- $\hat{y} = h(\mathbf{x})$, where $\mathbf{x} \in \mathcal{X}$ is a feature vector, and $\hat{y} \in \{y_1, \ldots y_n\}$ is a class label
- Error: false positives, false negatives

Quantification

- Given a labelled training set, learn a quantifier

$$q : \mathbb{N}^{\mathcal{X}} \rightarrow \Delta^{n-1}$$

- $\mathbf{p} = q(\sigma)$, with σ a sample of feature vectors, and p a vector of class prevalence values
- Error: underestimation, overestimation

Classification VS Quantification

Classification

- Given a labeled training set, learn a classifier

$$h : \mathcal{X} \to \mathcal{Y}$$

- $\hat{y} = h(\mathbf{x})$, where $\mathbf{x} \in \mathcal{X}$ is a feature vector, and $\hat{y} \in \{y_1, \ldots y_n\}$ is a class label
- Error: false positives, false negatives
- IID assumption

Quantification

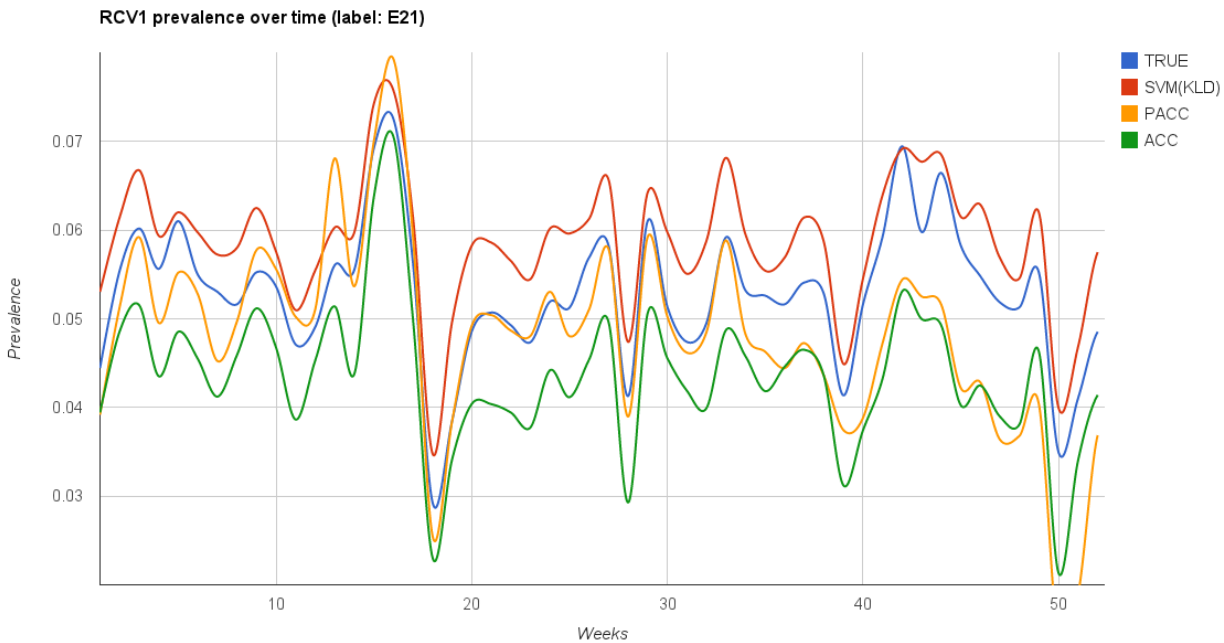
- Given a labelled training set, learn a quantifier

$$q : \mathbb{N}^{\mathcal{X}} \to \Delta^{n-1}$$

- $\mathbf{p} = q(\sigma)$, with σ a sample of feature vectors, and p a vector of class prevalence values
- Error: underestimation, overestimation
- Prior probability shift (PPS)

Prior probability shift (PPS)

- The need to perform quantification arises because of PPS.



- If we knew there is no shift, the problem would become trivial:
  - the training prevalence would already be a good estimator!

Prior probability shift (PPS)

- PPS is a special case of dataset shift in which

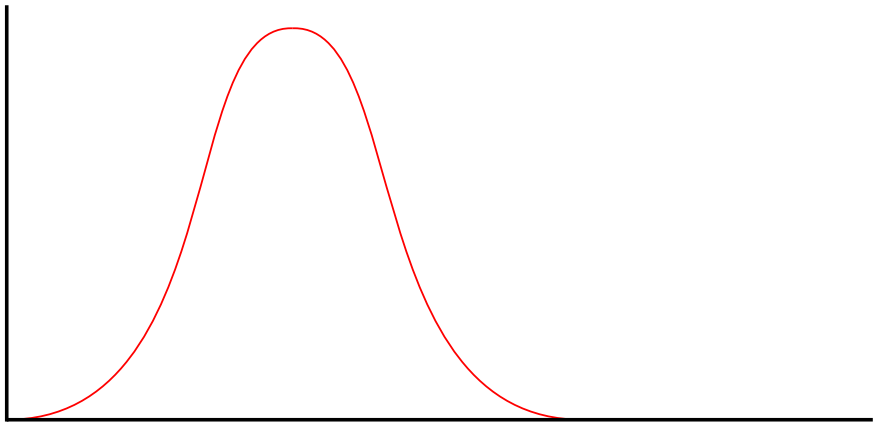
$$P(X, Y) \neq Q(X, Y)$$

- Factorization $P(X, Y) = P(X|Y)P(Y)$. PPS assumptions:

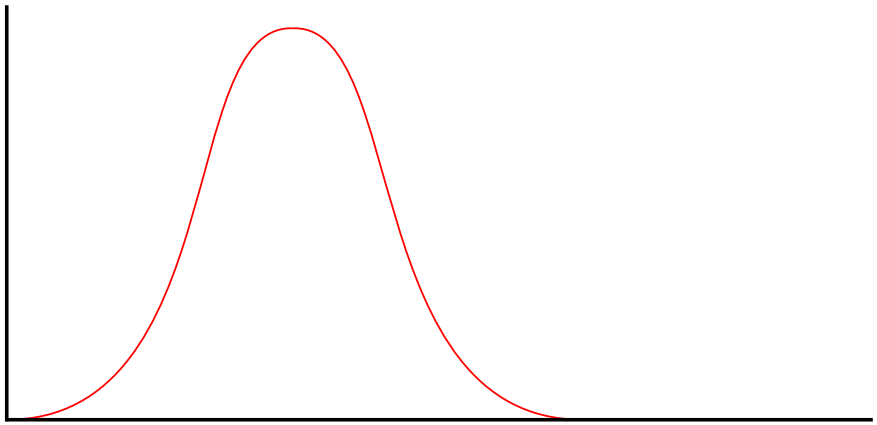
$$P(Y) \neq Q(Y)$$
$$P(X|Y) = Q(X|Y)$$

- Dataset shift may derive when
  - the environment is not stationary and the operating conditions are irreproducible at training time
  - in presence of sample selection bias, when the process of labelling training data introduces bias:
    - explicitly (e.g., by oversampling the minority class)
    - implicitly (e.g., if active learning is used)

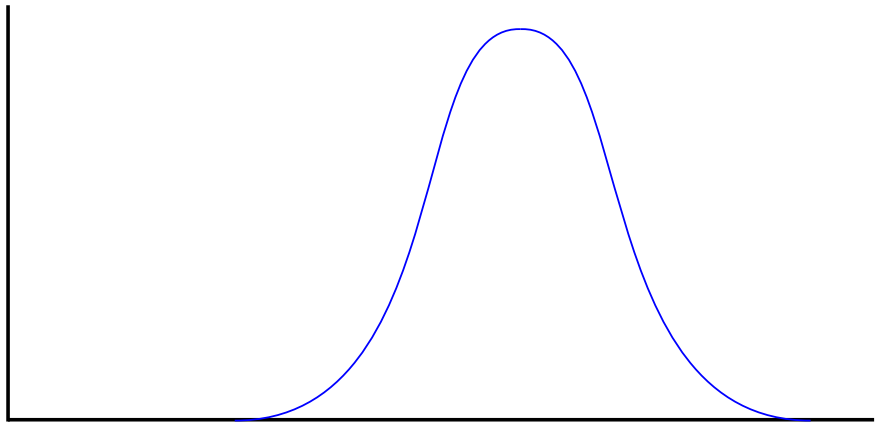
Prior Probability Shift (PPS)



Prior Probability Shift (PPS)

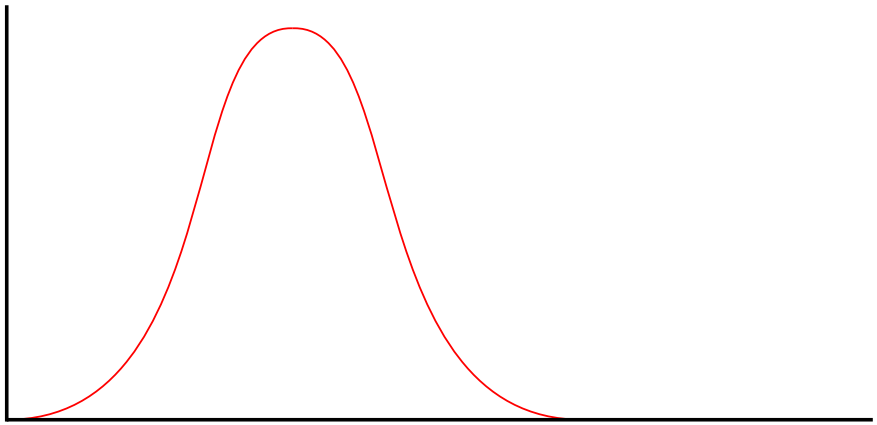


$P(X|Y = \ominus)$



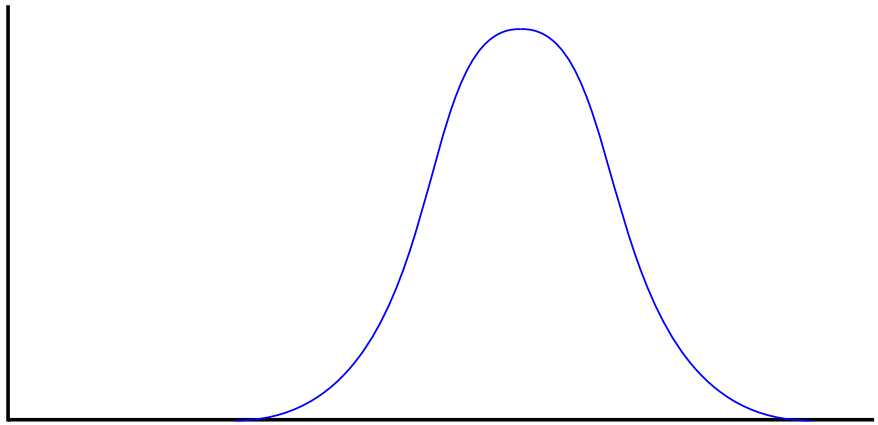
$P(X|Y = \oplus)$

Prior Probability Shift (PPS)



$P(X|Y = \ominus)$

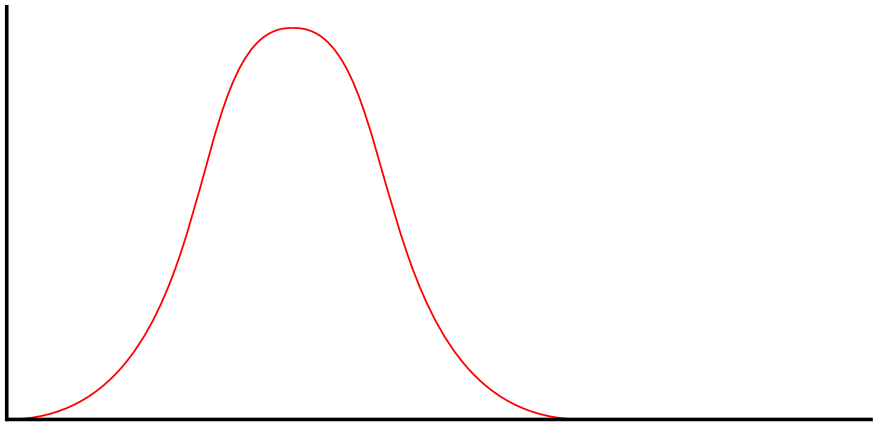
$Q(X|Y = \ominus)$



$P(X|Y = \oplus)$

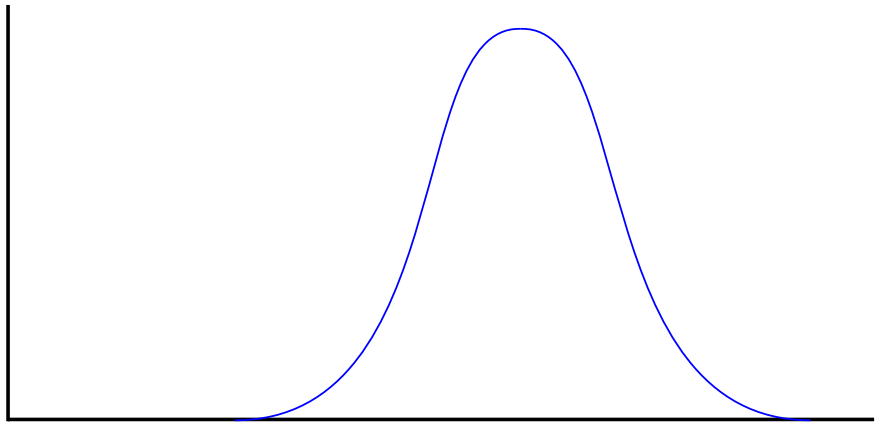
$Q(X|Y = \oplus)$

Prior Probability Shift (PPS)



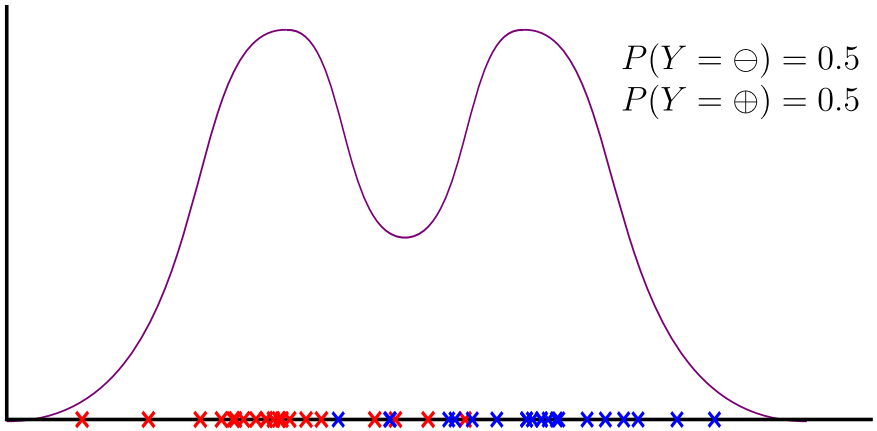
$P(X|Y = \ominus)$

$Q(X|Y = \ominus)$



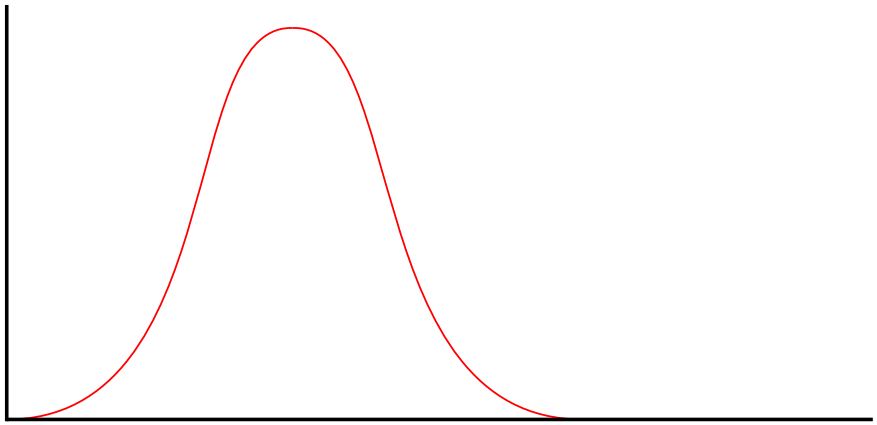
$P(X|Y = \oplus)$

$Q(X|Y = \oplus)$

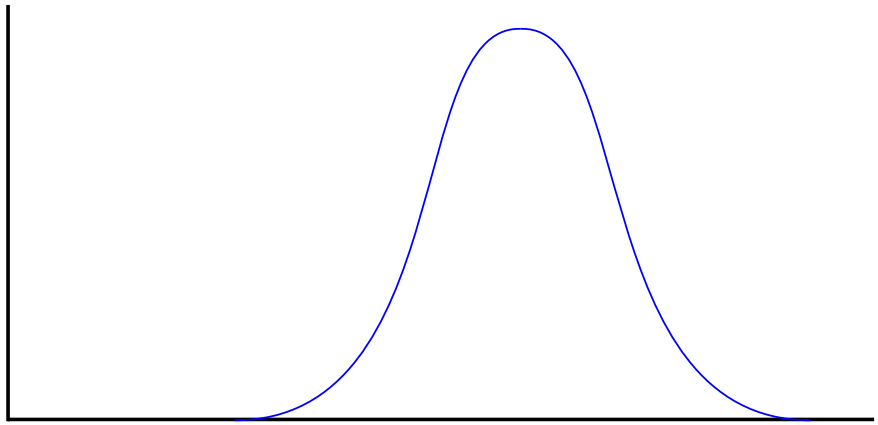


$P(X) = 0.5\ P(X|Y = \ominus) + 0.5\ P(X|Y = \oplus)$

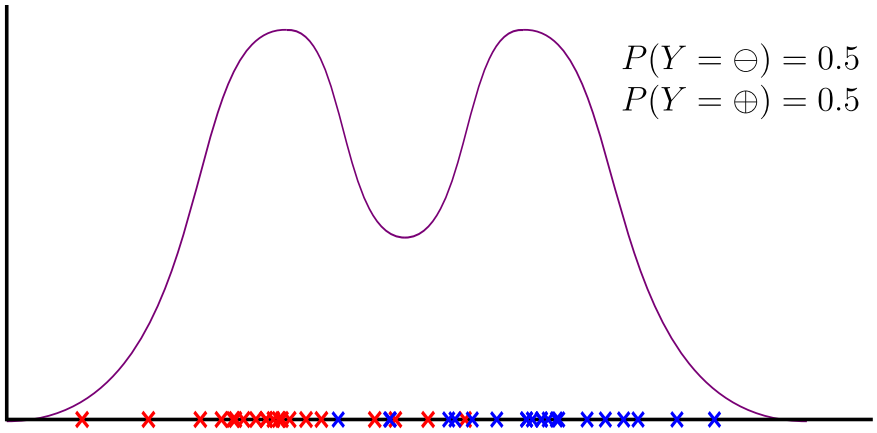
Prior Probability Shift (PPS)



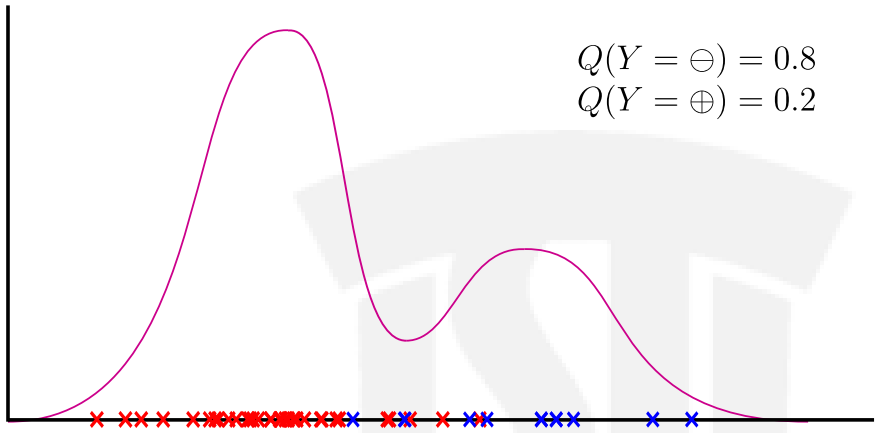
$P(X|Y = \ominus)$
$Q(X|Y = \ominus)$



$P(X|Y = \oplus)$
$Q(X|Y = \oplus)$



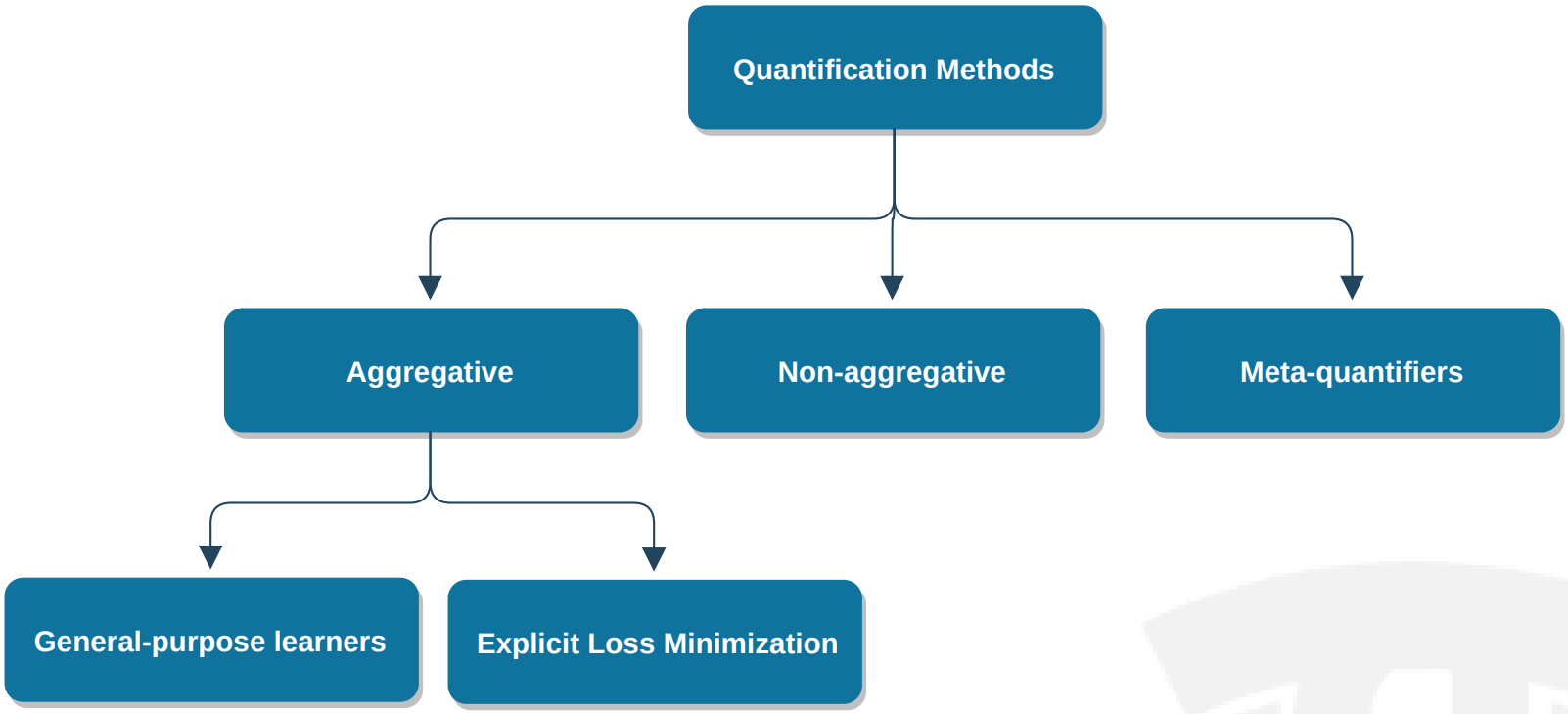
$P(X) = 0.5\ P(X|Y = \ominus) + 0.5\ P(X|Y = \oplus)$



$Q(X) = 0.8\ Q(X|Y = \ominus) + 0.2\ Q(X|Y = \oplus)$

Outline

An overview of the quantification methods



Aggregative Quantification

General-Purpose Learners

Classify & Count

Classify and Count (CC) consists of:

1. generating a classifier h from L
2. classifying the items in U
3. estimating $p_U(y_i)$ by counting the items predicted to be in y_i, i.e.,

$$\hat{p}_U^{\mathrm{CC}}(y_i) = \frac{|\{\mathbf{x} \in U : h(\mathbf{x}) = y_i\}|}{|U|}$$

Classify & Count

- But a good classifier is not necessarily a good quantifier:

	$y = \ominus$	$y = \oplus$
$\hat{y} = \ominus$	TN	FN
$\hat{y} = \oplus$	FP	TP

#ActualPositives=100 (16.7%)
#ActualNegatives=500 (83.3%)
#Instances=600

h_1	$y = \ominus$	$y = \oplus$
$\hat{y} = \ominus$	480	5
$\hat{y} = \oplus$	20	95

#Errors=25, Accuracy=96%
#PredictedPositives=115 (19.1%)
#ActualPositives=100 (16.7%)

h_2	$y = \ominus$	$y = \oplus$
$\hat{y} = \ominus$	470	30
$\hat{y} = \oplus$	30	70

#Errors=60, Accuracy=90%
#PredictedPositives=100 (16.7%)
#ActualPositives=100 (16.7%)

- Which classifier would you prefer?

Classify & Count

- But a good classifier is not necessarily a good quantifier:

	$y = \ominus$	$y = \oplus$
$\hat{y} = \ominus$	TN	FN
$\hat{y} = \oplus$	FP	TP

#ActualPositives=100 (16.7%)
#ActualNegatives=500 (83.3%)
#Instances=600

h_1	$y = \ominus$	$y = \oplus$
$\hat{y} = \ominus$	480	5
$\hat{y} = \oplus$	20	95

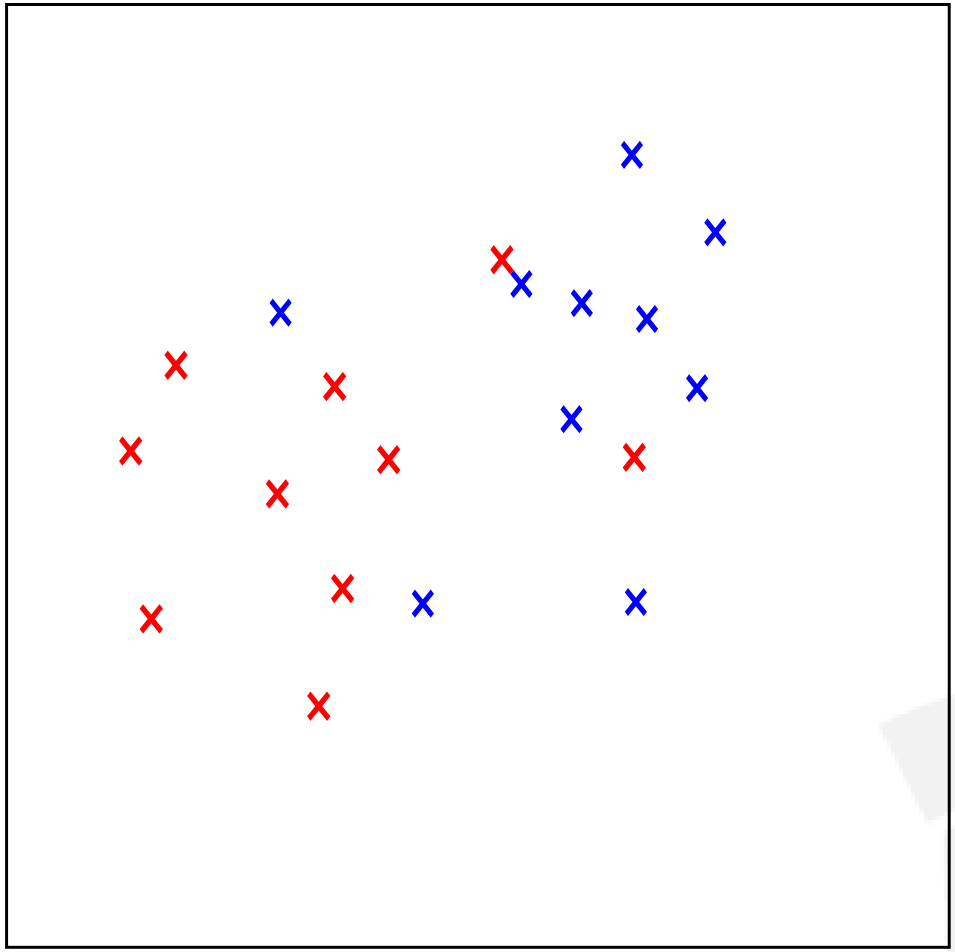
#Errors=25, Accuracy=96%
#PredictedPositives=115 (19.1%)
#ActualPositives=100 (16.7%)

h_2	$y = \ominus$	$y = \oplus$
$\hat{y} = \ominus$	470	30
$\hat{y} = \oplus$	30	70

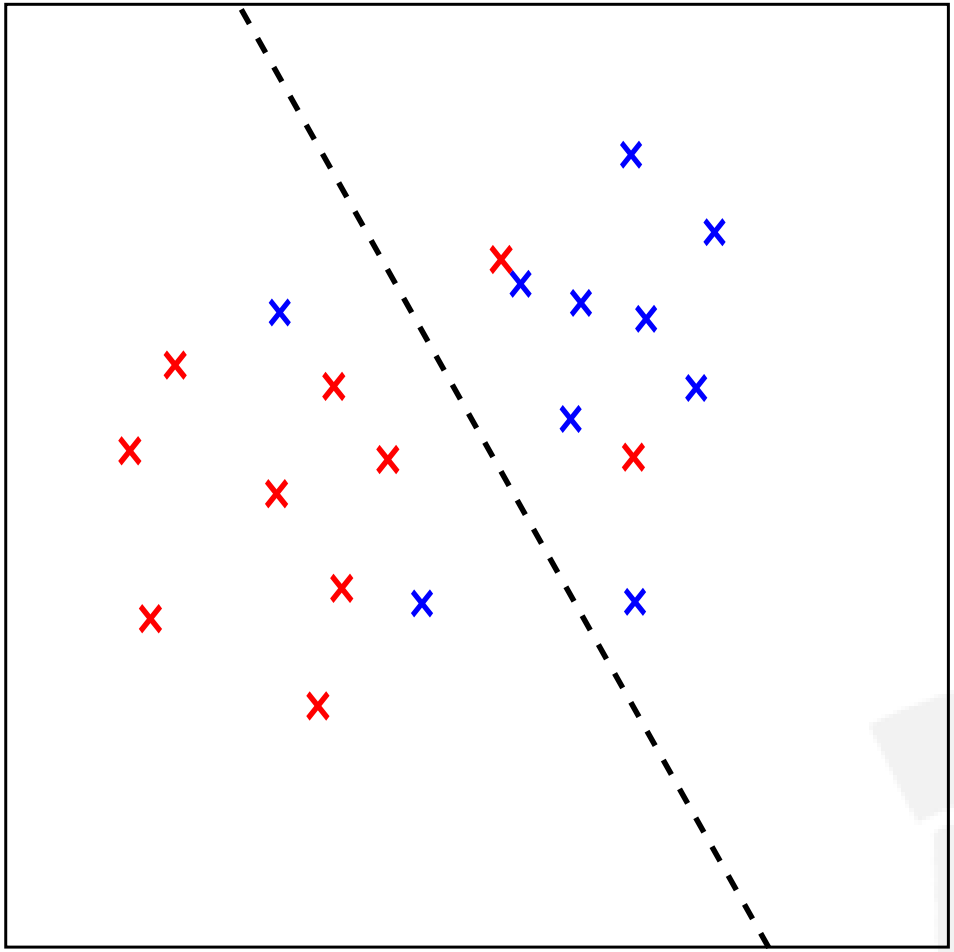
#Errors=60, Accuracy=90%
#PredictedPositives=100 (16.7%)
#ActualPositives=100 (16.7%)

- Paradoxically, for quantification purposes we should prefer h_2 to h_1
- Problem:
  - classifiers are tuned to minimize $(\mathrm{FP} + \mathrm{FN})$ (or a proxy of it)
  - quantifiers should minimize $|\,\mathrm{FP} - \mathrm{FN}\,|$

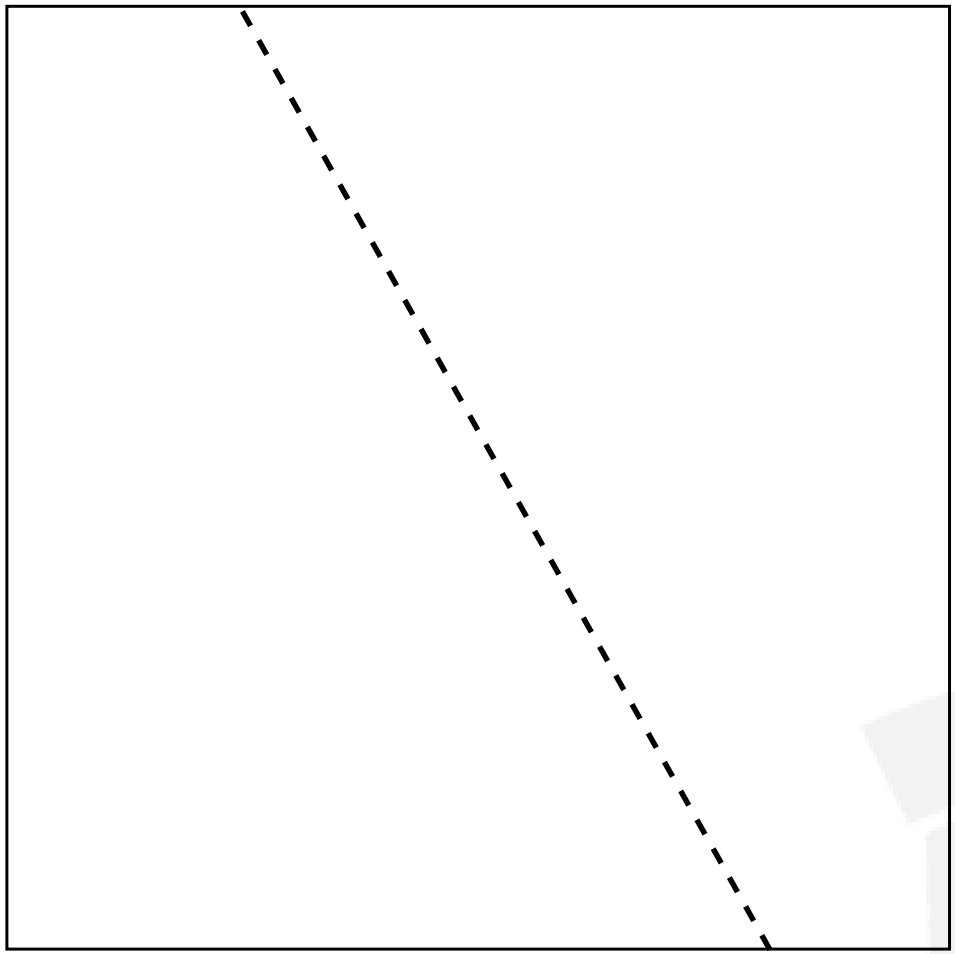
CC against PPS



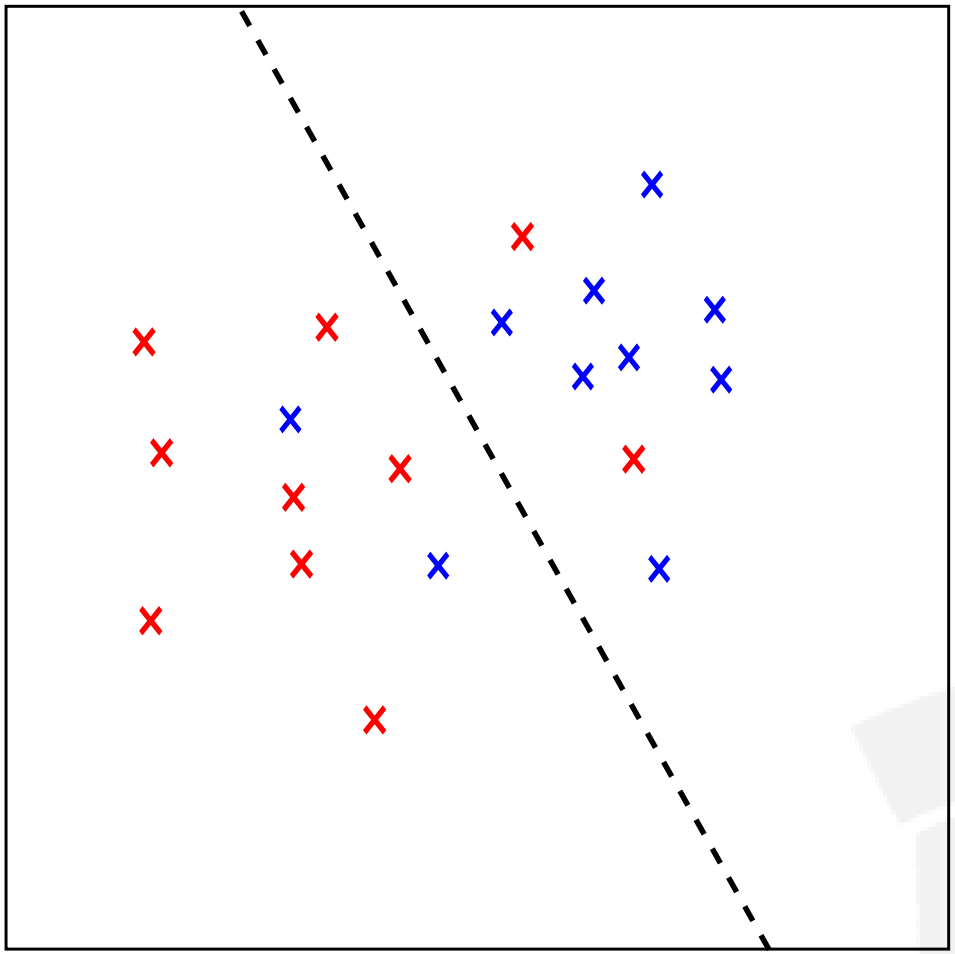
CC against PPS



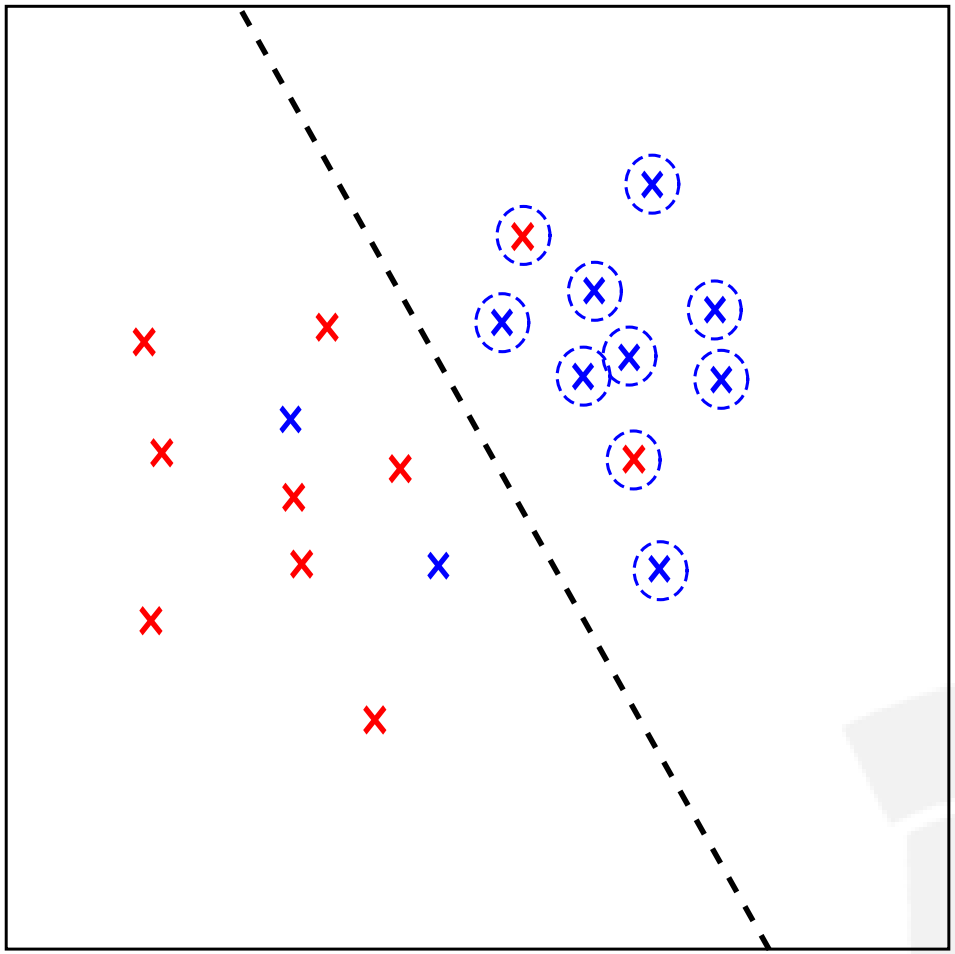
CC against PPS



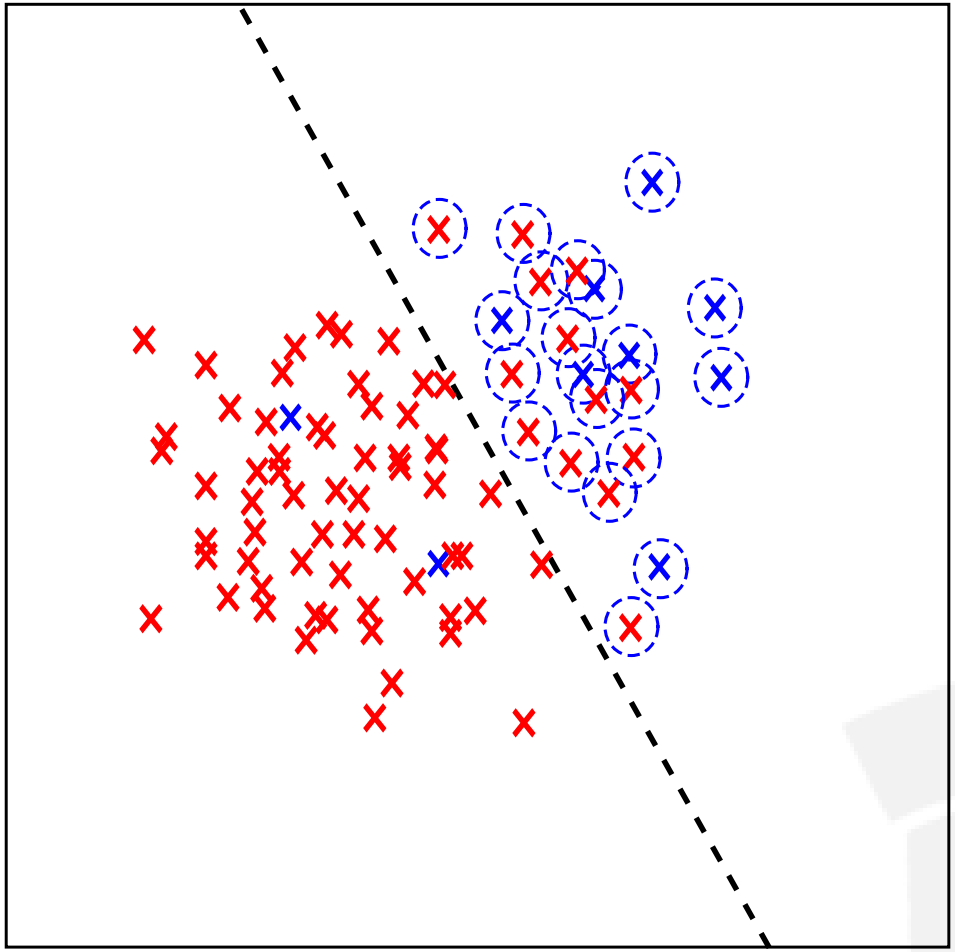
CC against PPS



CC against PPS



CC against PPS



ACC

- Adjusted Classify and Count (ACC, a.k.a. the Confusion Matrix Method) is based on the law of total probability:

$$Q(\hat{Y} = y_i) = \sum_{y_j \in \mathcal{Y}} Q(\hat{Y} = y_i | Y = y_j) \cdot Q(Y = y_j)$$

G. Forman. Counting positives accurately despite inaccurate classification. ECML 2005.

ACC

- Adjusted Classify and Count (ACC, a.k.a. the Confusion Matrix Method) is based on the law of total probability:

$$Q(\hat{Y} = y_i) = \sum_{y_j \in \mathcal{Y}} Q(\hat{Y} = y_i | Y = y_j) \cdot Q(Y = y_j)$$

- The $Q(\hat{Y} = y_i)$'s are observed; indeed, this is $\hat{p}^{\mathrm{CC}}(y_i)$.

G. Forman. Counting positives accurately despite inaccurate classification. ECML 2005.

ACC

- Adjusted Classify and Count (ACC, a.k.a. the Confusion Matrix Method) is based on the law of total probability:

$$Q(\hat{Y} = y_i) = \sum_{y_j \in \mathcal{Y}} Q(\hat{Y} = y_i | Y = y_j) \cdot Q(Y = y_j)$$

- The $Q(\hat{Y} = y_i)$'s are observed; indeed, this is $\hat{p}^{\text{CC}}(y_i)$.
- The $Q(\hat{Y} = y_i | Y = y_j)$'s represent the "class-conditional bias", that are unknown but that can be estimated on L via k-fold cross-validation.

G. Forman. Counting positives accurately despite inaccurate classification. ECML 2005.

ACC

- Adjusted Classify and Count (ACC, a.k.a. the Confusion Matrix Method) is based on the law of total probability:

$$Q(\hat{Y} = y_i) = \sum_{y_j \in \mathcal{Y}} Q(\hat{Y} = y_i | Y = y_j) \cdot Q(Y = y_j)$$

- The $Q(\hat{Y} = y_i)$'s are observed; indeed, this is $\hat{p}^{\mathrm{CC}}(y_i)$.
- The $Q(\hat{Y} = y_i | Y = y_j)$'s represent the "class-conditional bias", that are unknown but that can be estimated on L via k-fold cross-validation.
- The $Q(Y = y_j)$ are the true priors, that we want to estimate.

G. Forman. Counting positives accurately despite inaccurate classification. ECML 2005.

ACC

- Adjusted Classify and Count (ACC, a.k.a. the Confusion Matrix Method) is based on the law of total probability:

$$Q(\hat{Y} = y_i) = \sum_{y_j \in \mathcal{Y}} Q(\hat{Y} = y_i | Y = y_j) \cdot Q(Y = y_j)$$

- The $Q(\hat{Y} = y_i)$'s are observed; indeed, this is $\hat{p}^{\mathrm{CC}}(y_i)$.
- The $Q(\hat{Y} = y_i | Y = y_j)$'s represent the "class-conditional bias", that are unknown but that can be estimated on L via k-fold cross-validation.
- The $Q(Y = y_j)$ are the true priors, that we want to estimate.
- We have a system of n linear equations ($n = |\mathcal{Y}|$) with n unknowns!

G. Forman. Counting positives accurately despite inaccurate classification. ECML 2005.

ACC

- But why can we use the "class-conditional bias" of the training set as an estimate of the one in the test set? That is:

$$Q(\hat{Y} = y_i) = \sum_{y_j \in \mathcal{Y}} Q(\hat{Y} = y_i | Y = y_j) \cdot Q(Y = y_j)$$
$$= \sum_{y_j \in \mathcal{Y}} P(\hat{Y} = y_i | Y = y_j) \cdot Q(Y = y_j)$$

Lipton, Wang, Smola, Detecting and correcting for label shift with black box predictors, ICML 2018.

ACC

- But why can we use the "class-conditional bias" of the training set as an estimate of the one in the test set? That is:

$$Q(\hat{Y} = y_i) = \sum_{y_j \in \mathcal{Y}} Q(\hat{Y} = y_i | Y = y_j) \cdot Q(Y = y_j)$$
$$= \sum_{y_j \in \mathcal{Y}} P(\hat{Y} = y_i | Y = y_j) \cdot Q(Y = y_j)$$

Lipton, Wang, Smola, Detecting and correcting for label shift with black box predictors, ICML 2018.

ACC

- But why can we use the "class-conditional bias" of the training set as an estimate of the one in the test set? That is:

$$Q(\hat{Y} = y_i) = \sum_{y_j \in \mathcal{Y}} Q(\hat{Y} = y_i | Y = y_j) \cdot Q(Y = y_j)$$
$$= \sum_{y_j \in \mathcal{Y}} P(\hat{Y} = y_i | Y = y_j) \cdot Q(Y = y_j)$$

- This is a direct consequence of the PPS assumptions:

$$[P(X|Y) = Q(X|Y)] \rightarrow [P(Z|Y) = Q(Z|Y)]$$

Lipton, Wang, Smola, Detecting and correcting for label shift with black box predictors, ICML 2018.

ACC

- But why can we use the "class-conditional bias" of the training set as an estimate of the one in the test set? That is:

$$Q(\hat{Y} = y_i) = \sum_{y_j \in \mathcal{Y}} Q(\hat{Y} = y_i | Y = y_j) \cdot Q(Y = y_j)$$
$$= \sum_{y_j \in \mathcal{Y}} P(\hat{Y} = y_i | Y = y_j) \cdot Q(Y = y_j)$$

- This is a direct consequence of the PPS assumptions:

$$[P(X|Y) = Q(X|Y)] \rightarrow [P(Z|Y) = Q(Z|Y)]$$

- ... with $Z = f(X)$ a measurable mapping. In particular, we take $f = h$, our classifier, as the mapping $\hat{Y} = h(X)$, so it holds that $P(\hat{Y}|Y) = Q(\hat{Y}|Y)$

Lipton, Wang, Smola, Detecting and correcting for label shift with black box predictors, ICML 2018.

ACC: binary

- In the binary case, estimating prevalence of a sample σ comes down to

$$Q(\hat{Y} = \oplus) = Q(\hat{Y} = \oplus | Y = \oplus) \cdot Q(\oplus) + Q(\hat{Y} = \oplus | Y = \ominus) \cdot Q(\ominus)$$

ACC: binary

- In the binary case, estimating prevalence of a sample σ comes down to

$$Q(\hat{Y} = \oplus) = Q(\hat{Y} = \oplus | Y = \oplus) \cdot Q(\oplus) + Q(\hat{Y} = \oplus | Y = \ominus) \cdot Q(\ominus)$$

- This can be rewritten as:

$$\begin{aligned} \hat{p}_U^{\mathsf{CC}}(\oplus) &= \mathrm{tpr}_h \cdot Q(\oplus) + \mathrm{fpr}_h \cdot Q(\ominus) \\ &= \mathrm{tpr}_h \cdot Q(\oplus) + \mathrm{fpr}_h \cdot (1 - Q(\oplus)) \end{aligned}$$

ACC: binary

- In the binary case, estimating prevalence of a sample σ comes down to

$$Q(\hat{Y} = \oplus) = Q(\hat{Y} = \oplus | Y = \oplus) \cdot Q(\oplus) + Q(\hat{Y} = \oplus | Y = \ominus) \cdot Q(\ominus)$$

- This can be rewritten as:

$$\begin{aligned} \hat{p}_U^{\mathrm{CC}}(\oplus) &= \mathrm{tpr}_h \cdot Q(\oplus) + \mathrm{fpr}_h \cdot Q(\ominus) \\ &= \mathrm{tpr}_h \cdot Q(\oplus) + \mathrm{fpr}_h \cdot (1 - Q(\oplus)) \end{aligned}$$

- Where tpr_h and fpr_h are the true positive rate and the false positive rate:

$$Q(\oplus) = \frac{\hat{p}_U^{\mathrm{CC}}(\oplus) - \mathrm{fpr}_h}{\mathrm{tpr}_h - \mathrm{fpr}_h}$$

ACC: binary

- In the binary case, estimating prevalence of a sample σ comes down to

$$Q(\hat{Y} = \oplus) = Q(\hat{Y} = \oplus | Y = \oplus) \cdot Q(\oplus) + Q(\hat{Y} = \oplus | Y = \ominus) \cdot Q(\ominus)$$

- This can be rewritten as:

$$\begin{aligned} \hat{p}_U^{\mathrm{CC}}(\oplus) &= \mathrm{tpr}_h \cdot Q(\oplus) + \mathrm{fpr}_h \cdot Q(\ominus) \\ &= \mathrm{tpr}_h \cdot Q(\oplus) + \mathrm{fpr}_h \cdot (1 - Q(\oplus)) \end{aligned}$$

- Where tpr_h and fpr_h are the true positive rate and the false positive rate:

$$Q(\oplus) = \frac{\hat{p}_U^{\mathrm{CC}}(\oplus) - \mathrm{fpr}_h}{\mathrm{tpr}_h - \mathrm{fpr}_h}$$

- The ACC is obtained by replacing the true tpr_h and fpr_h with estimates obtained in training (PPS assumption):

$$\hat{p}_U^{\mathrm{ACC}}(\oplus) = \frac{\hat{p}_U^{\mathrm{CC}}(\oplus) - \hat{\mathrm{fpr}}_h}{\hat{\mathrm{tpr}}_h - \hat{\mathrm{fpr}}_h}$$

Forman's variants of ACC

- In binary cases, the denominator of ACC...

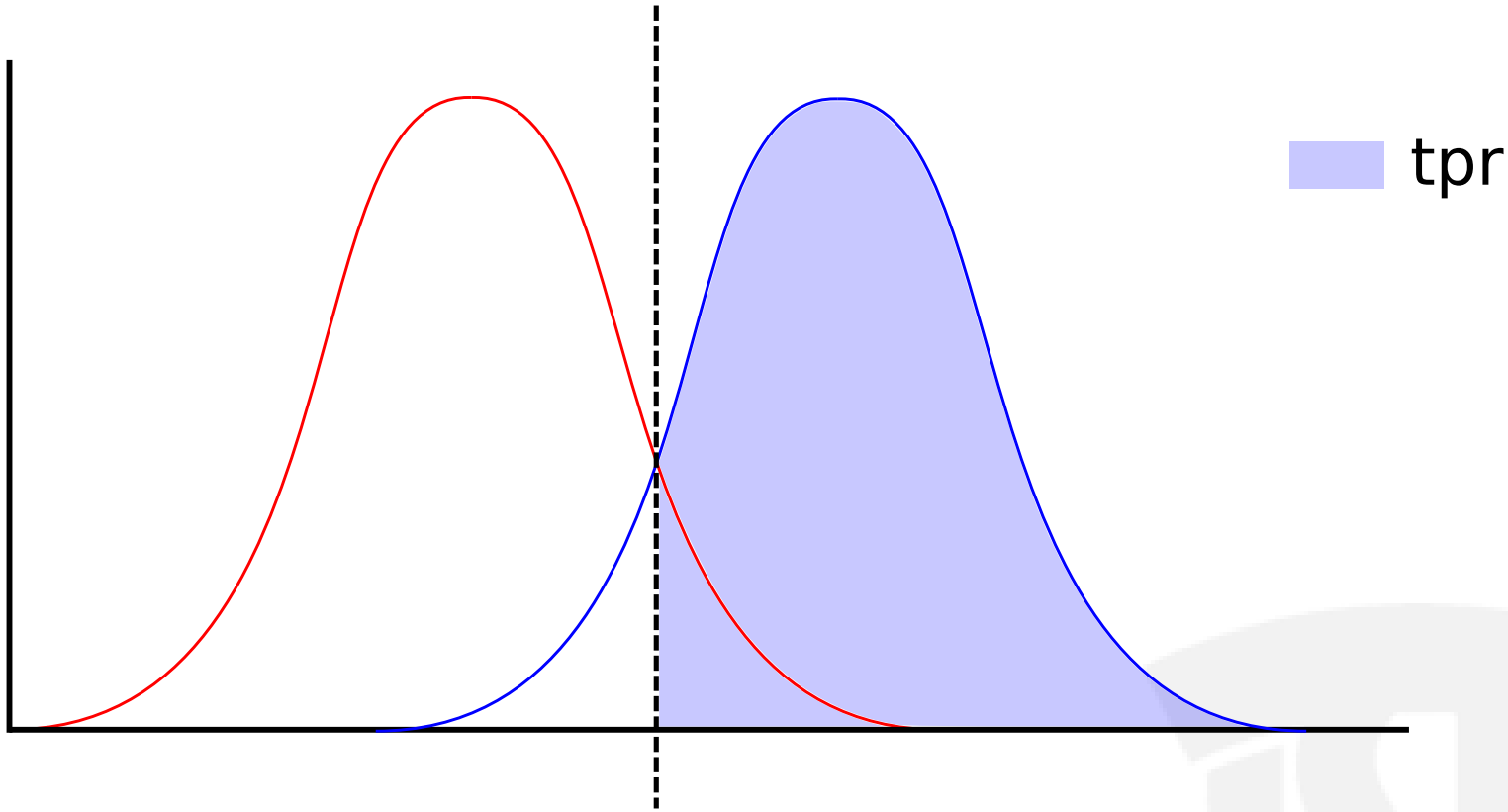
$$\hat{p}_\sigma^{\mathrm{ACC}}(\oplus) = \frac{\hat{p}_\sigma^{\mathrm{CC}}(\oplus) - \hat{\mathrm{fpr}}_h}{\hat{\mathrm{tpr}}_h - \hat{\mathrm{fpr}}_h}$$

- ... can become unstable when $\hat{\mathrm{tpr}}_h \approx \hat{\mathrm{fpr}}_h$.
- Forman proposes different heuristics for deciding the classification threshold, trying to fulfill the following conditions:
  - T50: $\hat{\mathrm{tpr}}_h \approx 0.5$
  - X: $\hat{\mathrm{tpr}}_h \approx (1 - \hat{\mathrm{fpr}}_h)$
  - MAX: maximize $(\hat{\mathrm{tpr}}_h - \hat{\mathrm{fpr}}_h)$
  - Median Sweep: compute ACC for all thresholds, then report the median
  - Median Sweep 2: compute ACC for all thresholds for which $(\hat{\mathrm{tpr}}_h - \hat{\mathrm{fpr}}_h) > \frac{1}{4}$, then report the median

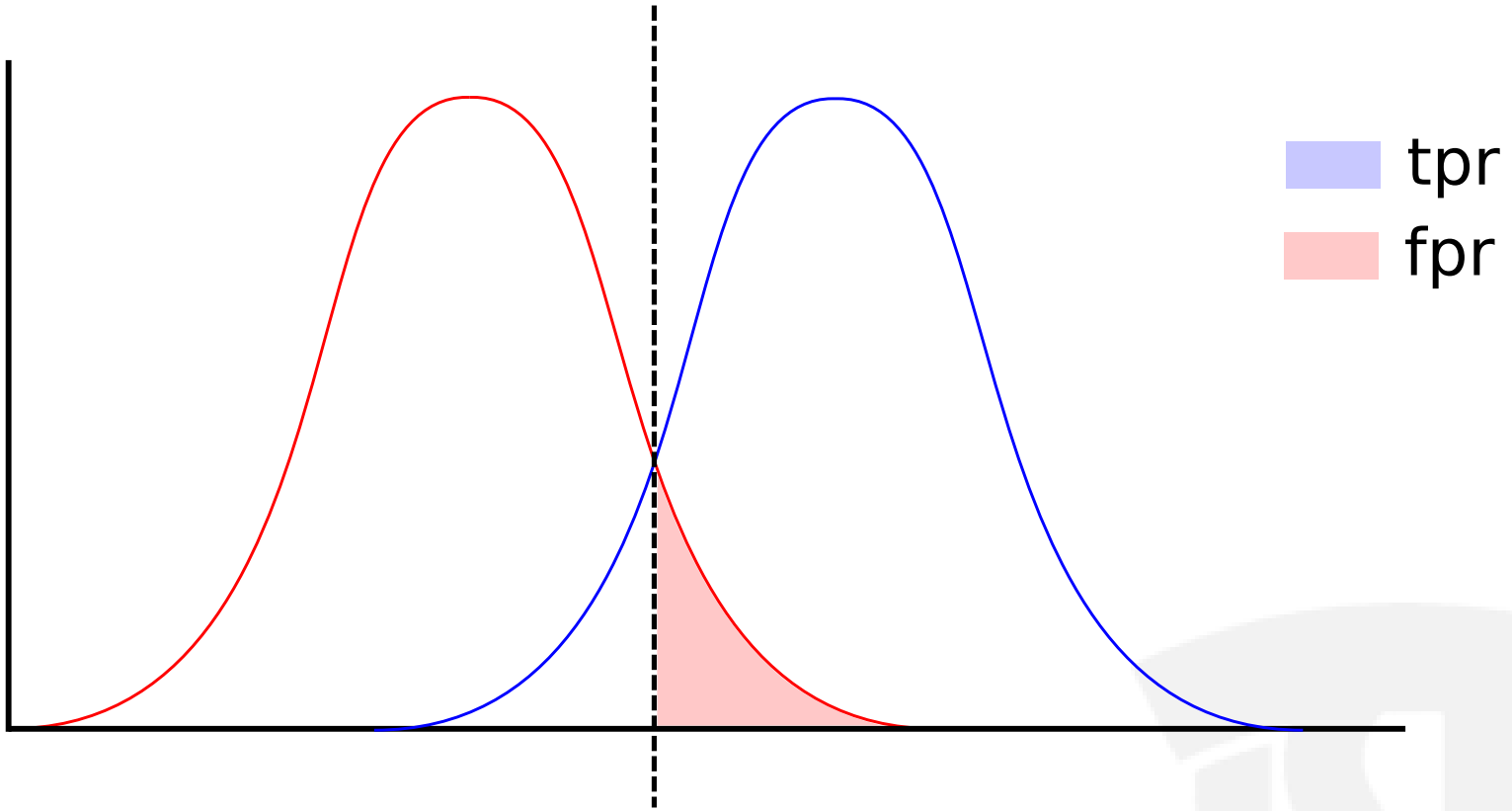
Forman, G., Quantifying trends accurately despite classifier error and class imbalance. KDD 2006.

Forman, G., Quantifying counts and costs via classification. Data Mining and Knowledge Discovery, 2008.

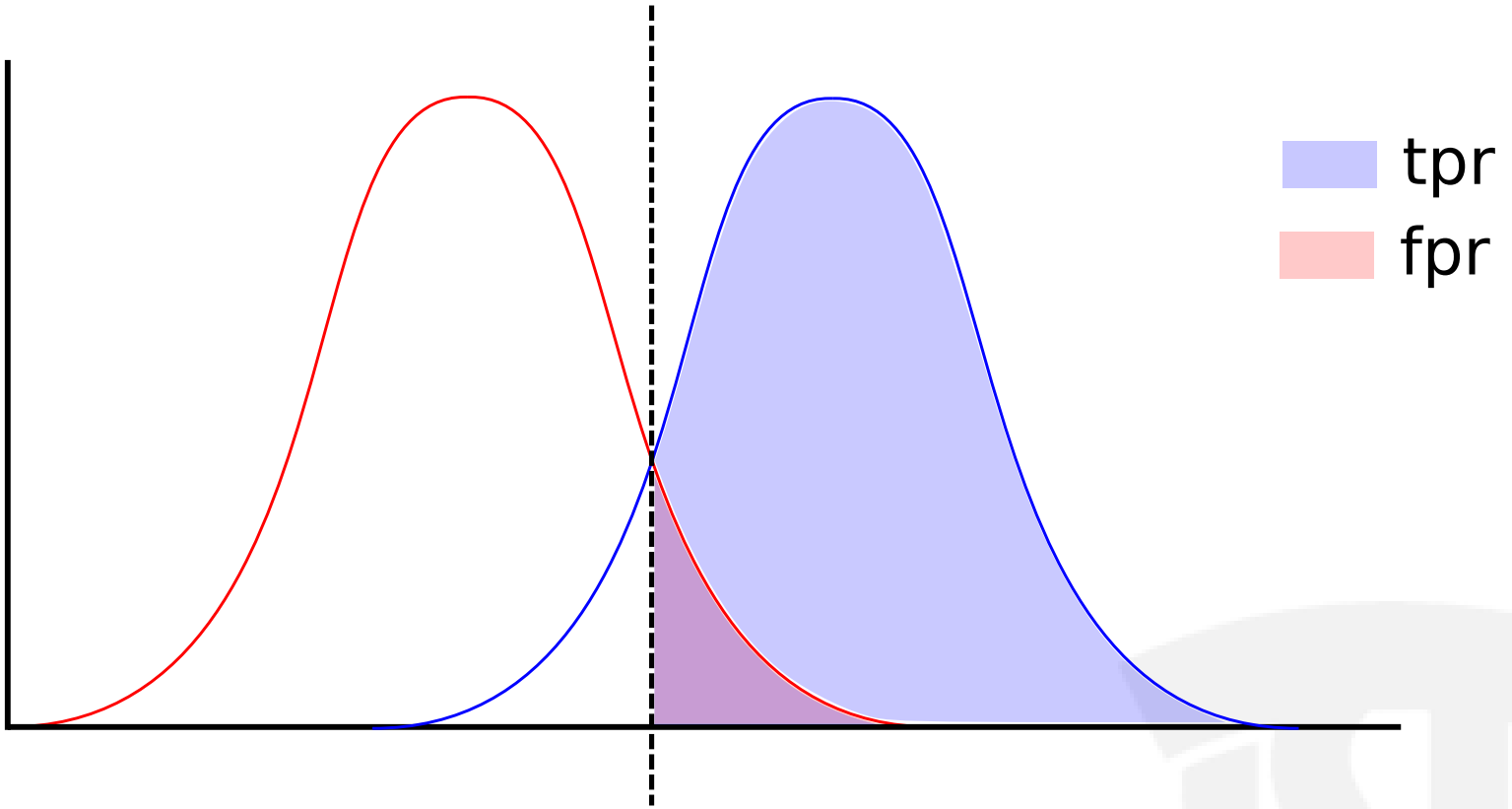
Forman's variants of ACC



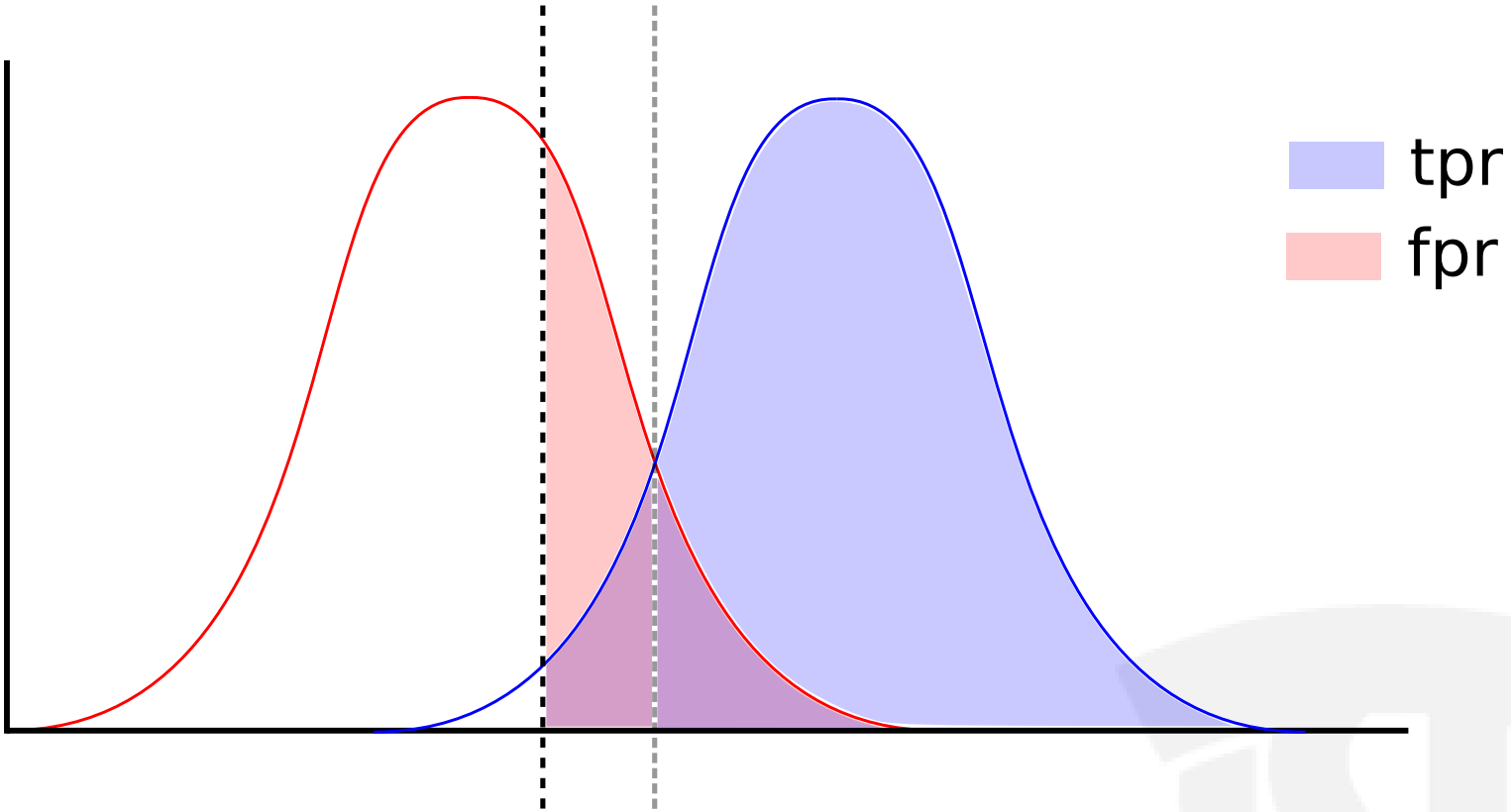
Forman's variants of ACC



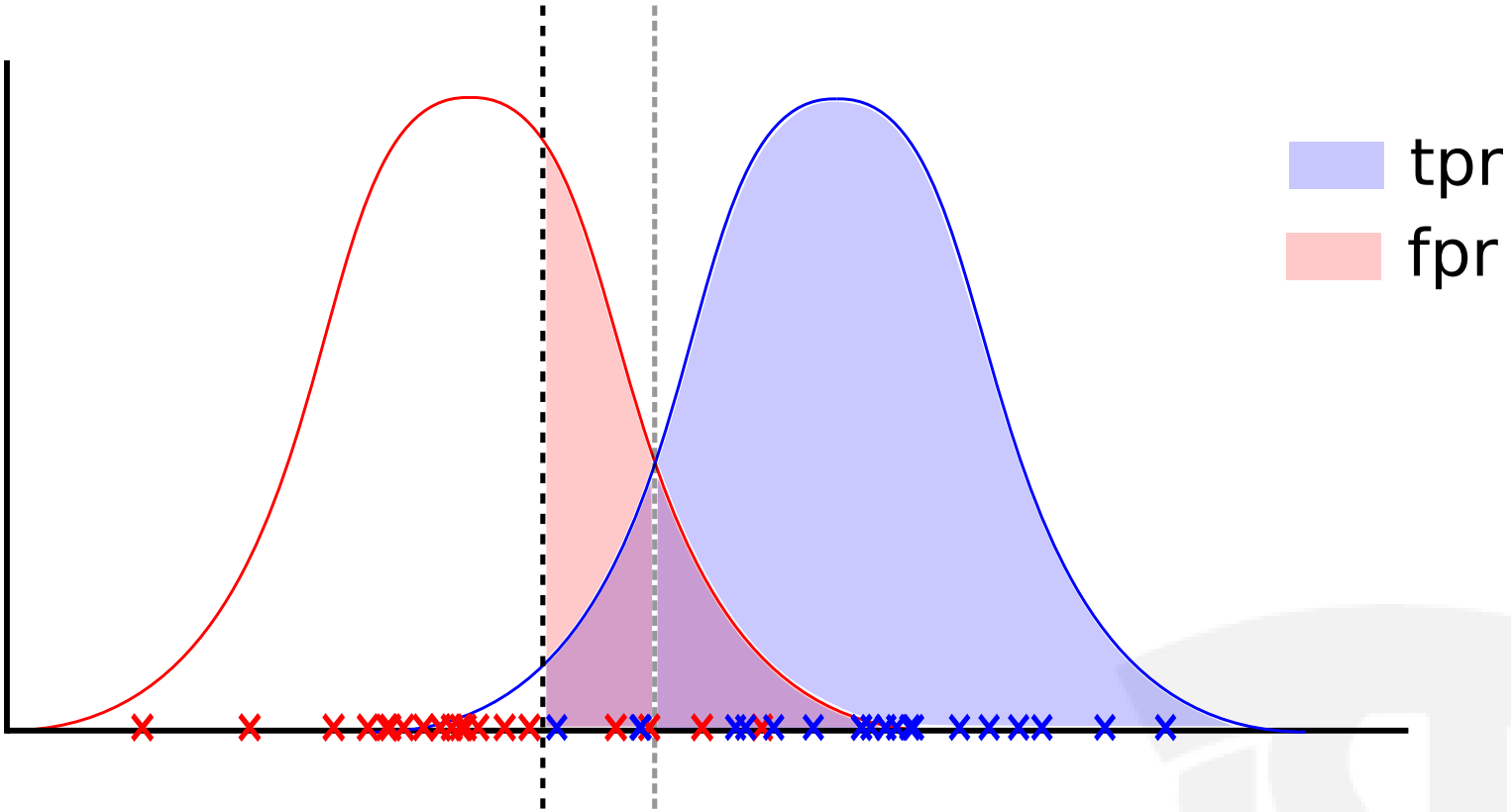
Forman's variants of ACC



Forman's variants of ACC



Forman's variants of ACC



ACC: Multiclass

- The system of linear equations can be written in matrix form:

$$\hat{\mathbf{p}}_U^{\mathrm{CC}} = \mathbf{M}_h \cdot \mathbf{p}_U^{true}$$

- Where
  - $\hat{\mathbf{p}}_U^{\mathrm{CC}} = (\hat{p}_U^{\mathrm{CC}}(y_1), \ldots, \hat{p}_U^{\mathrm{CC}}(y_n))^\top$
  - $\mathbf{M}_h \in \mathcal{R}^{n \times n}$ where $M_h[i,j] = Q(\hat{Y} = y_i | Y = y_j)$

- $\mathbf{M}_h$ is unknown, but we can get an estimate $\hat{\mathbf{M}}_h$ via k-fold cross-validation using L, so that:

$$\hat{\mathbf{M}}_h[i,j] = \frac{|\{(\mathbf{x}, y) \in L : h(\mathbf{x}) = y_i, y = y_j\}|}{|\{(\mathbf{x}, y) \in L : y = y_j\}|} \tag{1}$$

- ACC consists of solving this system, i.e., of correcting the class prevalence estimates $\hat{p}_U^{\mathrm{CC}}(y_i)$ obtained by CC according to the estimated system's bias:

$$\hat{\mathbf{p}}_U^{\mathrm{ACC}} = \hat{\mathbf{M}}_h^{-1} \cdot \hat{\mathbf{p}}_U^{\mathrm{CC}}$$

Further variants of ACC

- The system of linear equations ($\mathbf{p}_U^{CC} = \hat{\mathbf{M}}_h \cdot \mathbf{p}_U^{true}$) is sometimes unsolvable.

Bunse, M., On multi-class extensions of adjusted classify and count. LQ 2022.
Fernandes Vaz, Izbicki & Bassi Stern. Prior shift using the ratio estimator. JMLR 2019.

Further variants of ACC

- The system of linear equations ($\mathbf{p}_U^{CC} = \hat{\mathbf{M}}_h \cdot \mathbf{p}_U^{true}$) is sometimes unsolvable.
- Possible reasons:
  - The inverse $\hat{\mathbf{M}}_h^{-1}$ does not exist; this can happen when the classifier struggles to distinguish among 2 or more classes. Possible solutions:

Bunse, M., On multi-class extensions of adjusted classify and count. LQ 2022.
Fernandes Vaz, Izbicki & Bassi Stern. Prior shift using the ratio estimator. JMLR 2019.

Further variants of ACC

- The system of linear equations ($\mathbf{p}_U^{CC} = \hat{\mathbf{M}}_h \cdot \mathbf{p}_U^{true}$) is sometimes unsolvable.
- Possible reasons:
  - The inverse $\hat{\mathbf{M}}_h^{-1}$ does not exist; this can happen when the classifier struggles to distinguish among 2 or more classes. Possible solutions:
    - Use the Penrose pseudo-inverse

Bunse, M., On multi-class extensions of adjusted classify and count. LQ 2022.
Fernandes Vaz, Izbicki & Bassi Stern. Prior shift using the ratio estimator. JMLR 2019.

Further variants of ACC

- The system of linear equations ($\mathbf{p}_U^{CC} = \hat{\mathbf{M}}_h \cdot \mathbf{p}_U^{true}$) is sometimes unsolvable.
- Possible reasons:
  - The inverse $\hat{\mathbf{M}}_h^{-1}$ does not exist; this can happen when the classifier struggles to distinguish among 2 or more classes. Possible solutions:
    - Use the Penrose pseudo-inverse
    - Solve a constrained least squares problem

$$\hat{\mathbf{p}}_U^{ACC} = \underset{\mathbf{p} \in \Delta^{n-1}}{\arg\min} \, || \, \mathbf{p}_U^{CC} - \hat{\mathbf{M}}_h \cdot \mathbf{p} \, ||_2$$

Bunse, M., On multi-class extensions of adjusted classify and count. LQ 2022.
Fernandes Vaz, Izbicki & Bassi Stern. Prior shift using the ratio estimator. JMLR 2019.

Further variants of ACC

- The system of linear equations ($\mathbf{p}_U^{CC} = \hat{\mathbf{M}}_h \cdot \mathbf{p}_U^{true}$) is sometimes unsolvable.
- Possible reasons:
  - The inverse $\hat{\mathbf{M}}_h^{-1}$ does not exist; this can happen when the classifier struggles to distinguish among 2 or more classes. Possible solutions:
    - Use the Penrose pseudo-inverse
    - Solve a constrained least squares problem

$$\hat{\mathbf{p}}_U^{ACC} = \underset{\mathbf{p} \in \Delta^{n-1}}{\arg\min} \, || \, \mathbf{p}_U^{CC} - \hat{\mathbf{M}}_h \cdot \mathbf{p} \, ||_2$$

  - A solution exist, but is not feasible. For example, when some values fall outside the interval $[0, 1]$. Possible solutions:

Bunse, M., On multi-class extensions of adjusted classify and count. LQ 2022.
Fernandes Vaz, Izbicki & Bassi Stern. Prior shift using the ratio estimator. JMLR 2019.

Further variants of ACC

- The system of linear equations ($\mathbf{p}_U^{CC} = \hat{\mathbf{M}}_h \cdot \mathbf{p}_U^{true}$) is sometimes unsolvable.
- Possible reasons:
  - The inverse $\hat{\mathbf{M}}_h^{-1}$ does not exist; this can happen when the classifier struggles to distinguish among 2 or more classes. Possible solutions:
    - Use the Penrose pseudo-inverse
    - Solve a constrained least squares problem

$$\hat{\mathbf{p}}_U^{ACC} = \underset{\mathbf{p} \in \Delta^{n-1}}{\arg\min} \, || \, \mathbf{p}_U^{CC} - \hat{\mathbf{M}}_h \cdot \mathbf{p} \, ||_2$$

  - A solution exist, but is not feasible. For example, when some values fall outside the interval $[0, 1]$. Possible solutions:
    - Clipping and L1-normalize

Bunse, M., On multi-class extensions of adjusted classify and count. LQ 2022.
Fernandes Vaz, Izbicki & Bassi Stern. Prior shift using the ratio estimator. JMLR 2019.

Further variants of ACC

- The system of linear equations ($\mathbf{p}_U^{CC} = \hat{\mathbf{M}}_h \cdot \mathbf{p}_U^{true}$) is sometimes unsolvable.
- Possible reasons:
  - The inverse $\hat{\mathbf{M}}_h^{-1}$ does not exist; this can happen when the classifier struggles to distinguish among 2 or more classes. Possible solutions:
    - Use the Penrose pseudo-inverse
    - Solve a constrained least squares problem

$$\hat{\mathbf{p}}_U^{ACC} = \underset{\mathbf{p} \in \Delta^{n-1}}{\arg\min} || \mathbf{p}_U^{CC} - \hat{\mathbf{M}}_h \cdot \mathbf{p} ||_2$$

  - A solution exist, but is not feasible. For example, when some values fall outside the interval $[0, 1]$. Possible solutions:
    - Clipping and L1-normalize
    - Softmax

Bunse, M., On multi-class extensions of adjusted classify and count. LQ 2022.
Fernandes Vaz, Izbicki & Bassi Stern. Prior shift using the ratio estimator. JMLR 2019.

Further variants of ACC

- The system of linear equations ($\mathbf{p}_U^{CC} = \hat{\mathbf{M}}_h \cdot \mathbf{p}_U^{true}$) is sometimes unsolvable.
- Possible reasons:
  - The inverse $\hat{\mathbf{M}}_h^{-1}$ does not exist; this can happen when the classifier struggles to distinguish among 2 or more classes. Possible solutions:
    - Use the Penrose pseudo-inverse
    - Solve a constrained least squares problem

$$\hat{\mathbf{p}}_U^{ACC} = \underset{\mathbf{p} \in \Delta^{n-1}}{\arg\min} \, || \, \mathbf{p}_U^{CC} - \hat{\mathbf{M}}_h \cdot \mathbf{p} \, ||_2$$

  - A solution exist, but is not feasible. For example, when some values fall outside the interval $[0, 1]$. Possible solutions:
    - Clipping and L1-normalize
    - Softmax
    - Projecting $\hat{\mathbf{p}}_U^{ACC}$ the point to the simplex Δ^{n-1} (different methods)

Bunse, M., On multi-class extensions of adjusted classify and count. LQ 2022.
Fernandes Vaz, Izbicki & Bassi Stern. Prior shift using the ratio estimator. JMLR 2019.

PCC

- Probabilistic Classify and Count (PCC) consists of:
  1. generating a soft classifier s from L
  2. generating posterior probabilities for the items in U
  3. estimating $p_U(y_i)$ by counting the expected fraction of items predicted to be in y_i by s, i.e.:

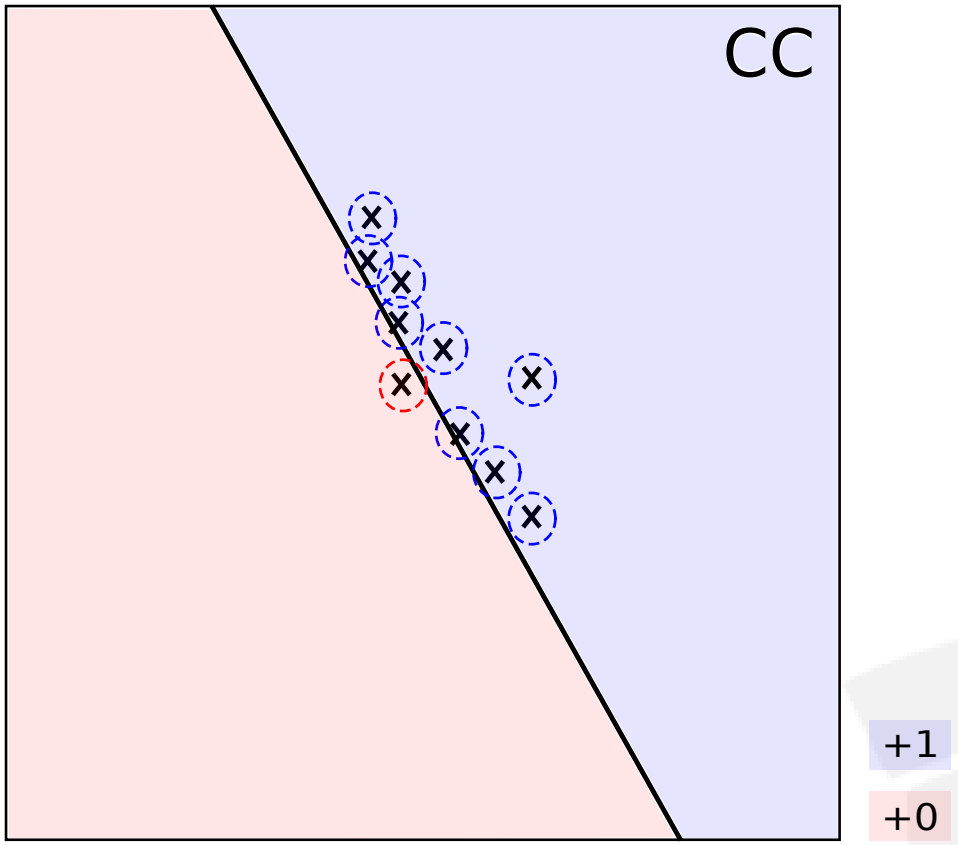
$$\begin{aligned}\hat{p}_U^{\text{PCC}}(y_i) &= E_U[\hat{Y} = y_i] \\ &\approx \frac{1}{|U|}\sum_{\mathbf{x}\in U} P(y_i|\mathbf{x}) \\ &= \frac{1}{|U|}\sum_{\mathbf{x}\in U} s_i(\mathbf{x})\end{aligned}$$

- Rationale: posteriors contain richer information than binary decisions.

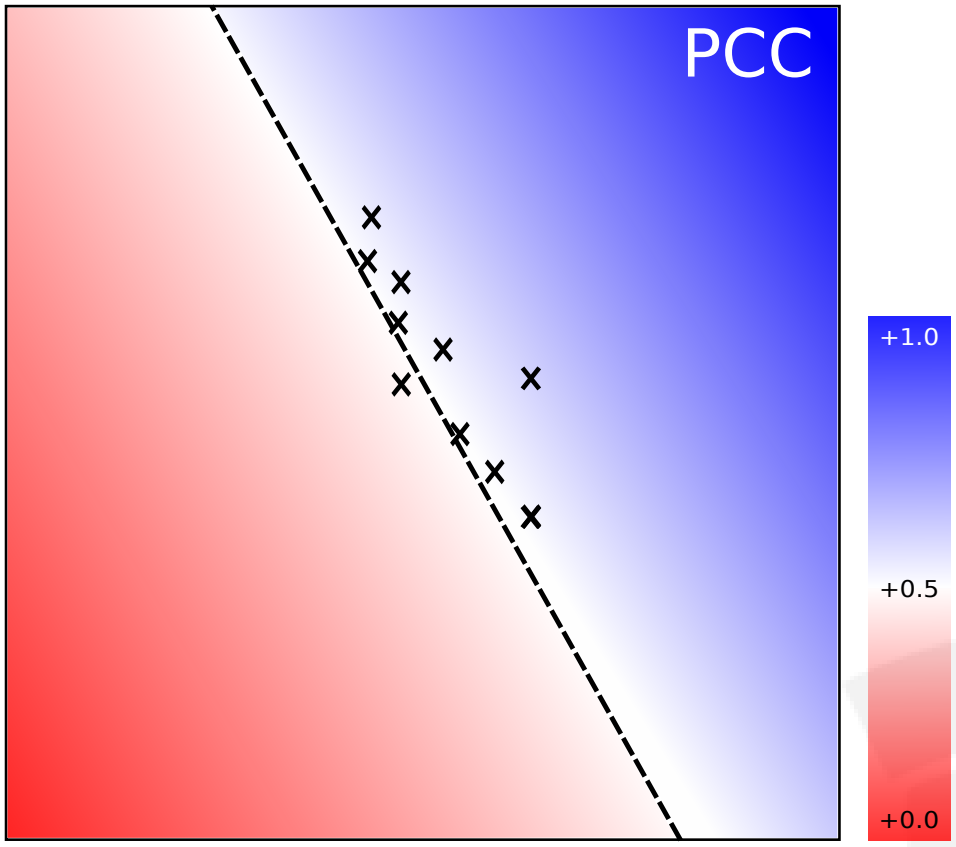
Bella, Ferri, Hernańdez-Orallo, Ramírez-Quintana. Quantification via probability estimators. ICDM 2010.

Lewis. Evaluating and optimizing autonomous text classification systems. SIGIR 1995.

CC vs PCC



CC vs PCC



PCC and Calibration

- PCC requires the classifier to return calibrated posterior probabilities $s_i(\mathbf{x}) = P(y_i|\mathbf{x})$ such that

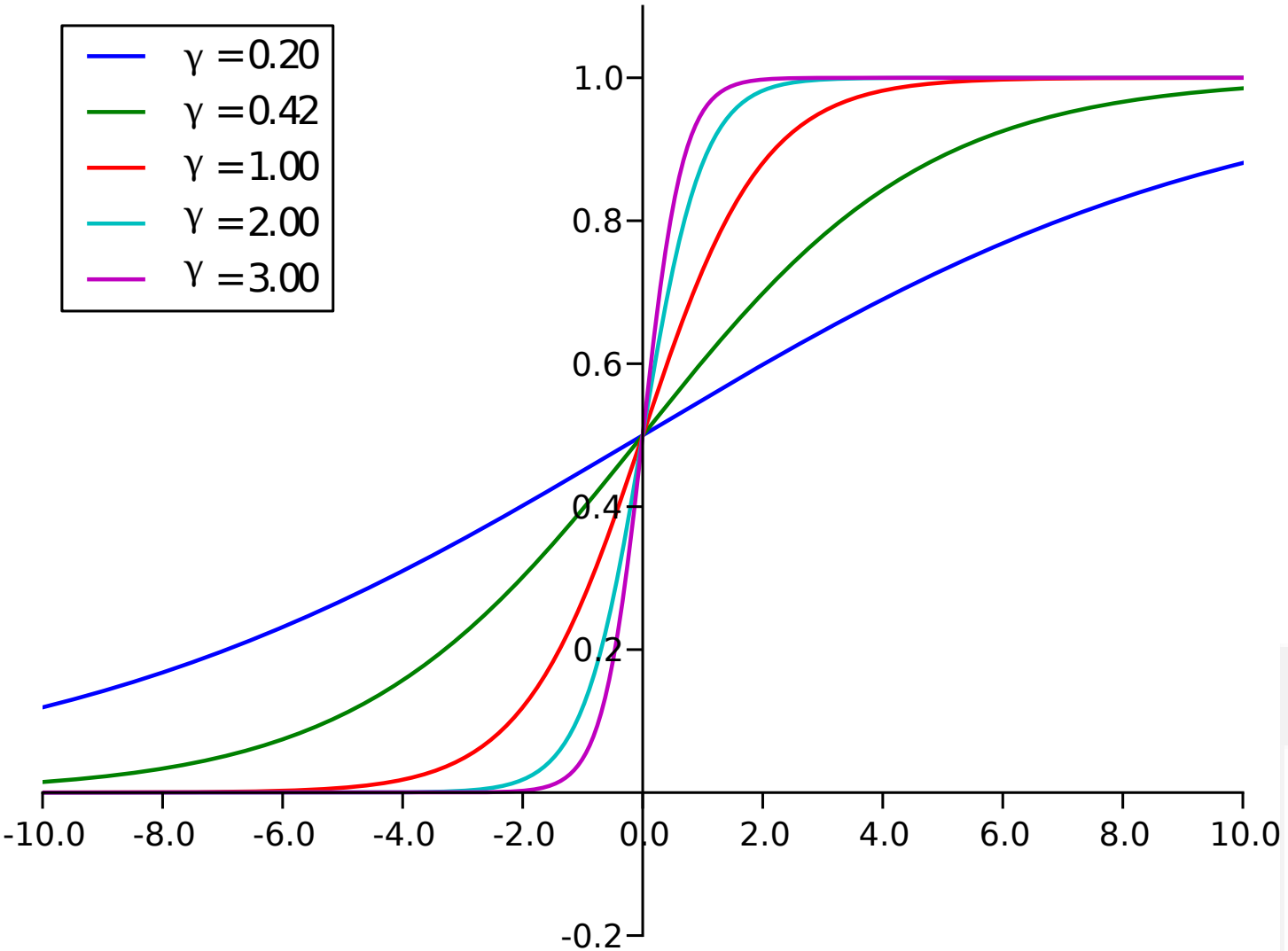
$$\lim_{|\sigma|\to\infty} \frac{|\{(\mathbf{x}, y) \in \sigma \mid s_i(\mathbf{x}) = \alpha, y = y_i\}|}{|\{(\mathbf{x}, y) \in \sigma \mid s_i(\mathbf{x}) = \alpha\}|} = \alpha \tag{2}$$

- E.g., 82% of the instances $\mathbf{x}$ for which $s_i(\mathbf{x}) = 0.82$, belong to y_i
- Confidence scores $s_i(\mathbf{x})$ that are not probabilities (e.g., SVMs) or are non-calibrated probabilities (e.g., NB) must be converted into calibrated posterior probabilities, e.g., by applying a sigmoidal (e.g., logistic) function

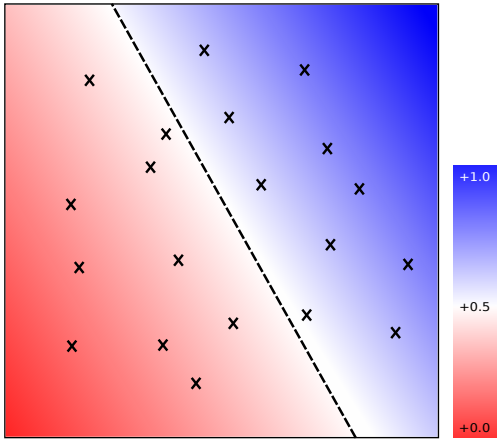
$$P(y_i|\mathbf{x}) = \frac{1}{1 + e^{\gamma s_i(\mathbf{x}) + \beta}}$$

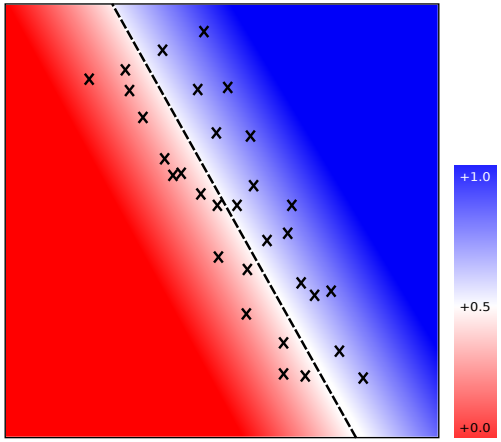
- Calibration consists in tuning γ and β so that the above holds

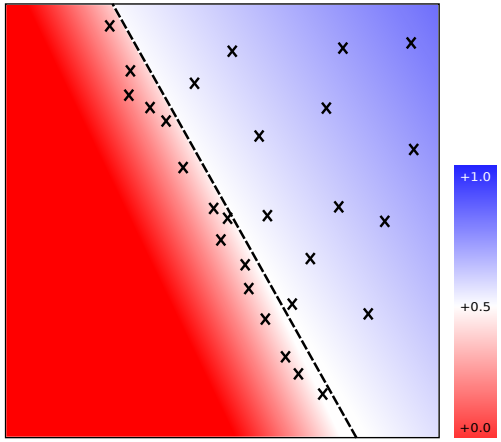
PCC and Calibration (cont'd)



PCC and Calibration (cont'd)







Calibration: Take-away Message

- It is not important you remember the technical details of the calibration function (there are many variants, actually).
- The important thing to remember is:
  - Calibration is a property defined with respect to a sample (let's call it σ)
  - σ is drawn IID from one distribution P
  - Assume we have a classifier s which is well-calibrated for σ
  - Assume σ' drawn IID from distribution Q
  - If P and Q are related through PPS then:

s cannot be well-calibrated for σ'

PACC: binary

- Probabilistic Adjusted Classify and Count (PACC) stands to ACC like PCC stands to CC.
- In the binary case ($\text{pos} = \{(\mathbf{x}, \oplus) \in U\}$ and $\text{neg} = \{(\mathbf{x}, \ominus) \in U\}$):
  - It uses $\hat{p}_U^{\text{PCC}}$ instead of $\hat{p}_U^{\text{CC}}$
  - It uses $\text{tpr}_s = \frac{1}{|\text{pos}|} \sum_{\text{pos}} s_\oplus(\mathbf{x})$ instead of tpr_h
  - It uses $\text{fpr}_s = \frac{1}{|\text{neg}|} \sum_{\text{neg}} s_\oplus(\mathbf{x})$ instead of fpr_h
- PACC then solves:

$$\hat{p}_\sigma^{\text{PACC}}(\oplus) = \frac{\hat{p}_\sigma^{\text{PCC}}(\oplus) - \text{fpr}_s}{\text{tpr}_s - \text{fpr}_s}$$

Bella, Ferri, Hernańdez-Orallo, Ramírez-Quintana. Quantification via probability estimators. ICDM 2010.

PACC: Multiclass

- Probabilistic Adjusted Classify and Count (PACC) stands to ACC like PCC stands to CC.
- In the multiclass case:
  - estimate $\mathbf{M}_s$ of a soft classifier s on L via k-fold cross-validation with:

$$\hat{\mathbf{M}}_s[i,j] = \frac{\sum_{\{(\mathbf{x},y)\in L\,:\,y=y_j\}} s_i(\mathbf{x})}{|\{(\mathbf{x},y)\in L\,:\,y=y_j\}|} \tag{3}$$

- PACC then solves a system of n equations with n unknowns:

$$\mathbf{p}_U^{\text{PACC}} = \hat{\mathbf{M}}_s^{-1}\cdot\mathbf{p}_U^{\text{PCC}}$$

Bella, Ferri, Hernańdez-Orallo, Ramírez-Quintana. Quantification via probability estimators. ICDM 2010.

EMQ

- An EM-based class prevalence estimation method for improving classification accuracy
- EMQ consists of an iterative, mutually recursive re-computation of the posteriors $p(y|\mathbf{x})$ and of the priors $p_U(y)$, until convergence
- Method originally devised for improving the posteriors $p(y|\mathbf{x})$. But if quantification is our goal we can use its "byproducts", i.e., the improved estimates of the priors $p_U(y)$.
- Note that EMQ observes U (and not only L) at training time. EMQ might thus be better described as a transductive algorithm.

Saerens, Latinne, & Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. Neural Computation 2002.

EMQ rationale

- A classifier calibrated on P is not calibrated for Q if PPS is at play.
- Bayes rule:

$$P(x|y_i) = \frac{P(y_i|x)P(x)}{P(y_i)} \qquad Q(x|y_i) = \frac{Q(y_i|x)Q(x)}{Q(y_i)}$$

EMQ rationale

- A classifier calibrated on P is not calibrated for Q if PPS is at play.
- Bayes rule:

$$P(x|y_i) = \frac{P(y_i|x)P(x)}{P(y_i)} \qquad Q(x|y_i) = \frac{Q(y_i|x)Q(x)}{Q(y_i)}$$

- Since $P(x|y_i) = Q(x|y_i)$ (PPS assumption):

$$Q(y_i|x) = \frac{P(x)}{Q(x)} \frac{Q(y_i)}{P(y_i)} P(y_i|x)$$

EMQ rationale

- A classifier calibrated on P is not calibrated for Q if PPS is at play.
- Bayes rule:

$$P(x|y_i) = \frac{P(y_i|x)P(x)}{P(y_i)} \qquad Q(x|y_i) = \frac{Q(y_i|x)Q(x)}{Q(y_i)}$$

- Since $P(x|y_i) = Q(x|y_i)$ (PPS assumption):

$$Q(y_i|x) = \frac{P(x)}{Q(x)} \frac{Q(y_i)}{P(y_i)} P(y_i|x)$$

- Since $\sum_{j=1}^{n} Q(y_i|x) = 1$:

$$\frac{P(x)}{Q(x)} = \left[\sum_{j=1}^{n} \frac{Q(y_i)}{P(y_i)} P(y_i|x) \right]^{-1}$$

EMQ rationale

- A classifier calibrated on P is not calibrated for Q if PPS is at play.
- Bayes rule:

$$P(x|y_i) = \frac{P(y_i|x)P(x)}{P(y_i)} \qquad Q(x|y_i) = \frac{Q(y_i|x)Q(x)}{Q(y_i)}$$

- Since $P(x|y_i) = Q(x|y_i)$ (PPS assumption):

$$Q(y_i|x) = \frac{P(x)}{Q(x)} \frac{Q(y_i)}{P(y_i)} P(y_i|x)$$

- Since $\sum_{j=1}^{n} Q(y_i|x) = 1$:

$$\frac{P(x)}{Q(x)} = \left[\sum_{j=1}^{n} \frac{Q(y_i)}{P(y_i)} P(y_i|x) \right]^{-1}$$

- The calibrated posterior for the test distribution is:

$$Q(y_i|x) = \frac{\frac{Q(y_i)}{P(y_i)} P(y_i|x)}{\sum_{j=1}^{n} \frac{Q(y_i)}{P(y_i)} P(y_i|x)}$$

EMQ (cont'd)

- We apply EM in the following way until convergence of the $\hat{p}^{(s)}(y)$:
  - **Step 0**: For each $y \in \mathcal{Y}$ initialize $\hat{p}_U^{(0)}(y) \leftarrow p_L(y)$
    For each $\mathbf{x} \in U$ initialize $p^{(0)}(y|\mathbf{x}) \leftarrow p(y|\mathbf{x})$
  - **Step s**: Iterate $s = 1, 2, \ldots$ until convergence:
    - **Step s(E)**: For each y compute:

$$\hat{p}_U^{(s)}(y) = \frac{1}{|U|} \sum_{\mathbf{x} \in U} p^{(s-1)}(y|\mathbf{x}) \tag{4}$$

    - **Step s(M)**: For each unlabelled item $\mathbf{x}$ and each y compute:

$$p^{(s)}(y|\mathbf{x}) = \frac{\frac{\hat{p}_U^{(s)}(y)}{\hat{p}_U^{(0)}(y)} \cdot p^{(0)}(y|\mathbf{x})}{\sum_{y \in \mathcal{Y}} \frac{\hat{p}_U^{(s)}(y)}{\hat{p}_U^{(0)}(y)} \cdot p^{(0)}(y|\mathbf{x})} \tag{5}$$

- Step s(E) re-estimates the priors in terms of the new posterior probabilities
- Step s(M) re-calibrates the posterior probabilities by using the new priors

EMQ (cont'd)

- We apply EM in the following way until convergence of the $\hat{p}^{(s)}(y)$:
  - **Step 0**: For each $y \in \mathcal{Y}$ initialize $\hat{p}_U^{(0)}(y) \leftarrow p_L(y)$
    For each $\mathbf{x} \in U$ initialize $p^{(0)}(y|\mathbf{x}) \leftarrow p(y|\mathbf{x})$
  - **Step s**: Iterate $s = 1, 2, \ldots$ until convergence:
    - **Step s(E)**: For each y compute:

$$\hat{p}_U^{(s)}(y) = \frac{1}{|U|} \sum_{\mathbf{x} \in U} p^{(s-1)}(y|\mathbf{x}) \tag{4}$$

    - **Step s(M)**: For each unlabelled item $\mathbf{x}$ and each y compute:

$$p^{(s)}(y|\mathbf{x}) = \frac{\frac{\hat{p}_U^{(s)}(y)}{\hat{p}_U^{(0)}(y)} \cdot p^{(0)}(y|\mathbf{x})}{\sum_{y \in \mathcal{Y}} \frac{\hat{p}_U^{(s)}(y)}{\hat{p}_U^{(0)}(y)} \cdot p^{(0)}(y|\mathbf{x})} \tag{5}$$

- Step s(E) re-estimates the priors in terms of the new posterior probabilities
- Step s(M) re-calibrates the posterior probabilities by using the new priors

EMQ (cont'd)

- We apply EM in the following way until convergence of the $\hat{p}^{(s)}(y)$:
  - **Step 0**: For each $y \in \mathcal{Y}$ initialize $\hat{p}_U^{(0)}(y) \leftarrow p_L(y)$
    For each $\mathbf{x} \in U$ initialize $p^{(0)}(y|\mathbf{x}) \leftarrow p(y|\mathbf{x})$
  - **Step s**: Iterate $s = 1, 2, \ldots$ until convergence:
    - **Step s(E)**: For each y compute:

$$\hat{p}_U^{(s)}(y) = \frac{1}{|U|} \sum_{\mathbf{x} \in U} p^{(s-1)}(y|\mathbf{x}) \tag{4}$$

    - **Step s(M)**: For each unlabelled item $\mathbf{x}$ and each y compute:

$$p^{(s)}(y|\mathbf{x}) = \frac{\frac{\hat{p}_U^{(s)}(y)}{\hat{p}_U^{(0)}(y)} \cdot p^{(0)}(y|\mathbf{x})}{\sum_{y \in \mathcal{Y}} \frac{\hat{p}_U^{(s)}(y)}{\hat{p}_U^{(0)}(y)} \cdot p^{(0)}(y|\mathbf{x})} \tag{5}$$

- Step s(E) re-estimates the priors in terms of the new posterior probabilities
- Step s(M) re-calibrates the posterior probabilities by using the new priors

EMQ (cont'd)

- We apply EM in the following way until convergence of the $\hat{p}^{(s)}(y)$:
  - **Step 0**: For each $y \in \mathcal{Y}$ initialize $\hat{p}_U^{(0)}(y) \leftarrow p_L(y)$
    For each $\mathbf{x} \in U$ initialize $p^{(0)}(y|\mathbf{x}) \leftarrow p(y|\mathbf{x})$
  - **Step s**: Iterate $s = 1, 2, \ldots$ until convergence:
    - **Step s(E)**: For each y compute:

$$\hat{p}_U^{(s)}(y) = \frac{1}{|U|} \sum_{\mathbf{x} \in U} p^{(s-1)}(y|\mathbf{x}) \tag{4}$$

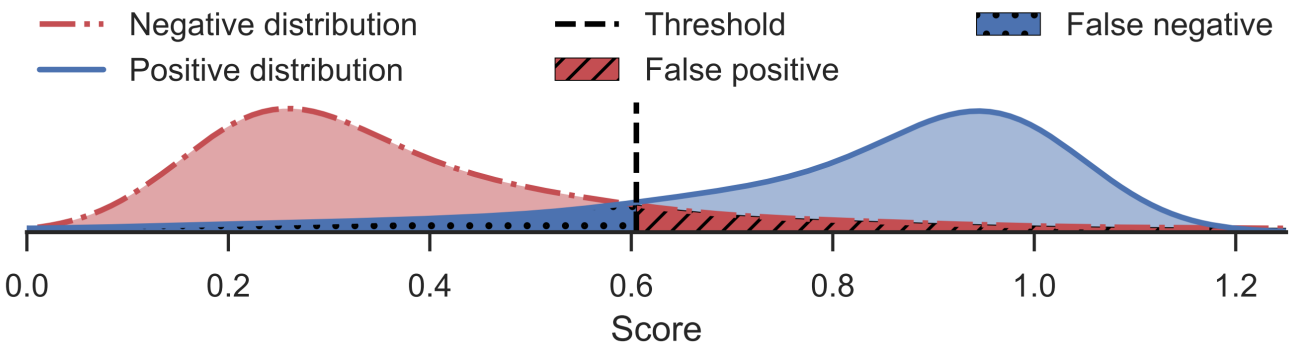
    - **Step s(M)**: For each unlabelled item $\mathbf{x}$ and each y compute:

$$p^{(s)}(y|\mathbf{x}) = \frac{\frac{\hat{p}_U^{(s)}(y)}{\hat{p}_U^{(0)}(y)} \cdot p^{(0)}(y|\mathbf{x})}{\sum_{y \in \mathcal{Y}} \frac{\hat{p}_U^{(s)}(y)}{\hat{p}_U^{(0)}(y)} \cdot p^{(0)}(y|\mathbf{x})} \tag{5}$$

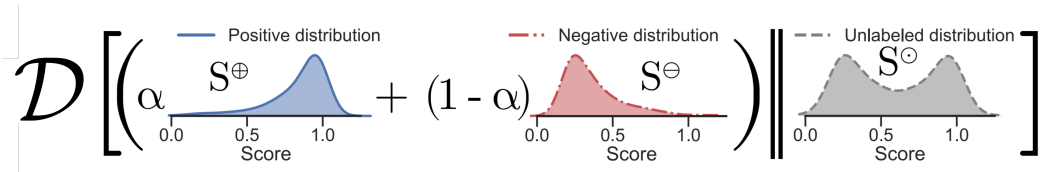
- Step s(E) re-estimates the priors in terms of the new posterior probabilities
- Step s(M) re-calibrates the posterior probabilities by using the new priors

Distribution Matching

- Training:



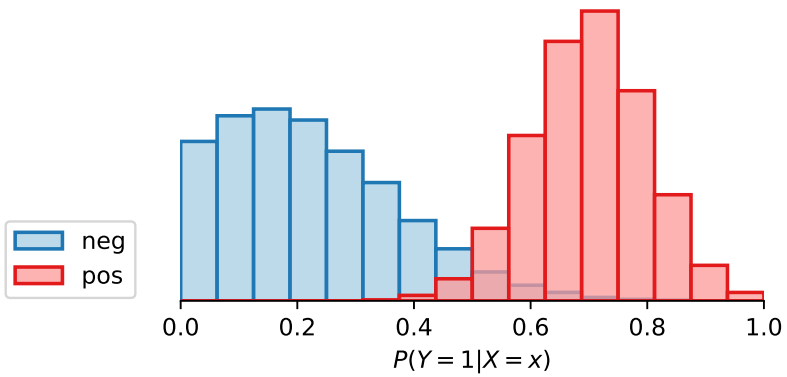
- Test:

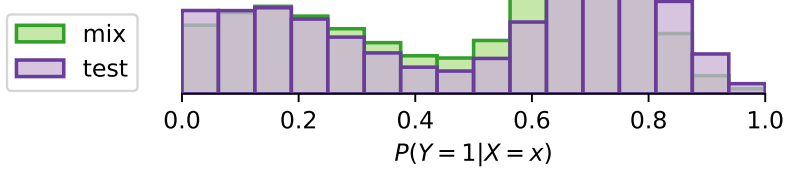


Images from: Hassan, Waqar, André Gustavo Maletzke, and Gustavo Batista. Pitfalls in Quantification Assessment. CIKM Workshops. 2021.

Distribution Matching: Hellinger Distance

- Density estimation of X is extremely difficult.
- Thanks to the PPS assumptions, we know $P(s(X)|Y) = Q(s(X)|Y)$, with s a soft classifier.
- In binary, we can consider only $s_{\oplus}(x) \approx P(Y = \oplus|X = x)$, since the negative one is $s_{\ominus}(x) = 1 - s_{\oplus}(x)$.
- We can model the density of $s_{\oplus}(X)$ using histograms.





González-Castro, V., Alaiz-Rodríguez, R., and Alegre, E. (2013). Class distribution estimation based on the Hellinger distance. Information Sciences, 218:146–164.

Distribution Matching: Hellinger Distance (cont'd)

- A histogram with b bins is represented by a list of values:
  - positives: $\mathbf{p}^{\oplus} = (p_1^{\oplus}, \ldots, p_{\mathrm{b}}^{\oplus})$ from the positives of L
  - negatives: $\mathbf{p}^{\ominus} = (p_1^{\ominus}, \ldots, p_{\mathrm{b}}^{\ominus})$ from the negatives of L
  - unlabelled: $\mathbf{q} = (q_1, \ldots, q_{\mathrm{b}})$ from U
- Use the Hellinger Distance between two discrete distributions $\mathbf{p}$ and $\mathbf{q}$

$$\mathrm{HD}(\mathbf{p}||\mathbf{q}) = \sqrt{1 - \sum_{i=1}^{b} \sqrt{p_i q_i}}$$

- HDy solves the following minimization problem:

$$p_U^{\mathrm{HDy}}(\oplus) = \arg\min_{0 \leq \alpha \leq 1} \mathrm{HD}((1-\alpha)\mathbf{p}^{\ominus} + \alpha \mathbf{p}^{\oplus} || \mathbf{q})$$

Distribution Matching: DyS

- In HDy the number of bins was explored in the range $(10, 20, \ldots, 110)$. All prevalence values are computed and the median is returned.
- It was later observed that the best number of bins typically is below 20; the number of bins should be an hyperparameter
- In the same paper, the authors considered the divergence function as another parameter (of which HD is one example).
- The DyS framework allowed to explore different divergence functions, and the Topsøe divergence was found to work better.
- Also, the search algorithm for α was improved: from brute force to ternary search.
- The framework was devised for binary-only quantification.

Maletzke, Moreira dos Reis, and Cherman, and Batista. DyS: A framework for mixture models in quantification. AAAI 2019.

A Framework for Multiclass Distribution Matching

- A generalized framework for multiclass quantification was proposed that takes the form:

$$\mathbf{q} = \mathbf{M}\mathbf{p}$$

- Where:
  - $\mathbf{q}$ is representation of the test data $\Phi(U)$
  - $\mathbf{M}$ is a matrix containing class-specific representations of the training data

$$\mathbf{M} = [\Phi(L_1), \ldots, \Phi(L_n)]$$

  - $\mathbf{p}$ is the sought prevalence vector
- Most distribution matching approaches (but not only) can be instantiated as a solution for

$$\mathbf{p}^* = \underset{\mathbf{p} \in \Delta^{n-1}}{\arg\min}\, \mathcal{L}(\mathbf{q}, \mathbf{M}\mathbf{p})$$

... by properly choosing the loss function and the representation function

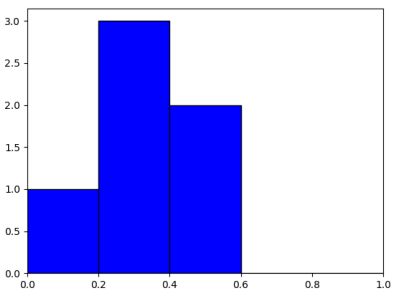
---

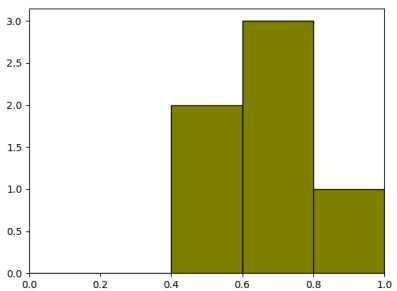
Firat. Unified framework for quantification. arXiv 2016.
Bunse. Unification of Algorithms for Quantification and Unfolding. INFORMATIK 2022.

Distribution Matching: Multiclass

- Framework: $\mathbf{q} = \mathbf{Mp}$
- Example: HDy takes
  - $\mathcal{L} =$ HD
  - $\Phi =$ histograms
- However, histograms are ill-defined in more than two classes.
- The trick for binary quantification is that a single histogram represents well the entire information, because $P(Y = \ominus|X) = P(Y = \oplus|X) - 1$.
- Assume we have $n = 2$ and $s(x) = (P(Y = \ominus|X = x), P(Y = \oplus|X = x))$.

$$A = \begin{cases} a_1 & = & (0.45, & 0.55) \\ a_2 & = & (0.42, & 0.58) \\ a_3 & = & (0.39, & 0.61) \\ a_4 & = & (0.35, & 0.65) \\ a_5 & = & (0.23, & 0.77) \\ a_6 & = & (0.10, & 0.90) \end{cases}$$





Distribution Matching: Multiclass (cont'd)

- In the multiclass case $\mathcal{Y} = \{1, \ldots, n\}$ we would need our Φ function to be $(n-1)$ histograms... right?
- No! Assume we have $n = 3$ and $s(x) = (P(Y = 1|X = x), P(Y = 2|X = x), P(Y = 3|X = x))$.

$$A = \left\{ \begin{array}{lcccc} a_1 & = & (0.1, & 0.2, & 0.7) \\ a_2 & = & (0.1, & 0.1, & 0.8) \\ a_3 & = & (0.2, & 0.3, & 0.5) \end{array} \right\} \quad B = \left\{ \begin{array}{lcccc} b_1 & = & (0.1, & 0.3, & 0.6) \\ b_2 & = & (0.1, & 0.2, & 0.7) \\ b_3 & = & (0.2, & 0.1, & 0.7) \end{array} \right\}$$

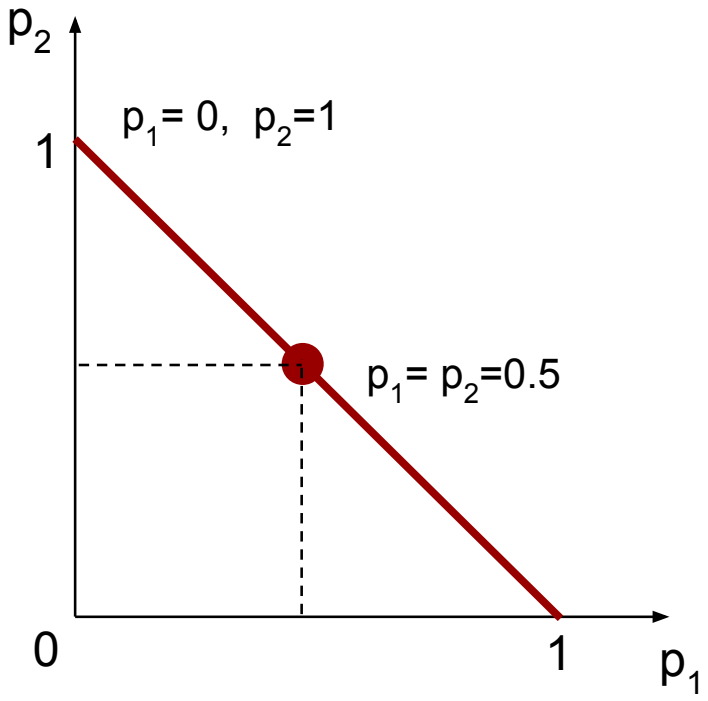
- We generate 3 class-wise histograms:

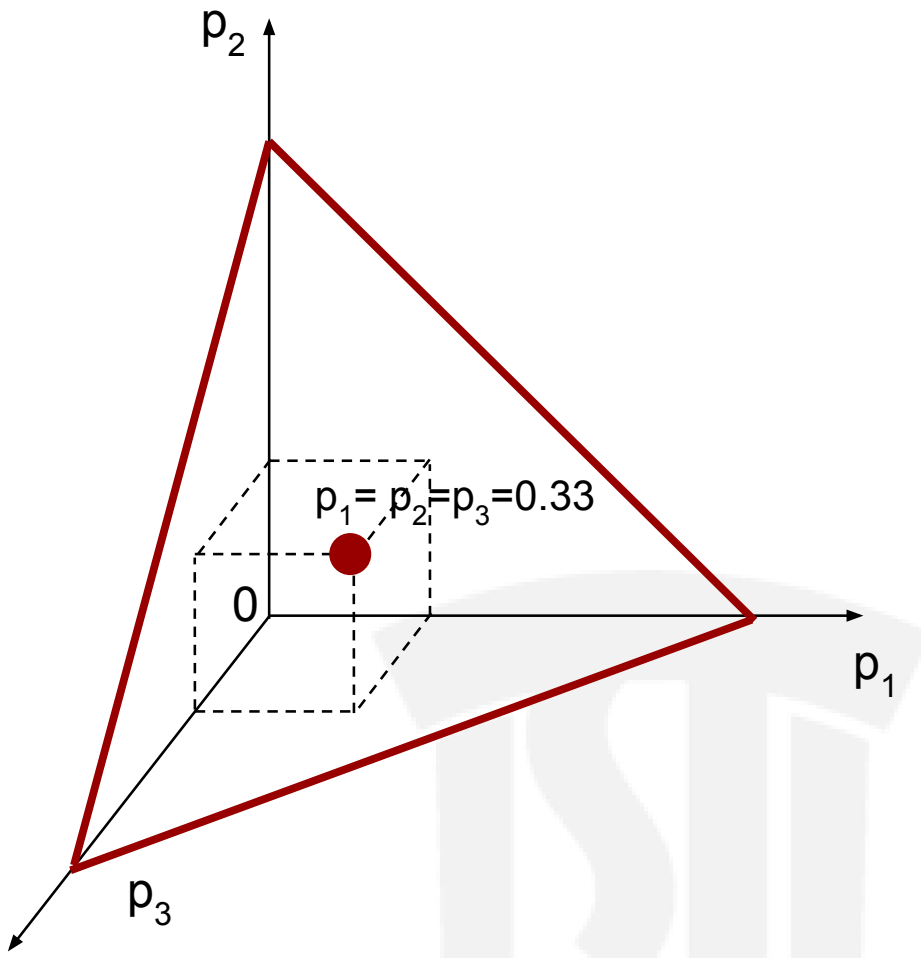
$$A' := \left\{ \begin{array}{l} H_1 = \text{hist}(\{0.1, 0.1, 0.2\}) \\ H_2 = \text{hist}(\{0.2, 0.1, 0.3\}) \\ H_3 = \text{hist}(\{0.7, 0.8, 0.5\}) \end{array} \right\} \quad B' := \left\{ \begin{array}{l} H'_1 = \text{hist}(\{0.1, 0.1, 0.2\}) \\ H'_2 = \text{hist}(\{0.3, 0.2, 0.1\}) \\ H'_3 = \text{hist}(\{0.6, 0.7, 0.7\}) \end{array} \right\}$$

- Note the following facts:
  1. $H_1 = H'_1$,
  2. $H_2 = H'_2$ since histograms are permutation-invariant functions,
  3. $H_3 \neq H'_3$: we need 3 histograms to distinguish between A and B!

Distribution Matching: Multiclass (cont'd)

- We know the posterior probabilities lie in Δ^{n-1} so we should at most use $(n-1)$ degrees of freedom... What is happening?





Distribution Matching: Multiclass (cont'd)

- Let's take a closer look, this time with $n = 4$. Consider A and B.

$$A = \begin{Bmatrix} a_1 & = & (0.1, & 0.2, & 0.3, & 0.4) \\ a_2 & = & (0.2, & 0.3, & 0.4, & 0.1) \\ a_3 & = & (0.3, & 0.4, & 0.1, & 0.2) \end{Bmatrix} \quad B = \begin{Bmatrix} b_1 & = & (0.1, & 0.3, & 0.4, & 0.2) \\ b_2 & = & (0.3, & 0.2, & 0.1, & 0.4) \\ b_3 & = & (0.2, & 0.4, & 0.3, & 0.1) \end{Bmatrix}$$

- The histograms we would obtain:

$$A' := \begin{Bmatrix} H_1 = \text{hist}(\{0.1, 0.2, 0.3\}) \\ H_2 = \text{hist}(\{0.2, 0.3, 0.4\}) \\ H_3 = \text{hist}(\{0.3, 0.4, 0.1\}) \\ H_4 = \text{hist}(\{0.4, 0.1, 0.2\}) \end{Bmatrix} \quad B' := \begin{Bmatrix} H'_1 = \text{hist}(\{0.1, 0.3, 0.2\}) \\ H'_2 = \text{hist}(\{0.3, 0.2, 0.4\}) \\ H'_3 = \text{hist}(\{0.4, 0.1, 0.3\}) \\ H'_4 = \text{hist}(\{0.2, 0.4, 0.1\}) \end{Bmatrix}$$

- Note that $A' \equiv B'$! The inter-class correlations are lost.

Distribution Matching: Multiclass via Density Estimation

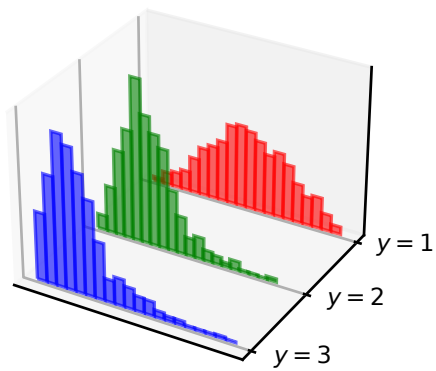
- Switch from histograms to Kernel Density Estimation.

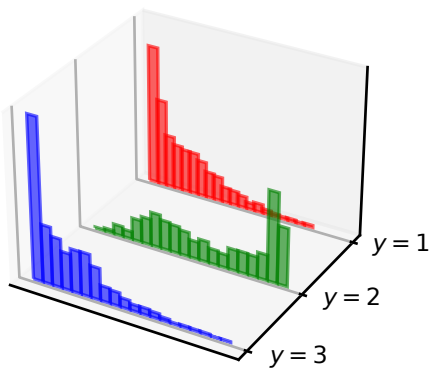
(A) Representation for class 1

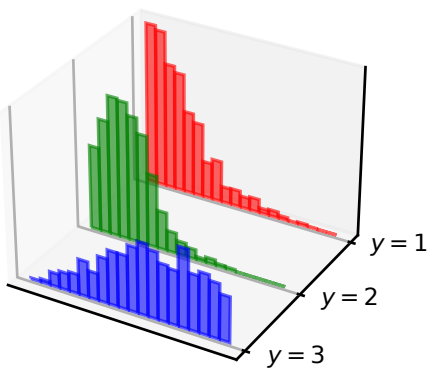
(B) Representation for class 2

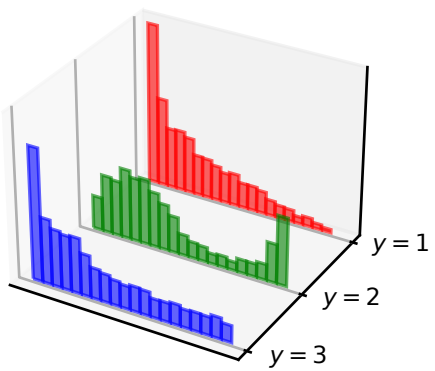
(C) Representation for class 3

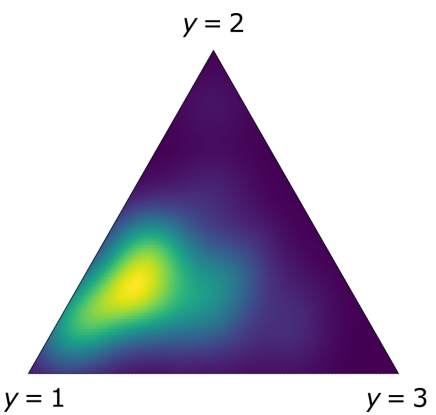
(D) Representation for test data

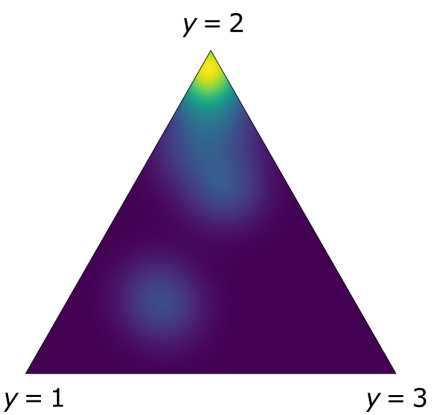


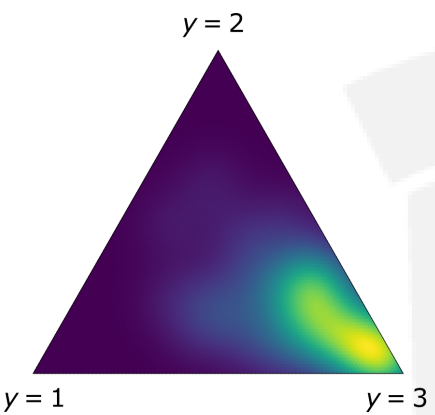


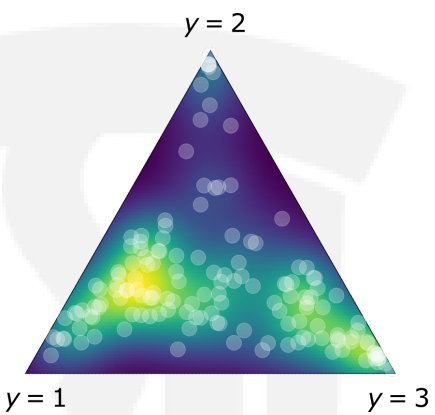












Distribution Matching: Multiclass via KDE

- A kernel density estimator (KDE) is given by

$$p(x) = \frac{1}{|X|} \sum_{x_i \in X} K\left(\frac{x - x_i}{h}\right)$$

- The density model for the posteriors $\widetilde{x} = s(x)$ of training data is a mixture of class-specific KDEs

$$\mathbf{p}_\alpha(\widetilde{x}) = \sum_{i=1}^{n} \alpha_i p_i(\widetilde{x})$$

- The density KDE of the (posteriors of the) test data is $q_U(\widetilde{x})$.
- The kernel is typically chosen to be the Gaussian kernel.

Distribution Matching: Multiclass via KDE

- The distribution matching problem of KDEy seeks to solve...

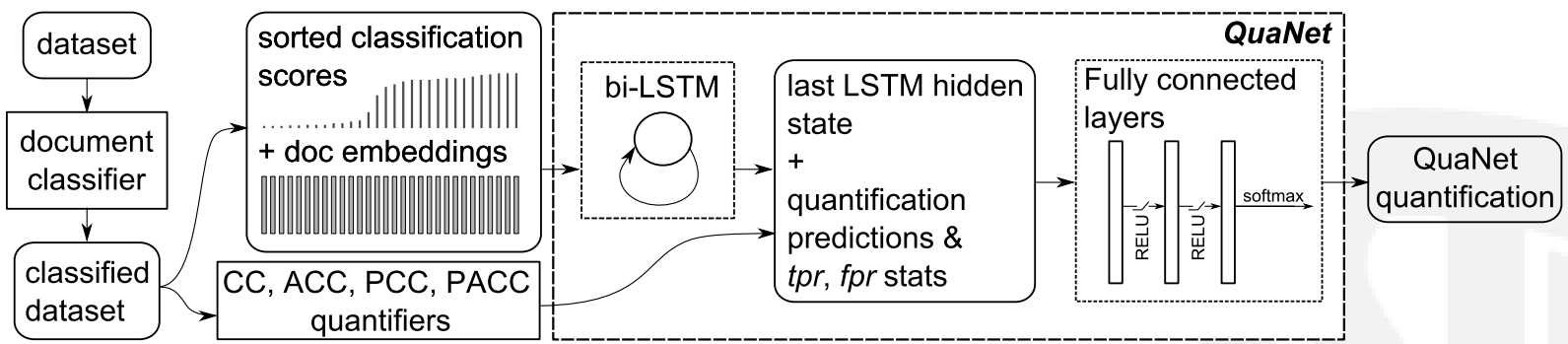
$$\hat{\alpha} = \underset{\alpha \in \Delta^{n-1}}{\arg\min}\, \mathcal{D}(\mathbf{p}_\alpha || q_U),$$

- ... but most divergences $\mathcal{D}$ involve dealing with an integral. This is computational costly.
- Proposed solutions for different divergences:
  - $\mathcal{D} =$ HD: Monte Carlo approximation
  - $\mathcal{D} =$ CS: close-form solution
  - $\mathcal{D} =$ KLD: a maximum likelihood solution
- The last one has been found to work better.

Moreo, González, del Coz. Kernel Density Estimation for Multiclass Quantification, arXiv 2024.

QuaNet

- QuaNet is a deep learning -based method for quantification
- The idea is to learn to produce higher-order quantification embeddings, i.e., an embedded representation of U from the:
  - observed posterior probabilites generated by a classifier
  - document embeddings
  - quantification predictions of simple aggregative methods
- QuaNet is trained across the full prevalence spectrum in order to learn how to adjust the counts it receives



Esuli, Moreo, & Sebastiani. A recurrent neural network for sentiment quantification. CIKM 2018.

Aggregative Quantification

Special-Purpose Learners

Explicit loss minimization

- Most methods use general-purpose classification algorithms
- Alternative: use special-purpose learning algorithms explicitly devised for quantification
- Idea: explicit loss minimization, directly optimize a quantification loss
- The loss functions most learners (e.g., AdaBoost, SVMs) can optimize must be linear, i.e., the error on the unlabelled set is a linear combination of the error generated by each unlabelled example
- Loss functions for quantification are instead nonlinear, i.e., the impact of the error on an unlabelled item depends on how the other unlabelled items have been classified

Esuli and Sebastiani. Sentiment quantification. *IEEE Intelligent Systems*2010.

Explicit loss minimization

- SVM$_{perf}$ is a structured output learning algorithm that can be optimized for arbitrary nonlinear / multivariate measures.
  - SVM(KLD) tailors SVM$_{perf}$ to use the Kullback-Leibler Divergence as a loss
  - SVM(Q) tailors SVM$_{perf}$ to use the Q-measure, a multi-objective measure (inspired by the Rijsbergen's F_β) defined as:

$$Q_\beta = \frac{(1+\beta^2)(M_C \cdot M_Q)}{\beta^2 M_C + M_Q} \tag{6}$$

where M_C is any evaluation measure for classification (here: *recall*) and M_Q is any evaluation measure for quantification (here: $|FP - FN|$)

Esuli, Sebastiani. Optimizing text quantifiers for multivariate loss functions. TKDD 2015.

Barranquero, Díez, del Coz. Quantification-oriented learning based on reliable classifiers. Pattern Recognition 2015.

Quantification trees

- Quantification trees are special-purpose decisions trees optimized for quantification; the basic idea is to use, in the learning phase, a measure of quantification as the splitting criterion at each node.
- Three different such measures were tested
  - (a proxy of) absolute error, i.e.,

$$D(p, \hat{p}) = \sum_{y_i \in \mathcal{C}} |\,\mathrm{FP} - \mathrm{FN}\,|$$

  - KLD
  - a "multiobjective" loss function, i.e.,

$$\begin{aligned} \mathrm{MOLF}(p, \hat{p}) &= \sum_{y_i \in \mathcal{Y}} |\,\mathrm{FP}_i^2 - \mathrm{FN}_i^2\,| \\ &= \sum_{y_i \in \mathcal{Y}} (\mathrm{FN}_i + \mathrm{FP}_i) \cdot |\,\mathrm{FN}_i - \mathrm{FP}_i\,| \end{aligned}$$

Milli, Monreale, Rossetti, Giannotti, Pedreschi, Sebastiani. Quantification Trees. ICDM 2013.

Non-aggregative Quantifiers

Methods that do not rely on classification

Vapnik's Principle and non-aggregative quantification

- Key observation: classification is a more general problem than quantification
- Vapnik's principle:

  "If you possess a restricted amount of information for solving some problem, try to solve the problem directly and never solve a more general problem as an intermediate step. It is possible that the available information is sufficient for a direct solution but is insufficient for solving a more general intermediate problem."

- This suggests solving quantification directly (without solving classification as an intermediate step, i.e., in a non-aggregative way) with the goal of achieving higher quantification accuracy than if we opted for the indirect solution

Dropping the assumptions of aggregative quantification

- Anti-causal learning: learning from causes to symptoms.
- Generation process of the type $Y \rightarrow X$
- In tasks of type $Y \rightarrow X$, one could try to directly model

$$P(x) = P(x|y)P(y) \tag{7}$$

- That is, one should avoid inferring $P(y|x)$ (akin to probabilistic classification) as an intermediate step.
- Example: Verbal autopsies (questionnaires about the symptoms of deceased people used by epidemiologists in countries with poor registration systems). Causes of death probabilistically determine symptoms

King and Lu. Verbal autopsy methods with multiple causes of death. Statistical Science 2008.

The ReadMe system

- ReadMe consists of estimating the class prevalence array $\mathbf{p}$ in the unlabeled set U, as defined (in matrix form) by:

$$\mathbf{X} = \mathbf{C}\mathbf{p} \tag{8}$$

where

  - $\mathbf{X}$ is a $2^K \times 1$ vector whose elements are the probability of each possible variate (binary vector) of K features
  - $\mathbf{C}$ is a $2^K \times |\mathcal{Y}|$ matrix where the j-th column has the class-conditional probabilities of all possible variates

- This poses a linear regression problem that could be resolved as:

$$\hat{\mathbf{p}} = (\mathbf{C}^\top \mathbf{C})^{-1} \mathbf{C}^\top \mathbf{X} \tag{9}$$

- ReadMe estimates $\mathbf{p}$ using $\hat{\mathbf{C}}$, modelled in L :

$$\hat{\mathbf{p}} = (\hat{\mathbf{C}}^\top \hat{\mathbf{C}})^{-1} \hat{\mathbf{C}}^\top \mathbf{X} \tag{10}$$

Hopkins and King. A method of automated nonparametric content analysis for social science. American Journal of Political Science 2010.

The ReadMe system (Cont'd)

- ReadMe thus has to deal with matrices with dimensions of the order of 2^K, which rapidly becomes intractable as K grows.
- To reduce the dimensionality of the matrix and to reduce variance, ReadMe applies bagging (i.e., takes random samples of $k = 5$ features) and averages the estimations. It relies on Bootstrapping to re-sample matrix rows and estimate the method variance.
- Later improvements:
  - iSA (Ceron et al. 2016) applies different heuristics to speed-up the method
  - `ReadMe2` (Jerzak et al. 2019) uses dense representations

Ceron, Curini, & Iacus. iSA: A fast, scalable and accurate algorithm for sentiment analysis of social media content. Information Sciences 2016.

Jerzak, King, & Strezhnev. An improved method for automated nonparametric content analysis for social science. 2019.

The HDx system

- There are distribution matching variants that do not rely on a classifier.
- One example is HDx for binary quantification problems.
- Given a matrix $\mathbf{X} \in \mathbb{R}^{m \times f}$ of m instances with f features, HDx generates a matrix $\mathbf{H} \in \mathbb{R}^{b \times f}$ of histograms with b bins per feature (columns).
- HDx generates one such matrix for:
  - $\mathbf{H}_{\oplus}$ for the positive items
  - $\mathbf{H}_{\ominus}$ for the negative items
  - $\mathbf{Q}$ for the test items
- For a given prevalence value α, the mixture $\mathbf{V}_{\alpha}$ is defined as

$$\mathbf{V}_{\alpha} = (1 - \alpha)\mathbf{H}_{\ominus} + \alpha\mathbf{H}_{\oplus}$$

- The HD is computed column-wise and the average is reported. The optimization problem comes down to

$$\alpha^* = \underset{0 \leq \alpha \leq 1}{\arg\min} \frac{1}{f} \sum_{i=1}^{f} \mathrm{HD}\left(\mathbf{V}_{\alpha}^{(i)} || \mathbf{Q}^{(i)}\right)$$

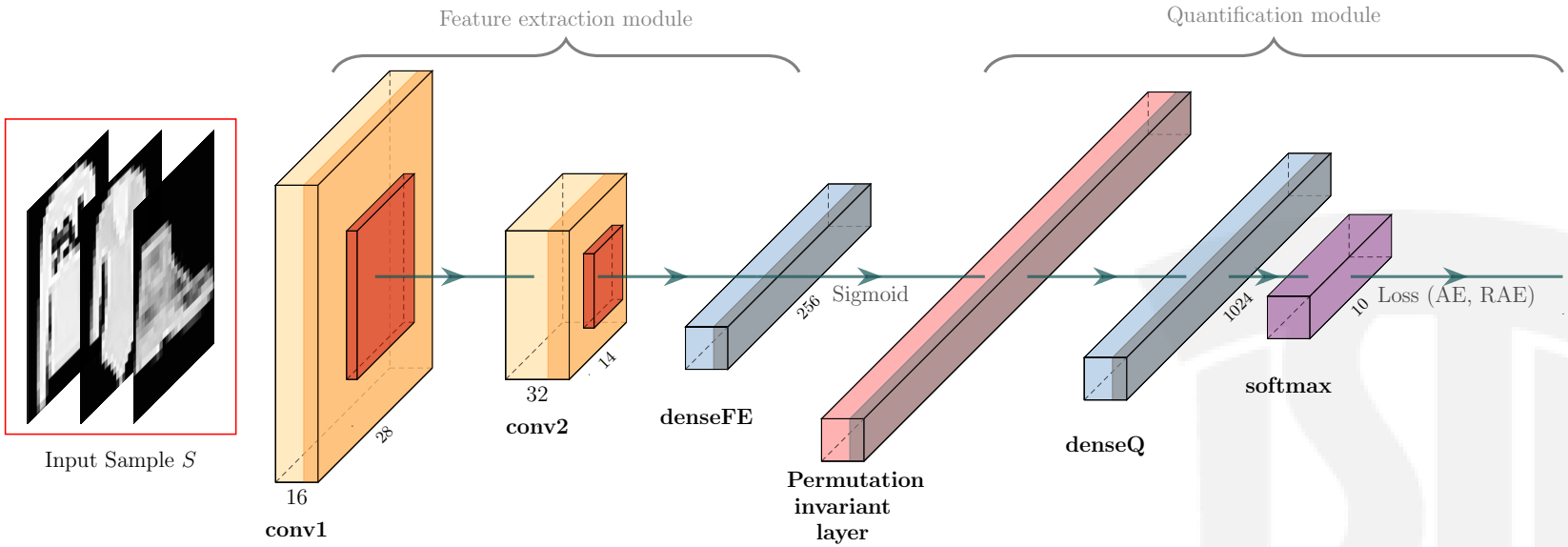
González-Castro, Alaiz-Rodríguez, & Alegre. Class distribution estimation based on the Hellinger distance. Information Sciences 2013.

Quantification as a Symmetric task

- Quantification has been regarded as an asymmetric task:
  - the training set $L = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$, instances labelled at the individual level
  - a test instance is a sample of individuals
  - a quantifier has to issue predictions at the aggregate level
- We can reframe the problem as a symmetric task by considering the training set be $D = \{(\sigma^{(i)}, \mathbf{p}^{(i)})\}_{i=1}^{m'}$ in which:
  - instance: $\sigma^{(i)} \in \mathcal{N}^{\mathcal{X}}$ is a bag (or multiset)
  - label: $\mathbf{p}^{(i)} \in \Delta^{n-1}$ is a vector of prevalence values
- In this way, the labels in the training set and the labels we need to predict, are both at the aggregate level.
- This is not restrictive, since from a dataset of "type L" we can create, via sampling, a dataset of "type D" (the opposite is not easy).
- Data for some problems (e.g., post-electoral results, demographic analysis, diagnosed diseases of regions) are indeed provided in this form.

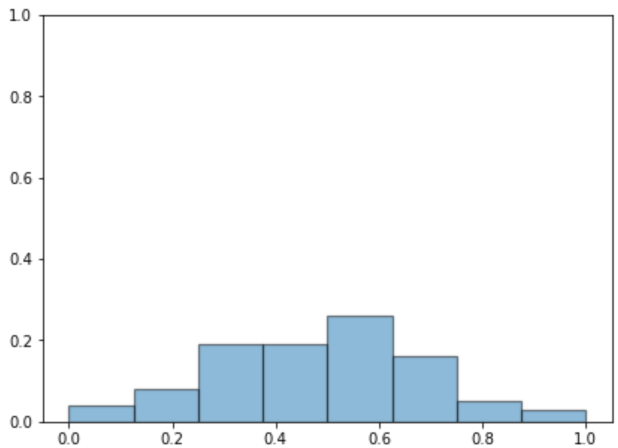
Symmetric Methods for Quantification

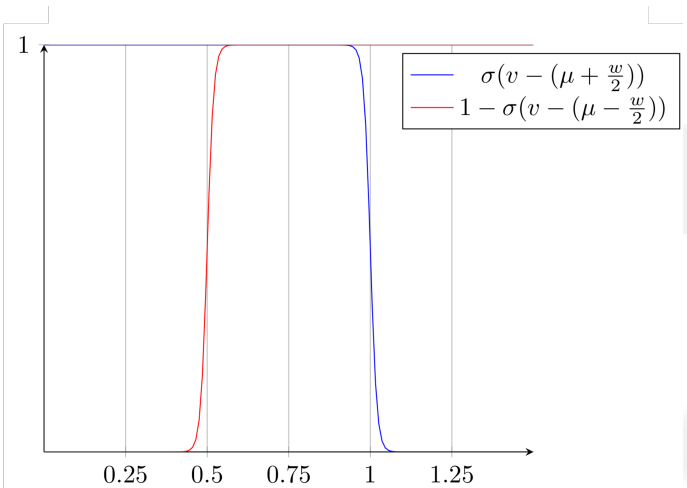
- Methods implementing the symmetric approach need to define a variable-size, permutation-invariant representation of the sample.
- A representation function Φ is said to be permutation invariant if $\Phi(\sigma) = \Phi(\pi(\sigma))$ for any permutation π.
- Examples: max, mean, median
- A possible general architecture:



HistNetQ

- HistNetQ implements this idea by means of histograms (one per dimension)
- Histograms are:
  - permutation-invariant
  - variable-size (if computed as "densities")
  - naturally geared towards counting
- However, histograms are not differentiable operators (required for training deep learning models)
- Differentiable approximations can be attained with pairs of sigmoids:





Meta Quantifiers

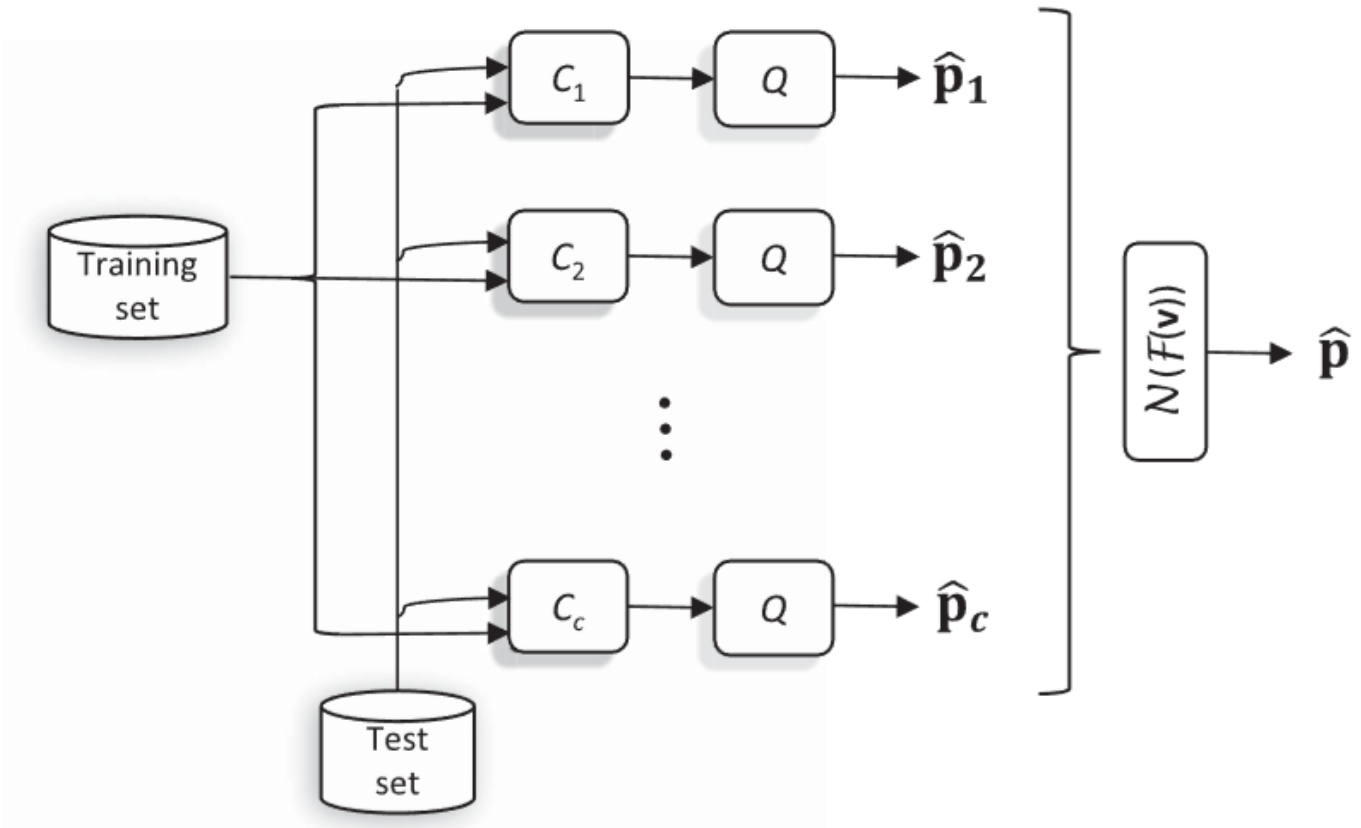
Quantifiers constructed on top of other base quantifiers

Meta-quantifiers

- Quantification methods that, in order to predict the class prevalence values, rely on the output of other quantifiers
- Pérez-Gállego et al. (2019) consider different base quantifiers, each trained in a bag of the training set characterized by a different prevalence value.
- At test time, all quantifiers issue a prevalence prediction, and the final output is derived via an aggregation of these:
  - Mean: simply average all predictions
  - Accuracy-based (static): at training time, estimate the accuracy of all base members and discard the least accurate ones
  - Training prevalence (dynamic): retains only members trained on samples that have a prevalence close to a preliminary estimate
  - Distribution similarity (dynamic): retains members trained on samples whose distribution of posteriors is closest, in terms of the HD, to the distribution of posteriors in the test sample

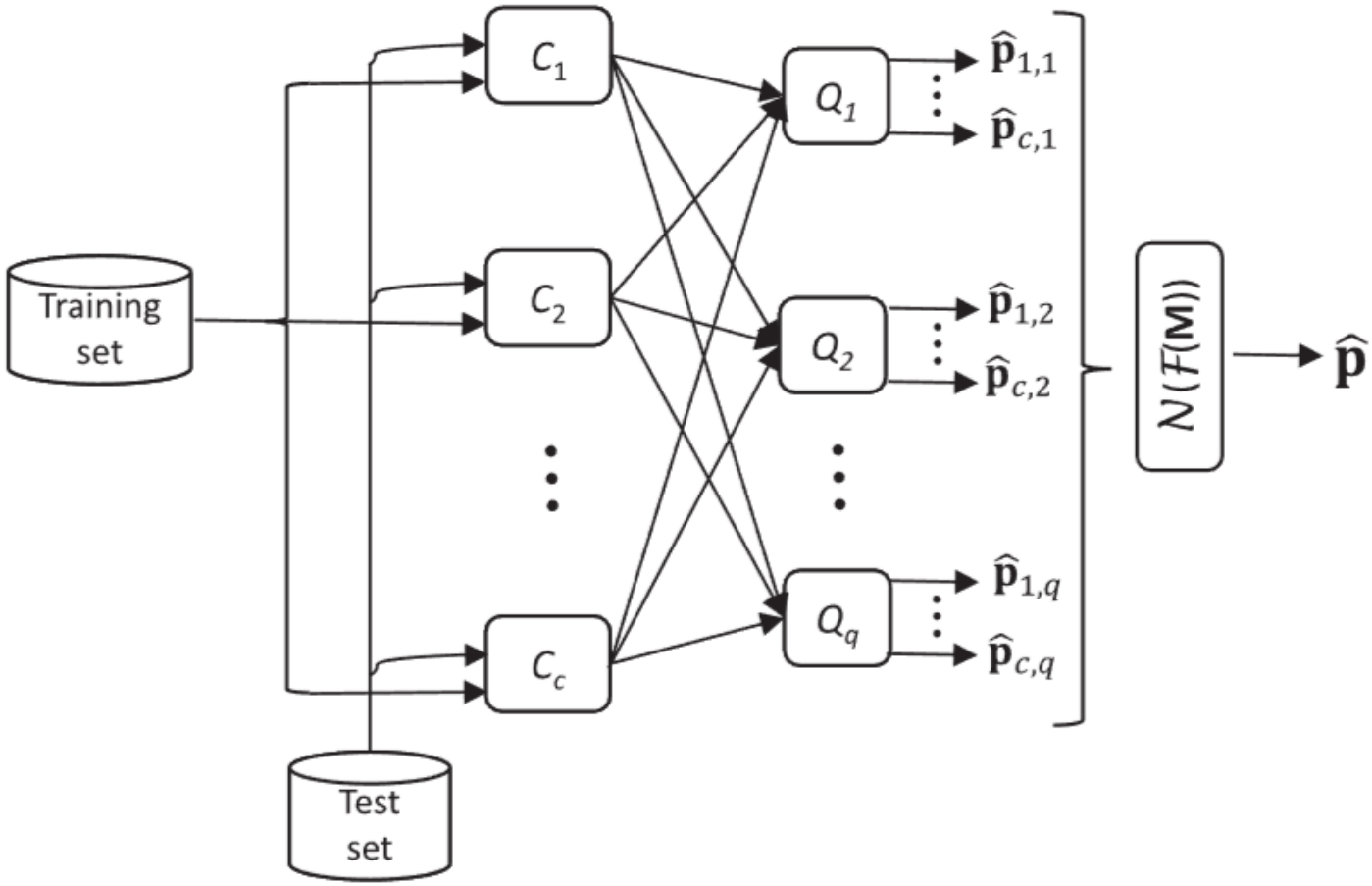
Pérez-Gállego, Castaño, Quevedo, and del Coz. Dynamic ensemble selection for quantification tasks. Information Fusion 2019.

MC-SQ



Donyavi, Serapiao, & Batista. MC-SQ and MC-MQ: Ensembles for Multi-class Quantification. TKDD 2024.

MC-MQ



Donyavi, Serapiao, & Batista. MC-SQ and MC-MQ: Ensembles for Multi-class Quantification. TKDD 2024.

Outline

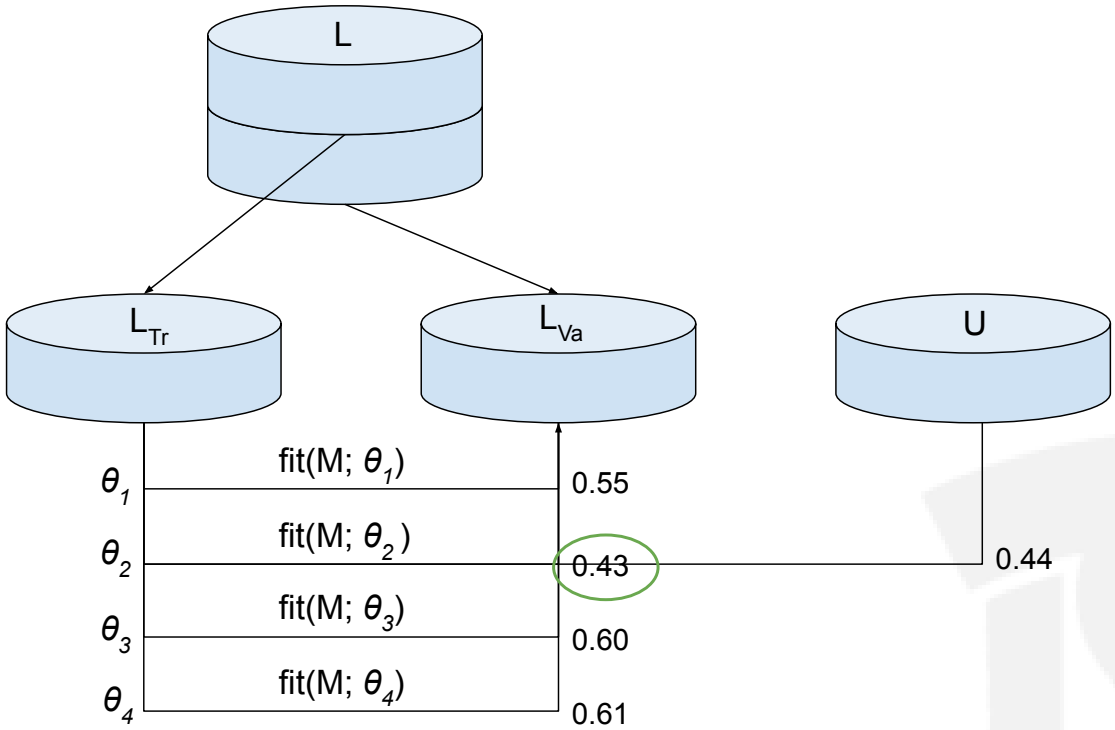
Model Selection in Quantification

- The performance of machine learning algorithms typically depends on how their hyperparameters are set.
- The process of hyperparameter optimisation is known as model selection, and consists of testing how well the model fares with different combinations of hyperparameters on held-out validation data.
- Model selection is inherently related to evaluation.
- Since quantification has specific evaluation measures and specific evaluation protocols, model selection should be in agreement with these.

Moreo & Sebastiani. Re-assessing the "classify and count" quantification method. ECIR 2021.

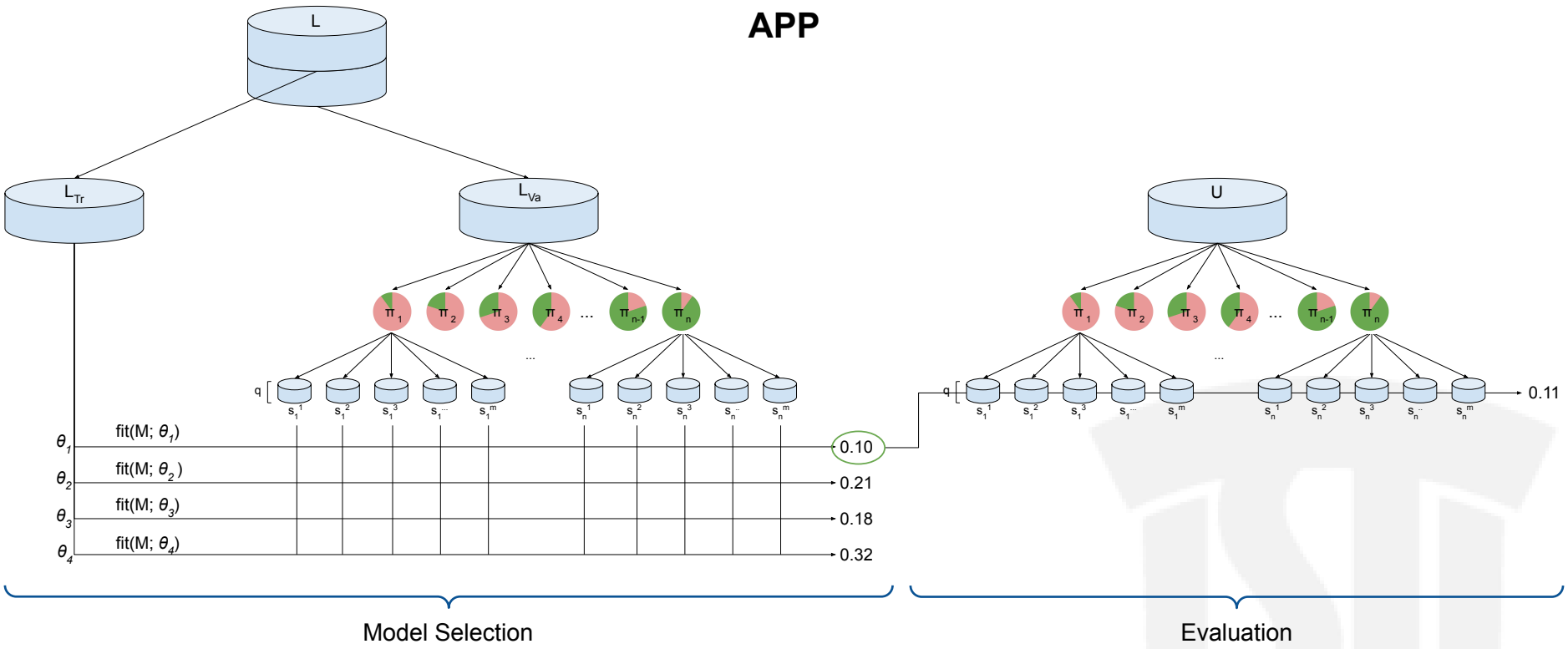
Model Selection in Quantification

- Many papers have instead carried out model selection mimicking the classification approach, i.e.:



Model Selection in Quantification

- This is theoretically flawed: model selection has to be carried out following a quantification-oriented evaluation protocol:



Outline

Conclusion

- Most methods in the literature are of type aggregative. Most of these try to mitigate the bias of the underlying classifier.
  - Are there better representation mechanisms that are not tailored to classification?
- Non-aggregative methods seem a more direct approach, and better principled, but are less studied.
  - Non-aggregative solutions are interesting in fields like fairness, since a classifier is an "user profiler".
- Quantification is a task in its own right.
  - Model selection has to target a quantification-oriented loss.

Thank you!

Questions?

For any question, contact us at

alejandro.moreo@isti.cnr.it

and

fabrizio.sebastiani@isti.cnr.it