Calibration → model that provides reliable confidence estimates, that allow us to make optimal decision in real world applications

→ Our model gives us score!

→ class posteriors → probability that tell us how likely it is that a given point belong to each class

How we know if a model is well calibrated?

act DCF — min DCF

cost using our decision threshold — cost with optimal threshold

well calibrated if this value are close

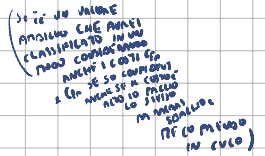
Why there is poor calibration?

Model without clear Probabilistic interpretation

→ SVM → score that represent distance

Overfit → Over confident model → overfit to train data → unrealistic posterior probabilities

output value close to 0 or 1 even in difficult case



Discriminative classifier

→ learn distinguish class

→ no learn data distribution

→ output posterior probabilities

Generative Classifier

→ model data distribution

objective → choose appropriate thresholds for making decision

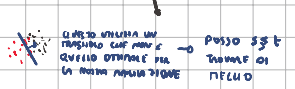
→ calibration is here

→ Poor calibration

→ good discriminant ability but poorly calibrated scores

→ sensitive to prior probability

→ no need to retrain model



assigne higher score to sample for correct class e lower to sample from incorrect class → good discriminant ability → Poor Discrimination → change classifier

Over-regularized Underconfident

→ too much regularization to combat overfitting

→ push probabilities toward 0,5 even when we have strong evidence for one class

Distribution Mismatch

→ data used for training differs sistematically from the reality

→ Poor discrimination

→ Miscalibration

For example the gaussian model for the project data is able to well discriminate the samples of two classes arranged in this way.

No anche se l'accuracy è buona la stima di discriminazione

SOMIGLIA MOLTO HIGH DENSITY AREA DOVE NON CE' UN GRUPPO DI DATO, QUINDI STIAMO I DATI INSIEME LA DENSITÀ RISULTERÀ PIÙ BASSA

QUESTO INVECE È UN GENERATIVE MODEL CHE NON RIESCE A FARE DISCRIMINATION

Nota: IL MODELLO È TARATO PER UNA SPECIFICA TASK DI CLASSIFICAZIONE, MA POTREBBE NON ESSERE LA MIA

How we select a good threshold → We can treat the decision threshold as a hyperparameter that can be optimize in validation set

PROBLEMA DI PRIMO CON IL NOSTRO SR (recognizer)

$x_t \to s(x_t) \lessgtr t \to$ CHE THRESHOLD USIAMO?

SE QUELLO THEORICO FA CALARE, DEGRADE I NOSTRI DATI NON SONO CALIBRATI BENE, NE TROVIAMO UNO CHE LAVORA MEGLIO

USIAMO minDCF TRATTANDO IL THRESHOLD COME UN PARAMETRO QUALUNQUE

QUINDI ABBIAMO

[Training set] [Val set] → [App Data]

IL THRESHOLD QUANDO DEVE ESSERE A QUESTO

minDCF → (t)

IL THRESHOLD CHE HA LE MIGLIORI PERFORMANCE SU VALIDATION SET

SE IL VALIDATION SET È SIMILE A L'APPLICATION SET ALLORA VA BENE

$t_{val} \approx t_{app}$

Problem

- Overfitting to Validation Set
  - validation data used to optimize multiple hyperparameters
    - data no truly unseen → overconfident
- Changing application contexts
  - optimal threshold depend heavily on specific application → prior probabilities and error cost.
    - we need to recompute minDCF ... (repeat the process)
- Fragility Problem
  - poorly calibrated score are sensitive
    - miss calibrated score
      - a small shift in the prior probability might require a completely different threshold

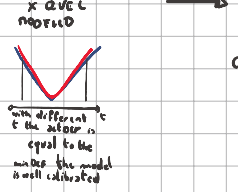
SE X ESEMPIO CONSIDERATO VA LOGISTIC REGRESSION WEIGHTED NON ALLENATO X UN APPLICATION CHE È SIMILE A QUELLA TRAINET $\tilde{\pi}_{train} \times t_{app}$ SE E APPROSSIMATA SIMILE LE COSE FUNZIONANO, SE CAMBIANO LA $\tilde{\pi}[prior]$ DOBBIAMO RICALCOLARE IL MODEL, IL THRESHOLD CAMBIA MOLTO, O COMUNQUE CE UNA BUONA RICALIBRATION, CAMBIANO SPESSO IL THRESHOLD

Solution? We change approach

we want well-calibrated score from the start so that

- Predictive Thresholds
  - optimal threshold → $\log(\pi(1-\pi))$ [if C_{fp} and $C_{fn}=1$]
- Application Independence
- No-pre-Application Tuning
  - no need to optimize threshold → well calibrated model

SE IL MODELLO È OTTIMO UTILIZZO IL DATA SOLO PER TROVARE IL THRESHOLD E IL MODELLO PERFORMA MOLTO MEGLIO CON QUESTO MODELLO



if we can't achive perfect calibration in the initial training

- we apply post-processing techniques to improve calibration → learn a mapping from the original scores to new scores
  - more like genuine log-likelihood ratios or calibrated probabilities
    - $Scal = f(s)$ transforming score

Information Preservation

↓

preserve relative ranking of the score

↓

score A > score B

↓

Monotonic function

↓

Le calibration doesn't destroy the discriminative information of the original classifier

Meaningful Output

↓

the transformed scores should

→ raw score into calibrated posterior probability $P(C|s)$ → then we can convert into llr easily

→ log-likelihood ratios → transform raw scores directly into calibrated log likelihood ratio

Hence like

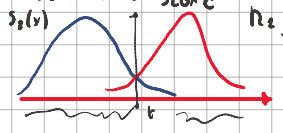
How we do this?

↓

Parametric Calibration Through Score Modeling

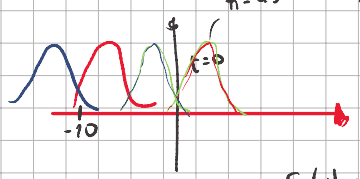
↓

Score well calibrated are score that can be trasholed in a good way, this means that if we have a $\pi=0.5$ and a model that gives us well calibrated score



Ne, se cambio l'application prima mi posso poi cambiare il trashold in modo appropriato

n_2 è ben calibrato ma se considerassi n_2 $S_2(t)$



$S_2(x) = S_2(x) - 10$

x cdf è implementato n_2 rimuove 10 da tutti gli score

$S_2(x)$ ha lo stesso shape ma è localizzato a -10, quindi non ho la stessa cosa perché considerando in maniera corretta tutti i punti della classe nostra e sbagliando i pesanti della classe blu.

L'optimal threshold non è più il teorico ma -10, t=0 I will do very bad

so possiamo trovare un modo che riallinea i nostri score, $n_2^{[a]} = S_2(x)+10$ così è well calibrated.

ciò si può fare usando modelli della score, andando così si comporta e lo trasforma in maniera tale da riallinearlo

↓

in this case $n_2^{[a]} = S_2(x_E) + \beta$ se so trovare questa linear trasformata ho i pesanti giusti

We can model the much simpler one-dimensional space of classifier score

↓

We treat score as datapoint and we model their distributions separately for each class

Practical Implementation

①

Calibration: prior-weighted logistic regression

We consider the non-calibrated scores as 1-D feature vectors

We assume an affine mapping from non-calibrated scores to calibrated scores

$$f(s) = \alpha s + \gamma$$

Since $f(s)$ should produce well-calibrated LLRs, $f(s)$ can be interpreted as the log-likelihood ratio for the two class hypotheses

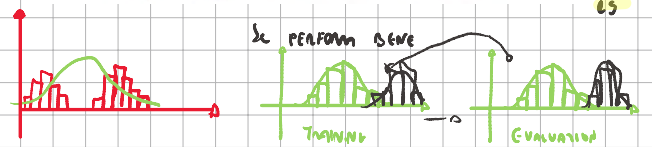
$$f(s) = \log \frac{f_{S|C}(s|\mathcal{H}_T)}{f_{S|C}(s|\mathcal{H}_F)} = \alpha s + \gamma$$

The class posterior probabilities for prior $\tilde{\pi}$ correspond to

$$\log \frac{P(C=\mathcal{H}_T|s)}{P(C=\mathcal{H}_F|s)} = \alpha s + \gamma + \log \frac{\tilde{\pi}}{1-\tilde{\pi}} = \alpha s + \beta$$

L'IDEA È QUINDI QUELLA DI PROVARE MODELLI CHE IMPLICITAMENTE ABBIA WELL CALIBRATED SCORE (CHE POSSIAMO ANCHE FARE MA CI USIAMO IN MODO TALE CHE NON VIOLINO)

X I MODELLI GENERATIVI, QUESTI POSSONO AVERE SCORE NON CALIBRATI WHEN THERE IS MISMATCH BETWEEN THE TRAINING AND THE APPLICATION DATA O NON FITTANO BENE I DATI ES



SE QUELLO CHE HO IMMAGINO FITTA ANCHE BENE L'EVALUATION DATA, IL MODEL WILL BE CALIBRATED WELL

BUT SE ALLENIAMO IL MODELLO SUI NOSTRI SAMPLE, QUESTI SONO SOLITAMENTE SMALL DIMENSIONAL, QUINDI AVERE UN BUON FIT È DIFFICILE NON È COME IL CASO UNIDIMENSIONALI NON ABBIAMO NECESSARIE UN NUMERO DI SAMPLE SUFFICIENTI A CAPIRE

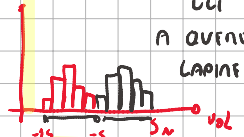
LE PROBLEMA È DESCRIVE LE FEATURE SE 6 I MODELLI CAMBIANO FANNO CACARE

COME FARE? SE NON CONSIDERIAMO LA POSSIBILITÀ DI MISMATCH, E CONSIDERIAMO 1 DIMENSIONAL DATA OTTENIAMO DELLI SCORE TALI ARE WELL CALIBRATED

IL NOSTRO $R = S_m = \log \frac{f(x|H_T)}{f(x|H_F)}$ (UNCALIBRATED) (SM) NON È CALIBRATO QUINDI EFFETTIVAMENTE NON SO QUANDO È CHE ATTRIBUIRE UN DETERMINATO VALORE AD UNA CLASSE POSITIVA O NEGATIVA

$S_m \gtrless t$? TRASFORMO IL PROBLEMA NEL TROVARE UNA PROBABILITÀ PER CUI SO QUANDO I SAMPLE MAPPATI SIANO UNA CLASSE NEGATIVA O UNA POSITIVA.

HO UN SET DI VALIDATION X CUI SCORE A QUESTO PUNTO POSSO CAPIRE SE IL SAMPLE HA UN S_m X LA CLASSE O NEGATIVA



TRAINING $R = 0$ $x_0 \cdots x_n$ TEST SCORES $s_{t0} \cdots s_{tn}$

$x_6 \to S_m = 0.5 \to$ LAVORANDO IN SET DI VALIDATION È PLAUSIBILE CHE APPARTENGA ALLA CLASSE NEGA

②

INVECE DI USARE IL THRESHOLD, USO IL SET DI VALIDATION (IL THRESHOLD NON SO COME USARLO LA MISS CALIBRATION HA SENSO NEI SCORE MA NON SO IN CHE DIREZIONE) SE HO UN SET DI VALIDATION SO IN CHE SITUAZIONE MI TROVO,

NOI NON GUARDIAMO L'ISTOGRAMMA, MA FACCIAMO

$f(S_m|H_T)$ FITTIAMO DELLE DISTRIBUTION SULL'ISTOGRAMMA E POI TROVIAMO LA LIKELIHOOD O QUANTO UNO APPARTIENE ALLA CLASSE POSITIVA O NEGATIVA.

$f(S_m|H_F)$

$\log \frac{f(S_m|H_T)}{f(S_m|H_F)}|_{S_m}$ IS A LLR THAT DESCRIBE THE DISTRIBUTION OF THE SCORE. SO I HAVE TO CREATE THIS MODEL ON TO SCORE USING A DATASET AND COMPARE THE LLR TO THAN COMPARE THE CLASSIFICATION.

IN THIS CASE THE MODEL IS WELL CALIBRATED NOW

COME FUNZIONA TUTTO ORA??

Training sample $x_0 \cdots x_n$ → DA CUI COSTRUIAMO R CHE DÀ SCORE UNCALIBRATED S_m

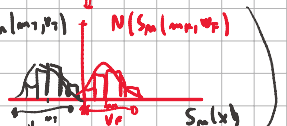
Validation set CALCOLIAMO UNCALIBRATED SCORE SU THE VALIDATION SET $x_0^V \cdots x_n^V$ → $S_m(x_0^V)$ $S_m(x_n^V)$

ES $R \to N(x|\mu_T, \Sigma_T), N(x|\mu_F, \Sigma_F)$

CLASSIFICATION

$x_6: S_m(x_6) = \log \frac{N(x_6|\mu_T, \Sigma_T)}{N(x_6|\mu_T, \Sigma_T)} \Rightarrow Scal(x_6) = \log \frac{N(S_m(x_6)|\mu_T, \sigma_T)}{N(S_m(x_6)|\mu_F, \sigma_F)}$

$= D^{Nc}(S_m|m_T, \sigma_T)$ $N(S_m|m_F, \sigma_F)$



HOW WELL THIS MODEL PERFORM DEPEND ON HOW WELL I PERFORMED THE DISTRIBUTION OF MY SCORE

LA NOSTRA TRASFORMAZIONE PRESERVA TUTTO, LA minDCF SARÀ LA STESSA DI QUELLA NON CALIBRATA CHE PORTA ALL'actDCF AD AVVICINARSI ALLA minDCF

APPLICAZIONE ALL'SVM

$x_0 \cdots x_n$ Training

$\frac{1}{2}\|w\|^2 + C\sum_i \max(0, 1 - z_i(w^T x_i + b))$

Val Set

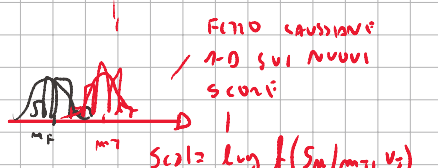
$x_0^V \cdots x_n^V$

$w^T x + b \; \cdots \; w^T x + b$

③

w, b

$S_m(x_t) = w^T x_t + b$



Scelta $\frac{\log f(S_m | m_T, v_T)}{\log f(S_m | m_F, v_F)}$

SO I HAVE TO FIND A GOOD MODEL FOR THE SCORE. AND USING A GAUSSIAN MODEL IS NOT ALWAYS THE RIGHT DECISION, I CAN USE OTHER MODEL THAT I HAVE SEEN DURING THE COURSE.

For example if I assume that the score distributions have the same covariance

$$\log \frac{f(S_m(x_T) | \mu_T, v_T)}{f(S_m(x_T) | \mu_F, v_F)} = \alpha S_m(x_t) + \beta$$

LINEAR BOUNDARY

affine mapping from non-calibrated to calibrate, usually a linear trasformation is good enough.

We can use a logistic regression using affine function

$\sigma(S_{cal}(x_T)) = \frac{P(c = H_T | x_t)}{P(c = H_F | x_t)}$ da cui x ottenere $S_{cal}(x_T) = S_{cal}(x_T) - \log \frac{\pi'_{emp}}{1 - \pi'_{emp}}$

(con cui si calibra con la)

Training set

A suo volta se non vogliamo usare la sottrazione con la calibrazione potremo applicare la formula weighted

The calibration transformation corresponds to the transformation of a prior-weighted logistic regression model

Once we have estimated α and β, the calibrated score $f(s)$ can be computed as

$$f(s) = \alpha s + \gamma = \alpha s + \beta - \log \frac{\tilde{\pi}}{1 - \tilde{\pi}}$$

Note that we have to specify a prior $\tilde{\pi}$, and we are still effectively optimizing the calibration for a specific application $\tilde{\pi}$. However, often this approach will provide good calibration for a wider range of different applications similar to the target one

We can also modify the prior-weighted logistic regression objective to compute directly α and γ:

$$R(\alpha, \gamma) = \sum_i w_i \log\left(1 + e^{-z_i(\alpha s_i + \gamma)}\right), \quad w_i = \begin{cases} \pi_T / n_T & \text{if } z_i = +1 \\ (1-\pi_T)/n_F & \text{if } z_i = -1 \end{cases}$$

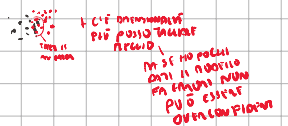
IN QUESTO CASO NUOVAMENTE π_T E NON π_{emp} TE CONSIGLIO

IN QUESTO CASO CI POTREBBERO ESSERE PROBLEMI DI CALIBRAZIONE?? SE LA CI OVERFITTA! ALLUNGHIAMO UN REGULARIZER MA IN THIS CASE PUÒ PORTARE AD UNDERFITTING, LO SCORE È POCO CALIBRATO

MA QUANDO HO EFFETTIVAMENTE OVERFITTING? CHI LO CAUSA? LA DIMENSIONALITÀ DEI DATI CHE SONO FEATURE TANTE FEATURE RENDONO DIFFICILE DI PIÙ PARAMETRI X LA STIMA, MA SE LA DIMENSIONE È PICCOLA NON RISCHIAMO OVERFITTING ⇒ NON ABBIAMO BISOGNO DI UN REGULARIZER

ESEMPIO:

ABBIAMO 2 DIMENSIONAL DATA



GS (SVM)

Training set $x_1 \dots x_n$

w, b $S_m(x) = w^T x + b$

Val Set $x_1^v \dots x_n^v$

$\Rightarrow S_m(x_1^v) \dots S_m(x_n^v)$

$\Rightarrow$

SCALE

1-D [ ] SU QUESTI APPLICHIAMO UNA PRIOR WEIGHTED = LOGISTIC REGRESSION

Training Scores vs set

$$\alpha^*, \beta^* = \arg\min_{\alpha,\beta} \frac{\pi_T}{n_T} \sum_{i \in T} \log\left(1 + e^{-(\alpha S_m + \beta)}\right) + \frac{\pi_F}{n_F} \sum_{i \in F} \log\left(1 + e^{-\alpha(\alpha S_m + \beta)}\right)$$

Prior scale

$S_m := S_m(x^t)$

α^* β^*

ALLENO IL MODELLO

$x_t \to S_m(x_t) \longrightarrow S_{post} = \alpha^* S_m(x_t) + \beta^* \to S_{cal}(x_t) = \alpha S_m(x_t) + \beta - \log \frac{\pi_T}{1 - \pi_T}$

log della prior di allenamento del modello

SE ALLENO QUANDO NON C'È BISOGNO DI TOGLIERE $\frac{\pi_T}{1-\pi_T}$

MA È LA STESSA COSA

$$R(\alpha, \gamma) = \sum_i w_i \log\left(1 + e^{-z_i(\alpha s + \gamma + \log \frac{\tilde{\pi}}{1-\tilde{\pi}})}\right)$$

OVVIAMENTE SE HO UN CASO COSÌ  CIÒ DOVE CHE POSSONO ESSERE COMPLETAMENTE SEPARATI, ALLORA HO OVERFITTING, (AVREI BISOGNO DEL REGULARIZER) MA DI SOLITO CIÒ NON ACCADE. (AVREMO BISOGNO DI PIÙ DATI)

NOTA: IL MODELLO DÀ SCORE WELL CALIBRATE È LAVORARE BENE IN BASE ALLA COMPLESSITÀ DELLA MAPPING FUNCTION.

SE HO APPLICATO DOVE LA PRIOR È 0,1 IL MODELLO LAVORERÀ BENE ANCHE PER APP 0.01, SE È TROPPO DISTANTE PERÒ DOVREI ALLENARE UN ALTRO MODELLO

CIÒ NON ACCADE X I MODELLI GENERATIVI PERÒ

SE L'ASSUNZIONE DELLA DISTRIBUZIONE È BUONA ALLORA VA BENE X DIÒ APPLICATION

Score models

- Require assumptions on the calibration transformation (e.g. affine mapping) or on the distribution of class scores
- Estimated over a training set, but allow for extrapolation outside of training score ranges
- Typically, fast to evaluate
- May provide good fit only for a small range of operating points

Example: Prior-weighted logistic regression

usiamo questo

Calibration: prior-weighted logistic regression

We consider the non-calibrated scores as 1-D feature vectors

We assume an affine mapping from non-calibrated scores to calibrated scores

$$f(s) = \alpha s + \gamma$$

Since $f(s)$ should produce well-calibrated LLRs, $f(s)$ can be interpreted as the log-likelihood ratio for the two class hypotheses

$$f(s) = \log \frac{f_{S|C}(s|\mathcal{H}_T)}{f_{S|C}(s|\mathcal{H}_F)} = \alpha s + \gamma$$

The class posterior probabilities for prior $\tilde{\pi}$ correspond to

$$\log \frac{P(C = \mathcal{H}_T|s)}{P(C = \mathcal{H}_F|s)} = \alpha s + \gamma + \log \frac{\tilde{\pi}}{1-\tilde{\pi}} = \alpha s + \beta$$

We can employ the (non-regularized) prior-weighted logistic regression model to learn the model parameters α, β from a set of training scores (we will refer to the set of samples employed to estimate the calibration parameters as calibration training set in the following)

In practice, we are treating socres as if they were 1-D feature vectors

As for model training and validation set, also the calibration set should be an independent dataset that does not overlap with neither the model training nor the validation set (we will see later how to effectively split the data)

The calibration transformation corresponds to the transformation of a prior-weighted logistic regression model

Once we have estimated α and β, the calibrated score $f(s)$ can be computed as

$$f(s) = \alpha s + \gamma = \alpha s + \beta - \log \frac{\tilde{\pi}}{1-\tilde{\pi}}$$

Note that we have to specify a prior $\tilde{\pi}$, and we are still effectively optimizing the calibration for a specific application $\tilde{\pi}$. However, often this approach will provide good calibration for a wider range of different applications similar to the target one

We can also modify the prior-weighted logistic regression objective to compute directly α and γ:

$$R(\alpha,\gamma) = \sum_i w_i \log\left(1 + e^{-z_i(\alpha s + \gamma + \log \frac{\tilde{\pi}}{1-\tilde{\pi}})}\right), \quad w_i = \begin{cases} \tilde{\pi}/n_T & \text{if } z_i = +1 \\ (1-\tilde{\pi})/n_F & \text{if } z_i = -1 \end{cases}$$

Alternatively to logistic regression, we can employ generative models to estimate calibrated LLRs

We model the class-conditional score distribution for the two classes

$$S|\mathcal{H}_T\,, \quad S|\mathcal{H}_F$$

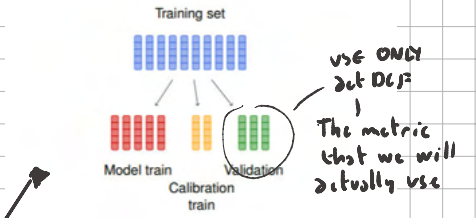
For example, we can employ a Gaussian model to estimate the two densities

$$f_{S|\mathcal{H}_T}(s) = \mathcal{N}(s|\mu_T, v_F)\,, \quad f_{S|\mathcal{H}_F}(s) = \mathcal{N}(s|\mu_T, v_F)$$

and the calibration transformation is given by the LLR for the score model

$$f_{cal}(s) = \log \frac{f_{S|\mathcal{H}_T}(s)}{f_{S|\mathcal{H}_F}(s)}$$

Tied models $v_T = v_F = v$ are usually employed, as they result in monotonic transformations



achieving low calibration error

↳ attention characteristics of the data used for calibration → calibration set should be similar to evaluation set → eval and use unseen data to avoid overfitting

Ideal → 3 set framework for calibration
- Training set → for learning
- Calibration set → used for calibrate the raw score true posterior probabilities

Validation set → for evaluation → we use act DCF vs minDCF → the score have been calibrated.

1. Miscalibration due to non-probabilistic scores, or to overfitting or underfitting models, but evaluation and training populations are similar: the calibration set can be extracted from the training set material

   In this case, calibration allows recovering a probabilistic interpretation of the scores, and, since we are training on 1-D data, the risk of overfitting is drastically reduced (and typically we don't need to add regularization to the calibration model objective function)

   we use this

2. Miscalibration due to mismatch between training and the application / evaluation population: the calibration set (and in this case the calibration validation set as well, if we want to assess the goodness of the calibration) should mimic the application population

   We need to collect data similar to the application population

   The amount of required matching data, however, is usually small compared to the amount required to train a complete classifier

   Some models (e.g. Gaussian) can be extended to exploit unlabeled samples (unsupervised or semi-supervised training), which are less expensive to acquire

How it work? → In lab → [We have saved the score compute on the validation set → subdivide the validation set into new partitions

train the classifier → then we apply it to calibration partition to generate score → feature for applay calibration technique → the applay the model to validation partition → use calibrated score metod to generate the score eval → compute actuacy

are simple → one for calibration → transformation → validation set

the other for evaluation → validation set

each dataset save one purpose in the pipeline

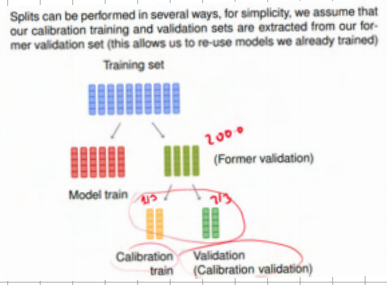
big problem

progressive reduction in dataset size available → underfitting/overfitting

Solution → Cross Validation

virtually use all the data for both training and validation

no more limitation data can serve for only one purpose



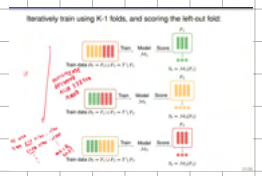
K-Folds cross validation

start with dividing the dataset into K approximately equal partitions → Folds

perfect equality is not required

1,000 samples into three folds of sizes 333, 333, and 334

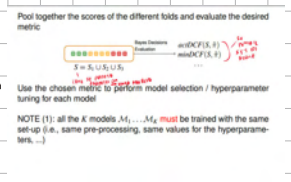
each fold maintains the statistical characteristic of the original dataset → same prior → train and evaluation cycle works with representative data that reflect over class distribution → for large dataset random sampling is sufficient



with data size increase model train on overlapping subset of data shrink coverage to similar solution

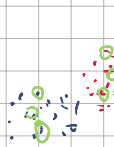
advantage → eliminate trade off → training and evaluation benefit → maximum possible data → indipendence is preserved

with can use cv for hyperparameter tuning → we use the same procedure for if svm c=1, c=10 → and based on the DCF value we can choose → We can also compare model SVM vs LR



Limitations → each training set differs by exactly one fold

each model that we trains can make different prediction → aggregated scores may not accurately represent what any single model would produce on the same evaluation data



FIRST FOLD THAT HAS OVER THAT SCORE HAS A VERY LOW DCF

BECAUSE IT IS COMPOSE OF DATA EASY TO CLASSIFY

I CAN'T USE DCF TO CHOISE

ACTUALLY I WILL NOT CHOISE ANY OF THE KFOLD

to have the best accuracy when we need to reduce the dimension of the folds

↓

we maximize the similarity between cross validation model

↓

leave one out is the best 1 sample for fold

↓

Unfortunately, leave-one-out may require training an excessively large number of models (we need to train N times a model over $N-1$ samples

We choose K as large as we can afford

- Large values of K: more robust analysis, more time required
- Small values of K: less robust analysis, but faster

validation fold where relatively small, compared to training sets

↓

additional data should have minimum impact on the model characteristics (hyperparameter computed with cross validation)

Solution: we train one more model over the whole training set, using the method we selected and the hyperparameters we estimated in the previous step

The model $\mathcal{M}$ will then be used to compute the scores for the application data

↓

for this reason we aggregate the OCR of each fold

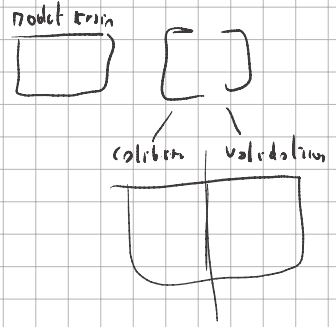
↓

and finally for the same reason

↓

how we compute the training parameter? weight

→ In laboratory we will use single-folds and k-folds on



Classifier Fusion → combining the prediction of multiple classifier

↓

can make some mistakes in some case

↓

svm in other cases

↓

This complimentary error pattern create opportunities for improving

↓

How we do it?

Majority voting

↓

what happen if we have only two classifier? → We need to consider also the confidence

C_1 H_1 C_1 $S=0,1$
C_2 H_1 C_2 $S=0,05$
C_3 H_2 C_3 $S=-3.5$

$\pi_1 = 0.5$

I should rely on the most confident

QUESTO È PIÙ CONFIDENTE ANCHE SE HO 2 CASI CHE DICONO CHE È UNA CLASSE VERA E UNA FALSA

Score based analisy

We have 3 classifier

n_1 n_2 n_3

s_1 s_2 s_3

come unisco gli score ?

$\gamma = 0$

il più alto?

$n_1 = n$

$n_2 = n$

$n_1 \to S = 10$

$n_2 \to S = 10$ — well calibrate

The problem is that we treat the classifier at the same way, but it's maybe better to weight them

↓

average is correct if the model are the same

↓

when the model are correlated strongly

IF I SUM $n_1 + n_2 = 20$ — THIS SCORE WILL NOT BE WELL CALIBRATED ANYMORE

IN THIS CASE WE SHOULD USE THE AVERAGE

LINEAR COMBINATION OF THE 3

FACCIAMO LA MEDIA $\frac{s_1 + s_2 + s_3}{3}$

$s_1 + s_2 + s_3$ — sum

it's a correct approach when classifier provide → indipendent information

the model need to be indipendent

Es n_1 produce a score looking at the first half of the features

n_2 at the second half

$\frac{f(x|H_1)}{f(x|H_0)} = \frac{f(x^1|H_1) \cdot f(x^2|H_1)}{f(x^1|H_0) f(x^2|H_1)}$ → problem if classifier are correlated we can't use fusion model

$= \log \frac{f(x^1|H_1)}{f(x^1|H_0)} + \dots \to s_{n_1} + s_{n_2}$

what we will do →

We can consider weighting the contribution of the different systems. Given scores $s_1 \dots s_m$ for a test sample x, we can compute a "fused" score as

$$s_{fused} = \alpha_1 s_1 + \alpha_2 s_2 + \alpha_3 s_3 + \dots + \alpha_m s_m + \gamma$$

We have the problem of estimating the weights α_i and the bias term γ

If we stack the scores and the weights as

$$s = \begin{bmatrix} s_1 \\ \vdots \\ s_m \end{bmatrix}, \quad \alpha = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{bmatrix}$$

the fused score can be represented as

$$s_{fused} = \alpha^T s + \gamma$$

USIAMO UN MODELLO X DARE UN PESO AD OGNI CLASSIFIER

↓

COME CALCOLIAMO γ, I PARAMETRI DI QUESTO MODELLO?

$s_1 \dots s_m$

m classifier $n_1 \dots n_m$

OTTENGO DI QUESTI → PRODUCE UNO SCORE X IL CAMPIONE X, SE CONSIDERIAMO $\begin{bmatrix} s_1 \\ \vdots \\ s_m \end{bmatrix}$ e $\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{bmatrix}$ sono i pesi

allora $s_{fused} = \alpha^T s + \gamma$

γ will be computed using logistic regression [eliminating the prior of the class]

typically we will fuse few system → 10 systems → we will not have overfitting problem (low dimensionality)

no regularization

so in other words we have

$(x_1, x_2) \dots (x_n, x_n)$

$x_1 \dots x_n$

$\begin{bmatrix} s_{11} \\ \vdots \\ s_{1n} \end{bmatrix} \dots \begin{bmatrix} s_{m1} \\ \vdots \\ s_{mn} \end{bmatrix}$

main logistic regression on this

↓ output

α, β

VALIDATION PHASE

x_0

$\begin{bmatrix} s_{01} \\ \vdots \\ s_{0m} \end{bmatrix} =$

$s_{fused} = \alpha^T s + \beta = \log \frac{p_1}{1-p_1}$

p_1

DECISION RULE TO SCORE