

Performance Evaluation

How to assess model quality in absolute numbers?

$$\hat{f} = \underset{\text{test}}{\operatorname{argmin}} \text{Remp}(f)$$

Risk after training $\sum_{(x,y) \in D_{\text{train}}} L(y, \hat{f}(x))$ which we minimize during optimization for models of hypothesis space H
can be overly optimistic, prone to overfitting when we choose model that just has the lowest training error

But "best" model should produce the most meaningful predictions on new unseen data D_{test} (while we usually don't have access to fresh data)

"out of sample testing"

true future performance

\Rightarrow Divide dataset manually into D_{train} and D_{test} . Use D_{test} to estimate Generalization Error, i.e. to calculate expected loss on new observation

Expectation over a single random test point $(x, y) \sim P_{xy}$

Generalization Error of Fixed Model: $GE(\hat{f}, L) := E_{X,Y} [L(y, \hat{f}(x))]$ assuming D_{test} drawn i.i.d. from same distribution as D_{train} ;
 $f(x) = \hat{y}^{(w)}$ independent of observations from D_{train}

Estimator for GE using dedicated test set of size m: $\hat{GE}(\hat{f}, L) = \frac{1}{m} \sum_{(x,y) \in D_{\text{test}}} L(y, \hat{f}(x))$ type of Monte Carlo Integration

Rarely possible, in the end we don't have enough data to fit the model (best result for \hat{f}), so final model can't really be used in construction of practical GE-estimator

\hookrightarrow Technically there is no data left to test! \Rightarrow instead we evaluate the next best thing, the Learner itself

unseen

Sometimes we want to use different measure to evaluate model/learner performance than what we used to fit the model (loss function, ERM)

That's why we differentiate two types of losses. Computational advantages push us to use different loss measures or simple implementation requirements.

Inner Loss: Loss function used for ERM \rightarrow model fit
(pointwise)

Outer Loss: Loss function used for performance evaluation of inner loss fitted model
(not necessarily point wise)

Inner loss must be differentiable w.r.t. parameters Θ . Outer Loss may be non-differentiable or even something abstract like ratio from confusion table

Example: Logistic Regression uses binomial inner loss and 0-1 loss outside e.g. number of misclassifications / n_{test}
when used inside for ERM it's a NP hard problem, discrete optimization

Set-Based Performance Metrics: Prediction quality measure only defined on complete test set, not for single observations

Scalar perf. metric $pe: \cup_{m \in M} (Y^m \times \mathbb{R}^{m \times g}) \rightarrow \mathbb{R}$, $(y, f) \mapsto pe(y, f)$ $m = \text{sample size of test dataset: } |D_{\text{test}}| = n_{\text{test}}$

Prediction matrix F is defined as $F := \begin{bmatrix} \hat{f}(x^{(1)}) \\ \vdots \\ \hat{f}(x^{(m)}) \end{bmatrix} \in \mathbb{R}^{m \times g}$ $\&$ refers to number of scoring functions same for linear regression
in binary classification it's actually just one, leading to scalar $f(x^{(w)})$

In multiclass case $\hat{f}(x^{(i)})$ denotes row vector with g entries $\&$ score outputs

Pointwise loss can be seen as special case $pe_L(y, f) = \frac{1}{m} \sum_{i=1}^m L(y^{(i)}, f^{(i)}) = \frac{1}{m} \sum_{i=1}^m L(y^{(i)}, \hat{f}(x^{(i)}))$
just average loss over D_{test}

Generalization Error of Inducer: Let t_{λ} be a Learner on n_{train} points from P_{xy} , then we evaluate perf. quality of models that are fitted with t_{λ} ,

Some pe might only converge for $n_{\text{test}} \rightarrow \infty$ (set-based pe)

$\Rightarrow GE(t_{\lambda}, n_{\text{train}}, pe) := \lim_{n_{\text{test}} \rightarrow \infty} E_{D_{\text{test}}} [pe(y, t_{\lambda}(D_{\text{train}}, \lambda))] = \underset{f}{\operatorname{losses}} pe_L(f) = E_{D_{\text{train}}, (x, y)} [L(y, t_{\lambda}(D_{\text{train}}, \lambda)(x))]$

yields $t \in H$, minimize f_{train}

using pointwise losses $pe_L(f)$ = no limit needed
(independently from P_{xy})

integrating out over x, y

Expected value over both D_{train} and D_{test} $\hat{=}$ overall possible randomly sampled training & test data sets, AVERAGE

Learner-GE captures randomness of full dataset in a certain way through stochasticity of learning algo and test sample $GE(t, L) = GE(t_{\lambda}, n_{\text{train}}, pe, L)$

- Problem: This assumes independent random sampling of all D_{train} and D_{test} from P_{xy} , lots of randomness associated if we use single holdout split

But interestingly: n_{train} tends to increase train error and decrease only test error (problem learning easier with less data points - we should not base our gain on this effect!)

- More data is always better, increases confid. We want $n_{\text{train}} \uparrow$ but for more certainty in Learner-GE estimate we also want larger n_{test} .

We need to hold back some data for testing, that's unavoidable. This makes GE-estimate tendentially larger than it could be because model fit during training is held back a tiny bit.

by design: Peaking-like Bias of GE, overestimates unknown true GE due to $n_{\text{train}} < n$!

Solution: Resampling techniques to properly estimate Learner GE. Reusing information from full dataset in each sampling iteration, randomness / variance cancels out ideally.

Best aggregate Performance

5) Choose Learner (config)
with minimal GE

Procedure with Resampling: 5-times resampling of dataset, so we have 5 splits / iterations

1) In each iteration find \hat{f}_i through ERM on $D_{\text{train}}^{(i)}$. These models are just intermediate results, model-type restricted by Inducer \Rightarrow use this specific learner

2) In each iteration calculate \hat{GE}_i with our chosen outer loss on \hat{f}_i predictions in $D_{\text{test}}^{(i)}$

3) Estimate $\hat{GE} = \frac{1}{5} \sum_i \hat{GE}_i$ as average GE across all resampling iterations \hookrightarrow fit final model \hat{f} (hyperparameters)

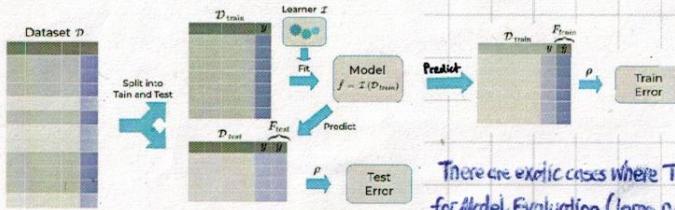
4) Repeat this for many different learners/learner configs on all data

Training- and Test-Error

Training Error and Test Error

Let $\hat{f} = I(D_{\text{train}}, \lambda)$ and let p be a performance metric. We define the

- Training Error as $p(\hat{y}_{\text{train}}, F_{\text{train}})$ = $\text{Remp}(\hat{f})$ when we use same perf. metric as inner loss used
 - Test Error as $p(\hat{y}_{\text{test}}, F_{\text{test}})$ estimator for GE of model / learner, decides model choice
- ! Choose D_{test} and D_{train} , s.t. $D_{\text{test}} \cap D_{\text{train}} = \emptyset$ to avoid optimistic Bias.



There are exotic cases where Train Error can't be used approx. for Model Evaluation (large n_{train} with weird spiral data example)

But usually it leads to bad results, choosing overfitted model of max. complexity!
⇒ "out-of-sample testing" with Test Error is correct approach

In basic form: evaluation on train data

Goodness-of-fit measures - like R^2 , χ^2 , AIC, BIC, deviance - is based on training error but are based on distributional assumptions and large enough data - therefore hard to use for high-dimensional or more complex data.

Decrease of n_{train} ⇒ increase of Test Error (in general) b.c. model generalizes better with more training data and worse with less training data.

Increase of complexity ⇒ decrease of Training Error (in general) b.c. it becomes easier to learn all patterns on small training data sets or with more features.

Decrease of n_{test} ⇒ increase of variance of test error.
↓ decrease of Var (Test error), more reliable
Test ↑ but it could lead to Test error ↑ (e.g. less random reduction effect, D_{test} now covers wider range)

GE for every iteration by trial. Train Error on super large D_{test} ⇒ average GE estimates true GE

r-simulations/repeated sampling of fixed $n_{\text{train}} = n$

assuming that $\hat{G}(f) \rightarrow \text{true } G(f)$

Hold-out sampling produces Bias-Variance-Tradeoff, that is controlled by split ratio

Bias - Variance - Tradeoff

[if D_{test} is very large $\hat{G}(f) \rightarrow \text{true } G(f)$]

The GE of the inducer can be estimated by the Test Error.

The quality of this estimate is based on the Bias(s) and the Variance of the Test Error. (s) Smaller Test Error means less Bias.

These are influenced by the complexity and amount of training data.

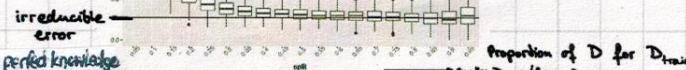
$$\text{Inference Theory: } \text{MSE}(\hat{G}(f), G(f)) = \text{Var}(\hat{G}(f)) + \text{Bias}^2(\hat{G}(f), G(f)) + \text{Bayes-Error}$$

$$\text{Bias}(\hat{G}(f), G(f)) = E(\hat{G}(f)) - G(f)$$

↓ Captures everything unknown that we can't include in our modeling

Bayes Error] irreducible error due to inherent randomness/noise... Occurs even with perfect knowledge

Lowest possible prediction error of optimal learner given a certain perf. measure/loss function



Rule of thumb for hold-out split: But in practice we should not use hold-outs, esp. for small n , and instead apply resampling!

$\frac{2}{3}$ of D as D_{train} and $\frac{1}{3}$ of D as D_{test}

Model influenced by: Learner Algorithm, chosen Loss functions, data, randomness

OF and UF are (theoretical) properties of ERM-optimal fixed models \hat{f} that we should never apply

Overfitting and Underfitting

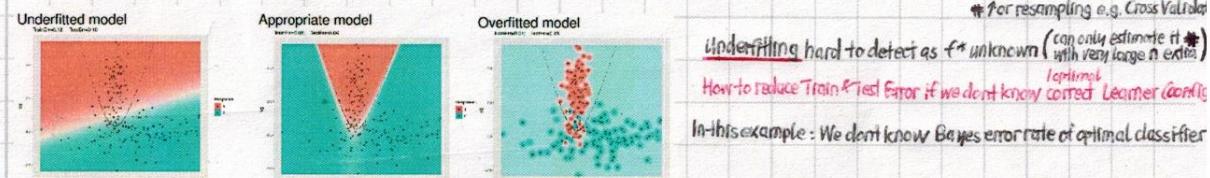
There is an overlap between UF and OF, no clear cutoff (terrible models do both)

Underfitting occurs when a model can't reflect the true shape of underlying function (given the data)

↳ High Test Error and high Training Error. $UF(\hat{f}, L) := GE(\hat{f}, L) - GE(f^*, L)$ f^* is unknown Bayes optimal model

Overfitting occurs when a model reflects noise or artifacts from D_{train} which do not generalize.

↳ Small Train Error and high Test Error. $OF(\hat{f}, L) = GE(\hat{f}, L) - \text{Remp}(\hat{f}, L)$ in practice $OF > 0$ if f is too big but problem starts when



Underfitting hard to detect as f^* unknown (can only estimate it #)

Optimal How to reduce Train & Test Error if we don't know correct Learner (cont'd)

In this example: We don't know Bayes error rate of optimal classifier

technically also underfitted [in these cases OF more dominant problem, has "prior"]

Overfitting is influenced by

- Complexity of Model, i.e. $\dim(\theta)$
- Amount of Training Data, i.e. n_{train}
- Dimensionality of Feature Space, i.e. $\dim(x)$
- Irreducible Noise, i.e. aleatoric uncertainty

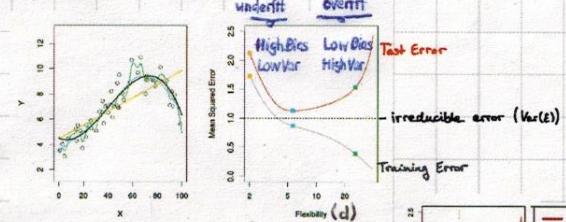
The less data we have and the more features we want to include, the more we should lean towards low complexity models, and vice versa. More data ⇒ less randomness that complex model can overfit on.

although its possible that ERM-result is simple LM and higher-degree off. all = 0

Polynomial Learner] degree 11 will lead to interpolation-like overfit

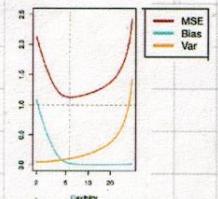
This explains one big problem of interpolators like Interpol. splines or Interpol. Gaussian processes whose predictions always perfectly match regression target, but interpolate random noise as well.

Influence of complexity:



We can deconstruct the MSE of the Test Error into its Variance and Bias:

We can see that we need to trade-off in order to minimize the MSE.



Resampling



Setup

We want to estimate $\text{GE}(I, \lambda, n, p_i) = \text{GE}(\hat{f}, L)$
but doing so with a single hold-out-split results in a high pessimistic Bias & high Variance of $\hat{\text{GE}}$

- So instead we split repeatedly and average the result.
- This way we reduce the variance from small test sets.
- also reduce pessimistic bias of $\hat{\text{GE}}$ with larger split ratio $\frac{1}{n_{\text{train}}}$ which resampling indirectly enables us to do by $(n_{\text{train}}, 1)$ works out because of resampling process favorizing

Resampling Strategies

We choose $J_{\text{train}} \in \{1, \dots, n\}^{n_{\text{train}}}$ and $J_{\text{test}} \in \{1, \dots, n\}^{n_{\text{test}}}$.
"Class i indices from $1, \dots, n$ with replacement"

We define $J := (J_{\text{train},1}, J_{\text{test},1}, \dots, J_{\text{train},B}, J_{\text{test},B})$

with $B \in \mathbb{N}$ and $n_{\text{train},1} \approx n_{\text{train},2} \approx \dots \approx n_{\text{train},B} \approx n_{\text{train}}$

Our estimate is then aggregation overall, usually mean:

$$\hat{\text{GE}}(I, \lambda, J, p) = \frac{1}{B} \sum_{k=1}^B p(y_{J_{\text{test},k}}, F_{J_{\text{train},k}, I(D_{\text{train},k})})$$

Typically, we re-use exact splits for Learner(config.) comparisons, comparable

Cross Validation

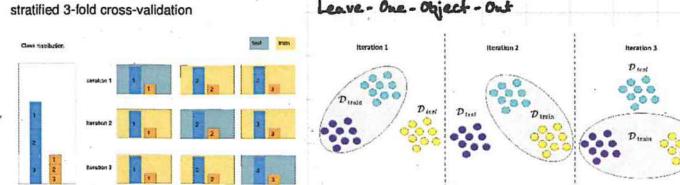
1. Split the data into k equally-sized partitions. Fraction $\frac{k-1}{k}$ for train, $\frac{1}{k}$ for test
(May also want to do this for imbalanced feature classes)
2. Each set is test set once, the rest is used for training \rightarrow Results in k test errors (and training errors)
3. Average the test errors Each datapoint is used exactly once for testing $\Rightarrow "n_{\text{test}} = n"$ in an abstract way using all info for testing
rhave-to-fit n models (computing time...) LOO-CV When n is small go for large k , but avoid extreme $k=1$
! 5 or 10 fold CV's are common. n -fold CV is also called Leave-one-out-CV \rightarrow Nearly unbiased but high variance.
! The test errors are not independent b.c. the splits aren't independent \rightarrow There is no unbiased estimate for $V[\text{GE}(E, \cdot)]$.
Esp for very small n we should consider integrating Monte Carlo approach into CV \Rightarrow Repeat CV with large k and average over it to reduce $\text{Var}(\hat{\text{GE}})$

Leave-one-Object-Out

This method is used when there are multiple observations per object \rightarrow Data is not i.i.d. anymore.
The data of one object should either be in the training set or test set ? Not in both \rightarrow Otherwise GE is biased.
This can be achieved by using CV on objects, i.e. split the objects into k groups...

Example CV

3-fold CV



Bias-Variance Analysis in Resampling

If there exists a dedicated Test set that is not used to train f , then $\hat{\text{GE}}(\hat{f}, L) = \frac{1}{m} \sum_{i=1}^m L(y_i, \hat{f}(x_i))$
usually in samples not independent there is no dedicated test set \rightarrow $\hat{\text{GE}}(\hat{f})$ biased!
with the m samples being i.i.d. and $E[\text{GE}(\hat{f}, L)] = E[L(y, \hat{f}(x))] = \text{GE}(\hat{f}, L)$ and $V[\text{GE}(\hat{f}, L)] = \frac{1}{m} V[L(y, \hat{f}(x))]$ linear $\text{Var}(\text{GE})$
decrease in test error.

We can apply the Central Limit Theorem to approximate the dist. of $\text{GE}(\hat{f}, L)$ and compute Confidence Intervals and hypothesis tests

If \hat{f} is trained on D we estimate $\text{GE}(I, \lambda, n, p)$ instead of $\text{GE}(\hat{f}, L)$ using a resampling based estimate $\hat{\text{GE}}(I, \lambda, J, p)$
 $E[\hat{\text{GE}}(I, \lambda, J, p)] = E[p(y_{J_{\text{test}}, F_{J_{\text{train}}, I(D_{\text{train}})}})] = E\left[\frac{1}{m} \sum_{i=1}^m L(y_i, I(D_{\text{train}}))(x_i)\right] = \text{GE}(I, \lambda, n_{\text{train}}, p_i) > \text{GE}(I, \lambda, n, p)$

! So our estimate for $\text{GE}(I, \lambda, n, p)$ is in expectation nearly correct, i.e. $n_{\text{train}} < n$ in our estimate
is pessimistically biased - on n it would perform better

Holdout Subsampling technically, also overlap (not random, not systematic) CV and Bootstrap: No scalar $\text{Var}(\text{GE}(\cdot))$, instead are correlated \rightarrow matrix (for CV)

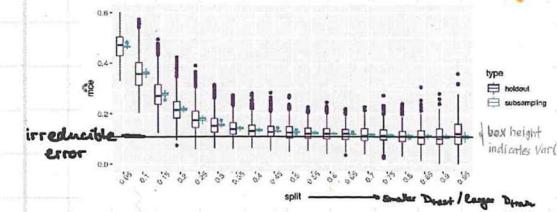
Subsampling/Monte Carlo CV

1. Split the data into two partitions
2. Use one set for training and the other for testing. \rightarrow Results in one test error
3. Repeat step 1 & 2. many times
4. Average over all test errors
- ! $\frac{3}{10}$ or $\frac{9}{10}$ of the data for training is common.

Bootstrap

1. Draw a training set of size n_{train} with replacement from the full data set D . The data of D that is not unique test points, $D_{\text{test}} = D \setminus D_{\text{train}}$. Replicated train points can lead to problems and artifacts during fitting!
 2. Train and evaluate the model \rightarrow Results in a test error
 3. Repeat 1 & 2. many times
 4. Average over all test errors
- ! A training set contains about $3g$ unique points (for $n \rightarrow \infty$)
 $\rightarrow P((X_{1:n}) \in D_{\text{train}}) = 1 - (1 - 1/n)^n \xrightarrow{n \rightarrow \infty} 1 - 1/e \approx 63.2\%$

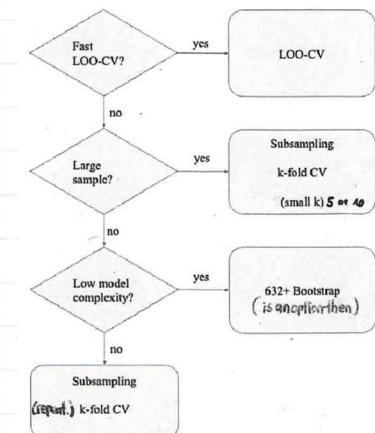
Bias-Variance-Trade-off in Subsampling



! Bias(Subsampling | split-rate) = Bias(Hold-out | split-rate)
Variance (Subsampling | split-rate) < Variance (Hold-out | split-rate)
 \Rightarrow "optimal" split-rate is higher in Subsampling compared to one Hold-out-split.



Guidelines



Also watch out for class effects and consider stratification (esp. with simple holdout split)
Even large datasets with $n \geq 100,000$ can have "hidden" small sample sizes, e.g. one group very small.

$$\hat{\text{GE}}(\hat{f}, L) = \frac{1}{m} \sum_{i=1}^m L_i, l_i \text{ thresample}$$

$$\text{estimate } \frac{1}{m} \sum_{i=1}^m (\hat{l}_i - l_i)^2, l_i \text{ i.i.d.}$$

Performance Measures for Linear Regression

Pointwise outer Loss

$$\text{Mean squared error (MSE)}: f_{\text{MSE}}(\hat{\eta}, \tilde{\eta}) = \frac{1}{m} \sum_{i=1}^m (\eta^{(i)} - \hat{\eta}^{(i)})^2 \in [0, \infty)$$

Sum of squared errors (SSE): $f_{\text{SSE}} = m \cdot f_{\text{MSE}}$ very similar

- MSE generalizes L2 loss to performance metric. Then we use equivalent inner & outer losses (without gradient) $\rightarrow f_{\text{RMSE}}(\hat{\eta}, \tilde{\eta}) = \sqrt{f_{\text{MSE}}} \rightarrow$ back to original scale
- Outliers with large prediction error heavily increase MSE as they enter quadratically

$$\text{Mean absolute error (MAE)}: f_{\text{MAE}} = \frac{1}{m} \sum_{i=1}^m |\eta^{(i)} - \hat{\eta}^{(i)}| \in [0, \infty)$$

Even more robust against outliers is Median Absolute Error $f_{\text{MedAE}} = \text{Med}(|\eta^{(i)} - \hat{\eta}^{(i)}|)$

$$\text{Mean absolute percentage error (MAPE)}: f_{\text{MAPE}} = \frac{1}{m} \sum_{i=1}^m \left| \frac{\eta^{(i)} - \hat{\eta}^{(i)}}{\eta^{(i)}} \right| \in [0, \infty)$$

- Small $|\eta^{(i)}|$ have more influence; can handle $\eta^{(i)} = 0$

$$\text{Mean absolute scaled error (MASE)}: f_{\text{MASE}} = \frac{f_{\text{MAE}}}{\text{"Naive MAE"} \text{ (with } \text{MAE}_{\text{naive}} = \frac{1}{m-1} \sum_{i=1}^{m-1} |\eta^{(i)} - \eta^{(i+1)}|)}$$

- Scale independent error ratio, used for time series forecasts

- aim for MASE < 1, forecast more accurate than naive forecast (on average)

Set-based outer Loss

$$R^2: f_{R^2}(\hat{\eta}, \tilde{\eta}) = 1 - \frac{\sum_{i=1}^m (\eta^{(i)} - \hat{\eta}^{(i)})^2}{\sum_{i=1}^m (\eta^{(i)} - \bar{\eta})^2} = 1 - \frac{\text{SSE}}{\text{SST}} \quad \text{"Fraction of } \eta \text{-variance explained" by the model}$$

$$= 1 - \frac{f_{\text{MSE}}(\hat{\eta}, \text{LM}(\hat{\eta}))}{f_{\text{MSE}}(\hat{\eta}, \bar{\eta})} = 1 - \frac{\text{SSE LM}}{\text{SSE const. Model}} \quad \text{More specifically the influence of features on target}$$

$$\frac{\text{SSM}}{\text{SST}} = \frac{\sum (\hat{\eta}^{(i)} - \bar{\eta})^2}{\sum (\eta^{(i)} - \bar{\eta})^2}$$

SSE reduction of complex model vs. constant model baseline, generalized to ML with outer Loss: $f_{R^2} = 1 - \frac{\text{Loss}_{\text{complex Model}}}{\text{Loss}_{\text{simple Model}}}$

for model comparisons beyond LM even different model types

In Linear Modeling R^2 is typically only used for evaluating on train data where $R^2 \in [0, 1]$ always, can't be worse than const. $\bar{\eta}$ -prediction

$R^2 = 1$: perfect prediction $R^2 = 0.9$: LM reduces SSE / Loss by factor 10 $R^2 = 0$: predict as badly as const. model

On test set $R^2 < 0$ is possible because LM not fitted to new data, Quad. summenzerlegung ungültig $\Rightarrow \text{SSE LM}$ can be > SSE const. Model

strong indicator for terrible underfitted model or highly overfitted model that generalizes horrendously to new unseen data!

* Higher R^2 does not always equate better fit, not even on training data

R^2 can simply be increased by using more data (wider range) while fit measured with MSE stays the same or even worse (less data \rightarrow easier to fit patterns)

R^2 increases with more features used and higher complexity of model, but only on train data. For new data this makes overfitting more likely

✓ R^2 invariant w.r.t. linear scaling of η , which is not true for MSE

$$\text{Spearman Correlation (rank-based)}: f_{\text{Spearman}}(\eta, \tilde{\eta}) = \frac{\text{Cov}(\text{rg}(\eta), \text{rg}(\tilde{\eta}))}{\sqrt{\text{Var}(\text{rg}(\eta)) \text{Var}(\text{rg}(\tilde{\eta}))}} \in [-1, 1]$$

- Very robust against outliers, don't care about actual values just ranks

- Invariant under monotone transformations of η

If $f_{\text{Spearman}} = 1$, so $f_{\text{sp}} \in [-1, 1]$ means perfect monotonic relationship between η and predictions $\tilde{\eta}$

Measures for Classification

Pointwise outer Loss using class Labels

Missclassification Error Rate (MCE): $p_{MCE} = \frac{1}{m} \sum_{i=1}^m \mathbb{I}_{[\hat{y}^{(i)} \neq y^{(i)}]} = \frac{\text{# Mislabeled}}{\text{Total observations}} \in [0, 1]$ could also call this a 0-1 loss function $L_{0-1}(y, \hat{y}) = \frac{1}{m} \sum_{i=1}^m \mathbb{I}_{[y^{(i)} \neq \hat{y}^{(i)}]}$

Accuracy (ACC): $p_{ACC} = \frac{1}{m} \sum_{i=1}^m \mathbb{I}_{[\hat{y}^{(i)} = y^{(i)}]} = \frac{\text{# True Pos. + True Neg.}}{\text{Total Observations}} \in [0, 1]$ "Proportion of correctly classified observations" $p_{ACC} = 1 - p_{MCE}$

Problems | MCE says nothing about how good/skewed the predicted probabilities are | Accuracy is bad performance measure when Label dist. is unbalanced,

for example if only tiny fraction of 0.05% have certain disease

Errors on all classes are weighted equally which is often inappropriate!

→ Always predicting "No disease" (1-0) has ACC = 0.9995 and MCE = 0.0005

Example disease prediction: False Negative might be much worse than False Positive!

Terrible system that sends all sick patients home, "Accuracy Paradox"

"Imbalanced Cost Problem"

Pointwise outer Loss using probabilities

Brier Score (BS): $p_{BS} = \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2$ MSE for probabilities encoded as 0/1 in binary case

Multiclass Brier: $p_{BS, MC} = \frac{1}{m} \sum_{i=1}^m \sum_{k=1}^K (\hat{y}_k^{(i)} - y_k^{(i)})^2$

[$y_k^{(i)}$] One-hot-encoded class label

Log/Bernoulli-Loss (LL): $p_{LL} = \frac{1}{m} \sum_{i=1}^m (-y^{(i)} \log(\hat{y}^{(i)}) - (1-y^{(i)}) \log(1-\hat{y}^{(i)}))$

↑ Logistic Regression

Multiclass: $p_{LL, MC} = \frac{1}{m} \sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log(\hat{y}_k^{(i)})$

Both Brier-Score and Log Loss: Optimal value is 0, "confidently wrong" predictions are penalized (with Log Loss much more heavily at the edges)

Brier Score for individual obs. $\in [0, 1]$ so capped at 1 | Log-Loss $\in [0, \infty)$ => tries really hard to avoid False Pos. & False Neg. Predictions

with high $\hat{y}^{(i)}$ with low $\hat{y}^{(i)}$

Confusion Matrix

		True classes			n _{pred}
Predicted		A	B	C	
Classes	A	50	0	0 (0)	50
	B	0	46	4 (4)	50
	C	0	4	46 (4)	50
error	(0)	(4)	(4)	[8]	-
n _{true}	50	50	50	-	150

cell value = number of class k predictions for given True class

		True Class y		# Pos. Pred.
Binom:		y pos.	y neg.	
Pred. Class k	# pos.	True Positive (TP)	False Positive (FP)	
	# neg.	False Negative (FN)	True Negative (TN)	# Neg. Pred.
# Pos. Class		# Neg. Class		0

Cost Matrix: Basically another Matrix with same dimensions on top of Confusion Matrix

Usually only #FN, #FP relevant for cost calculation

(Usually true predictions get 0 entry in cost matrix)

Assign different costs to different errors of Confusion Matrix | Costs = $\frac{1}{n} \sum_{i=1}^n C([y^{(i)}, \hat{y}^{(i)}])$ | Then costs = $\frac{1}{n} \text{tr}((\text{Conf. mat})^T \cdot (\text{cost. mat}))$

(for certain classes)

This allows us to put different weights on False Positive/Negative or focus on one error type entirely and setting all other cost matrix entries to zero!

However: in practice how to specify costs precisely? We could evaluate from different perspectives, with multiple metrics, to get an idea of system quality.

An important subfield of ML is cost-sensitive learning

Set based outer Loss / ROC-analysis

Many different metrics based on Confusion Matrix. Metric Focus requires domain knowledge!