Machine Learning

Lecture 5: Evaluation

Prof. Dr. Aleksandar Bojchevski

24.04.24

Clever Hans

Outline

Recipe

Benchmarks and datasets

Statistical testing

Metrics

A model selection recipe

1. Train a few models $f_1, f_2, \ldots$
2. Estimate the performance (error) of each model
3. Choose the one with highest performance (lowest error)

Some important questions are:

- Which performance **metric** should we use?
- On which **data** do we compute the metric?
- How do we account for **random** effects?

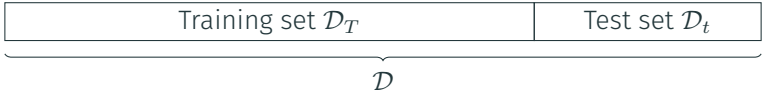
Generalization

Goal is **generalization**: find a model that performs best on unseen (future) data.

Never evaluate on the training set because:

- It is not an unbiased estimator of the population risk.
- Leads us to choose models that overfit (to the noise in the data).
- Often not useful since the training loss can be zero for many models, e.g. a decision tree with completely pure nodes.

One (often flawed) idea is to split the data into a *training* set and *test* set.
Use the training set to learn the model, and the test set for evaluation.



What about hyperparameters?

Example test set performance for different k in k-NN on synthetic data:

k	1	2	**3**	4	5	6
Accuracy	92%	87%	**93%**	91%	89%	76%

What about hyperparameters?

Example test set performance for different k in k-NN on synthetic data:

k	1	2	**3**	4	5	6
Accuracy	92%	87%	**93%**	91%	89%	76%

Sample a fresh test set but keep the same model (same training set):

k	**1**	2	3	4	5	6
Accuracy	**94%**	85%	91%	89%	90%	79%

Now a different value of k is “optimal”. What is happening?

What about hyperparameters?

Example test set performance for different k in k-NN on synthetic data:

k	1	2	**3**	4	5	6
Accuracy	92%	87%	**93%**	91%	89%	76%

Sample a fresh test set but keep the same model (same training set):

k	**1**	2	3	4	5	6
Accuracy	**94%**	85%	91%	89%	90%	79%

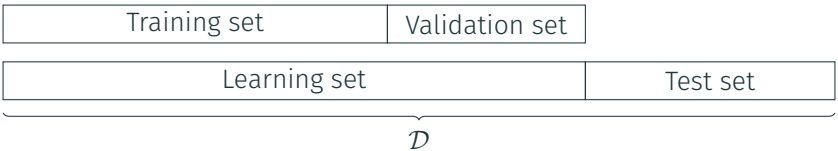
Now a different value of k is "optimal". What is happening?

We are overfitting again! This is an instance of the multiple testing problem.

Never reuse your test data!

A better recipe

To choose a model (e.g. k-NN vs. Trees) and/or hyperparameters (e.g. values of k):



During model development (loop for different choices):

1. Train on training set
2. Evaluate on validation set

Final round at the end:

1. Train on training set (sometimes you can also add the validation set)
2. Evaluate **only once** on the test set

This does not prevent multiple testing, just provides a final failsafe to detect it.

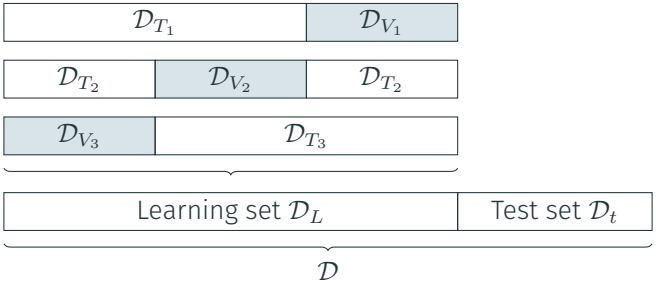
Cross validation

Split your learning set into K folds (5-fold and 10-fold CV are common).

Use $K-1$ folds for training and the remaining fold for evaluation.

Average the performance metric over all folds to get an estimate. Pick one model.

Use the entire learning set for training. We still evaluate **only once** on the test set.



The extreme case - LOOCV

In *leave-one-out-cross validation* (LOOCV) we train on all but one sample.

If we have N samples, this is the same as N-fold cross-validation.

LOOCV is interesting if we do not have a lot of data and we want to use as much of it for training as possible but still get a good estimate of model performance.

But it also means that we need to train our model N times.

For sufficiently large data and/or computationally expensive training stick to lower numbers of K or a single validation set.

There is also nested cross validation. In the outer loop we split $\mathcal{D}$ into multiple learning/test sets. In the inner loop we split each learning set into training/validation sets.

Early stopping

Often even for a fixed set of hyperparameters learning is an iterative procedure:

1. Initialize the model $f^{(0)}$: e.g. empty tree, random parameters $\boldsymbol{\theta}$
2. Repeat for T epochs
   2.1 Update from $f^{(t-1)}$ to $f^{(t)}$: e.g. add a new node, gradient descent step
3. Return $f^{(T)}$

Early stopping

Often even for a fixed set of hyperparameters learning is an iterative procedure:

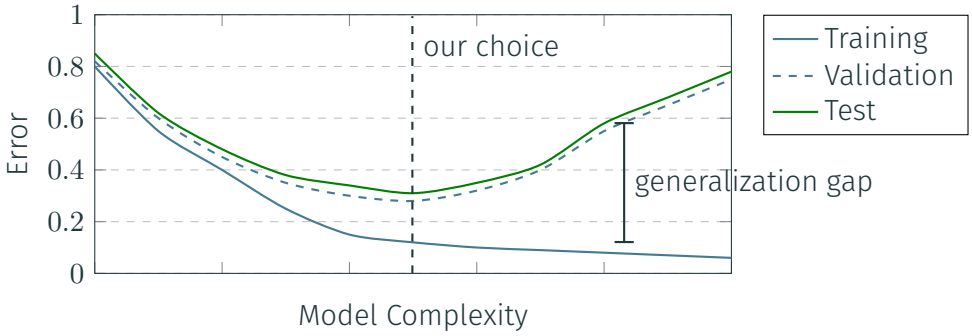
1. Initialize the model $f^{(0)}$: e.g. empty tree, random parameters $\boldsymbol{\theta}$
2. Repeat for T epochs
   2.1 Update from $f^{(t-1)}$ to $f^{(t)}$: e.g. add a new node, gradient descent step
   2.2 If $\mathcal{L}(f^{(t)}, \mathcal{D}_{\text{val}}) > \mathcal{L}(f^{(t-1)}, \mathcal{D}_{\text{val}})$ return $f^{(t-1)}$
3. Return $f^{(T)}$

In 2.2. we stop early if the loss on the validation set increases.

In practice we often also use a *patience* parameter P: if no improvement in P epochs, return $f^{(t)}$ for which $\mathcal{L}(f^{(t)}, \mathcal{D}_{\text{val}})$ was lowest (special case $P = 1$ above).

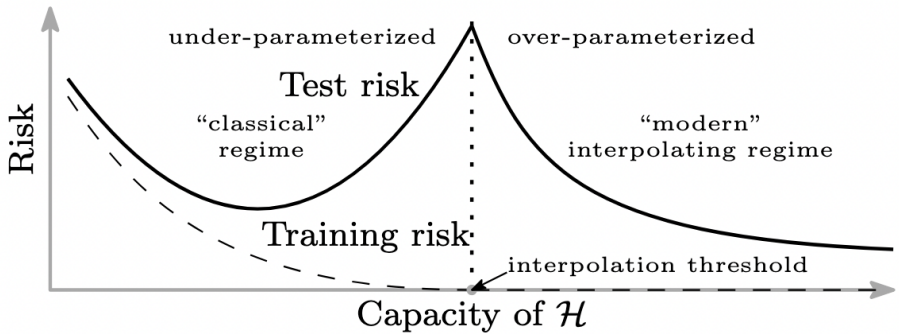
Example error curves

By increasing model complexity we can always reduce the training error, but the validation/test error starts to increase.



Double descent

Modern over-parametrized classifiers (e.g. massive neural networks) exhibit a double descent behavior, where the test risk keeps decreasing after a threshold.



Which hyperparameters to try?

Trial-and-error (intuition): probably the most common approach.

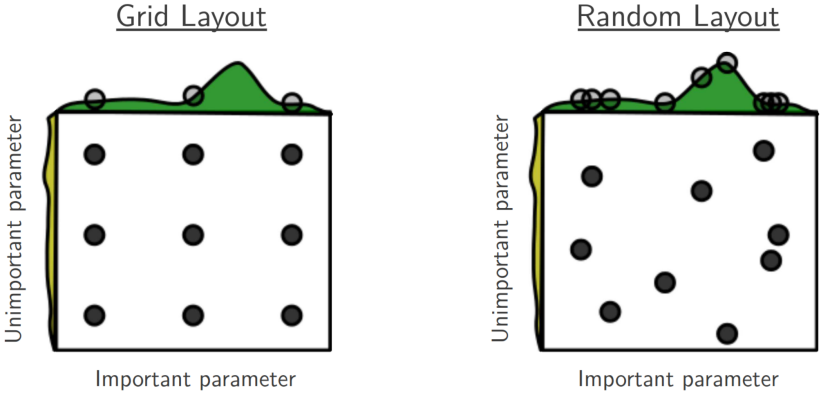
Grid search: define a finite set of values per hyperparameter, try all combinations.

Random search: randomly select hyperparameter configurations.

Bayesian optimization: probabilistic model of impactful hyperparameters.

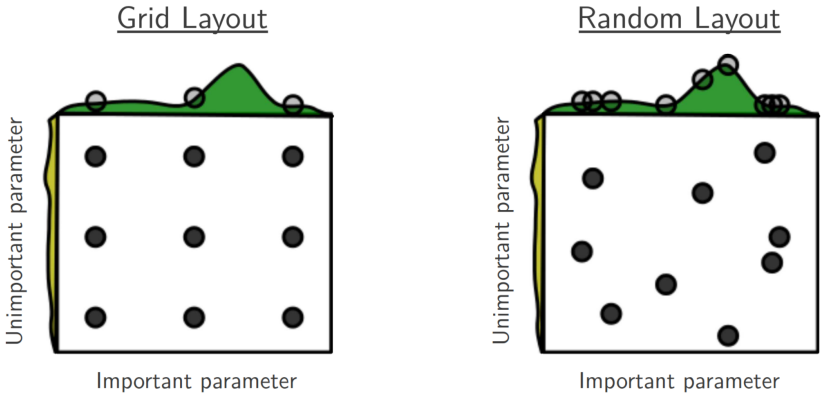
Telescopic search: first, find the best order of magnitude, then do a fine-grained search. For example, first try $k \in \{0.01, 0.1, 1, 10, 100, 1000\}$. If $k = 10$ is best try $k = \{5, \ldots, 95\}$.

Grid search vs. Random search



Source: http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

Grid search vs. Random search



With random 9 different values per parameter (compared to 3) at the same cost.

Source: `http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf`

What's wrong with the following procedure:

1. Compute $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ on $\mathcal{D}$
2. Standardize the data: $\boldsymbol{x}_i' = \frac{\boldsymbol{x}_i - \boldsymbol{\mu}}{\boldsymbol{\sigma}}$
3. Split $\mathcal{D}$ into $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}}$
4. Train on $\mathcal{D}_{\text{train}}$, select model with $\mathcal{D}_{\text{val}}$, evaluate on $\mathcal{D}_{\text{test}}$

Leakage

What's wrong with the following procedure:

1. Compute $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ on $\mathcal{D}$
2. Standardize the data: $\boldsymbol{x}_i' = \frac{\boldsymbol{x}_i - \boldsymbol{\mu}}{\boldsymbol{\sigma}}$
3. Split $\mathcal{D}$ into $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}}$
4. Train on $\mathcal{D}_{\text{train}}$, select model with $\mathcal{D}_{\text{val}}$, evaluate on $\mathcal{D}_{\text{test}}$

By computing $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ on the entire $\mathcal{D}$ information from the test set **leaked**.

We have a type of leakage known as premature featurization.

Leakage

What's wrong with the following procedure:

1. Compute $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ on $\mathcal{D}$
2. Standardize the data: $\boldsymbol{x}_i' = \frac{\boldsymbol{x}_i - \boldsymbol{\mu}}{\boldsymbol{\sigma}}$
3. Split $\mathcal{D}$ into $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}}$
4. Train on $\mathcal{D}_{\text{train}}$, select model with $\mathcal{D}_{\text{val}}$, evaluate on $\mathcal{D}_{\text{test}}$

By computing $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ on the entire $\mathcal{D}$ information from the test set **leaked**.

We have a type of leakage known as premature featurization.

To prevent data leakage: preprocessing must be done within each split.

In the example above compute $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ on just $\mathcal{D}_{\text{train}}$ and then apply to $\boldsymbol{x} \in \mathcal{D}$.

Other types of leakage

Feature leakage caused by one of the following: a duplicate label, a proxy for the label, or the label itself. In general: don't use features not available at test time.

- e.g.: using "minutes late" feature when predicting "is late".

Duplicate instances in train/validation/test, e.g. bootstrap before splitting.

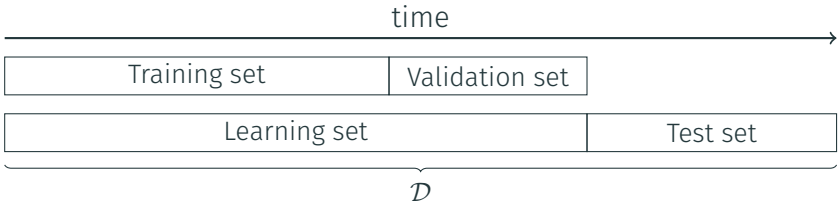
Group leakage – random instead of group-wise split, e.g. multiple x-rays of the same patient (some in train and some in test set).

Time leakage – random split instead of newer data in the test split.

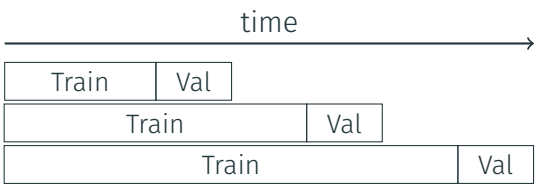
If the results seem too good to be true you probably have leakage or a bug!

Temporal data

Split according to time, newer (future) data in the test split:



Alternative, walk-forward or rolling cross validation:



How to set up your experiments?

Do not just blindly apply cross validation (or 3-way split).

Test set evaluation is a simulation of production.

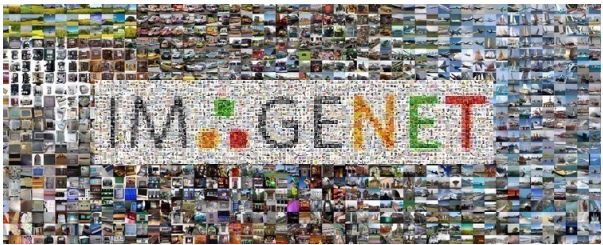
Validation set evaluation is a simulation of test set evaluation.

Outline

The benefits of benchmarks

Some argue that benchmarks (datasets) are one of the key drivers of progress.

Example: ImageNet and the large scale visual recognition challenge (ILSVRC).



Many others: Open Graph Benchmark, SQuAD, (Super)GLUE, RLBench, ...

The downsides of benchmarks

Datasets used for benchmarks often have a single fixed split $\rightarrow$ a fixed test set.

Often the test set is public, but sometimes (e.g. competitions) it is keep hidden (e.g. until the end of the competition).

Thousands of papers on the same benchmark data (e.g. CIFAR-10, ImageNet).

The downsides of benchmarks

Datasets used for benchmarks often have a single fixed split → a fixed test set.

Often the test set is public, but sometimes (e.g. competitions) it is keep hidden (e.g. until the end of the competition).

Thousands of papers on the same benchmark data (e.g. CIFAR-10, ImageNet).

Does this mean we are overfitting to the *datasets*?

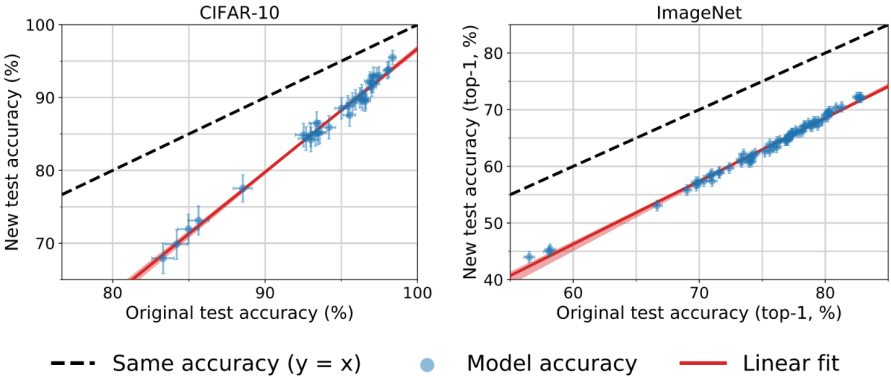
Yes! Information from the test set (indirectly) leaks.

New papers can adapt their methods based on findings from previous papers.

How can we know if we have made any progress?

Replication

Researchers created **new test sets** for the CIFAR-10 and ImageNet classification benchmarks, by carefully following the original procedure for creating the dataset.



All models suffer a significant drop, yet high correlation between old and new set.

The downsides of benchmarks

Harms associated with data:

- Representational harm and biases.
- Privacy violations.
- Problem framing and comparisons with humans.
- ...

We will discuss these harms in more detail in the last two weeks of the lecture.

Outline

Recipe

Benchmarks and datasets

Statistical testing

Metrics

Statistical testing

Unobservable	Observable
true data distribution p	dataset $\mathcal{D} \sim p$
true loss $\mathcal{L}(f) = \mathbb{E}_{(\boldsymbol{x},y)\sim p}[\ell(y, f(\boldsymbol{x}))]$	empirical loss $\mathcal{L}(f, \mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} \ell(y_i, f(\boldsymbol{x}_i))$
true confidence interval	empirical confidence interval

We cannot compare $\mathcal{L}(f)$ vs. $\mathcal{L}(g)$, but only $\mathcal{L}(f, \mathcal{D}_{\text{test}})$ vs. $\mathcal{L}(g, \mathcal{D}_{\text{test}})$.

Are the difference between models $(f, g, h, \dots)$ real or due to **chance**?

Statistical testing

Unobservable	Observable
true data distribution p	dataset $\mathcal{D} \sim p$
true loss $\mathcal{L}(f) = \mathbb{E}_{(\boldsymbol{x},y)\sim p}[\ell(y, f(\boldsymbol{x}))]$	empirical loss $\mathcal{L}(f, \mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} \ell(y_i, f(\boldsymbol{x}_i))$
true confidence interval	empirical confidence interval

We cannot compare $\mathcal{L}(f)$ vs. $\mathcal{L}(g)$, but only $\mathcal{L}(f, \mathcal{D}_{\text{test}})$ vs. $\mathcal{L}(g, \mathcal{D}_{\text{test}})$.

Are the difference between models $(f, g, h, \dots)$ real or due to **chance**?

Confidence intervals

A $100(1-\alpha)\%$ confidence interval for a parameter estimate θ is any interval $I(\mathcal{D}) = (l(\mathcal{D}), u(\mathcal{D}))$ derived from a dataset $\mathcal{D}$ such that:

$$p(\theta \in I(\mathcal{D}) \mid \mathcal{D} \sim \theta) = 1 - \alpha$$

Usually we set $\alpha = 0.05$ to get a 95% confidence interval (CI).

Interpretation: If we repeatedly sample data $\mathcal{D}$, and compute $I(\mathcal{D})$ for each dataset, then 95% of such intervals will contain the true parameter θ.[1]

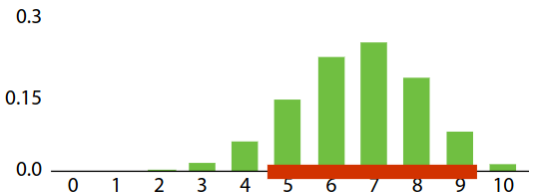
Does **not** mean that for any particular dataset that $\theta \in I(\mathcal{D})$ with 95% probability! This is what a Bayesian credible interval computes.

[1] The true parameter is fixed and the data is random.

Confidence interval for the accuracy

Let θ be the parameter of the Bernoulli distribution.

If we perform N trials the number of success is distributed as $\mathrm{Binom}(N, \theta)$.



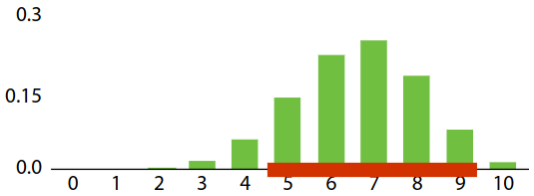
Clopper–Pearson interval: exact $1 - \alpha$ coverage.

Normal approximation: $\theta_{\mathrm{MLE}} \pm z\sqrt{\frac{\theta_{\mathrm{MLE}}(1-\theta_{\mathrm{MLE}})}{N}}$ where z is the $1 - \alpha/2$ quantile of the standard normal distribution and θ_{MLE} is the sample mean.

Confidence interval for the accuracy

Let θ be the parameter of the Bernoulli distribution.

If we perform N trials the number of success is distributed as $\mathrm{Binom}(N, \theta)$.



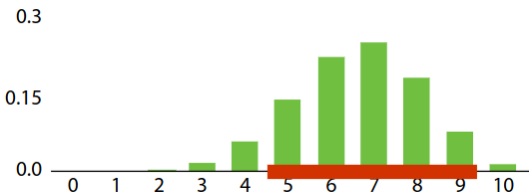
Clopper–Pearson interval: exact $1 - \alpha$ coverage.

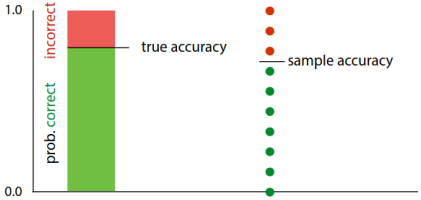
Normal approximation: $\theta_{\mathrm{MLE}} \pm z\sqrt{\frac{\theta_{\mathrm{MLE}}(1-\theta_{\mathrm{MLE}})}{N}}$ where z is the $1 - \alpha/2$ quantile of the standard normal distribution and θ_{MLE} is the sample mean.

Confidence interval for the accuracy

Let θ be the parameter of the Bernoulli distribution.

If we perform N trials the number of success is distributed as $\mathrm{Binom}(N, \theta)$.



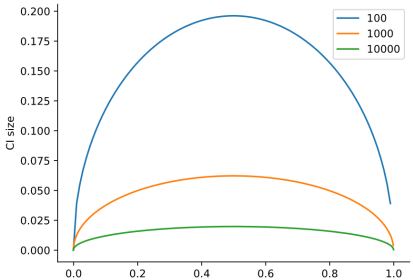


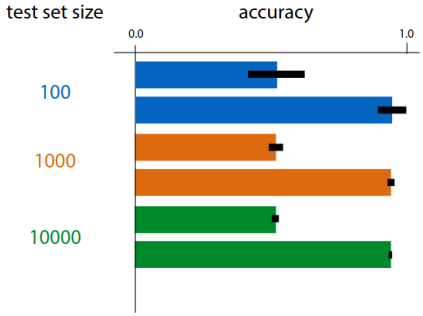
Clopper–Pearson interval: exact $1 - \alpha$ coverage.

Normal approximation: $\theta_{\mathrm{MLE}} \pm z\sqrt{\frac{\theta_{\mathrm{MLE}}(1-\theta_{\mathrm{MLE}})}{N}}$ where z is the $1 - \alpha/2$ quantile of the standard normal distribution and θ_{MLE} is the sample mean.

Confidence interval for accuracy

The size of the CI is a function of: the true accuracy and the test set size.





Avoid small test sets. Alpaydin's combined 5x2 F test[2] is an alternative.

[2] https://www.cmpe.boun.edu.tr/~ethem/files/papers/NC110804.PDF

Standard error

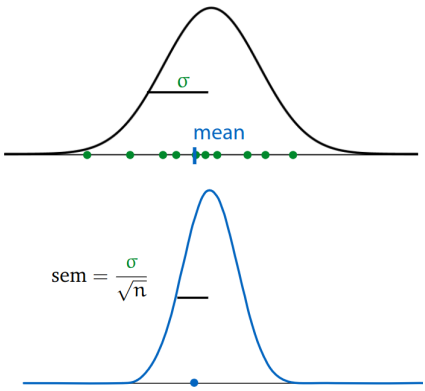
Given N samples $x_1, \ldots, x_N$ from a distribution with a standard deviation of σ.

The mean $\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$ has an associated **standard error of the mean (SEM)**

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{N}}$$

In practice $\sigma_{\bar{x}} \approx \frac{\sigma_x}{\sqrt{N}}$ where σ_x is the corrected sample standard deviation:

$$\sigma_x^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2$$

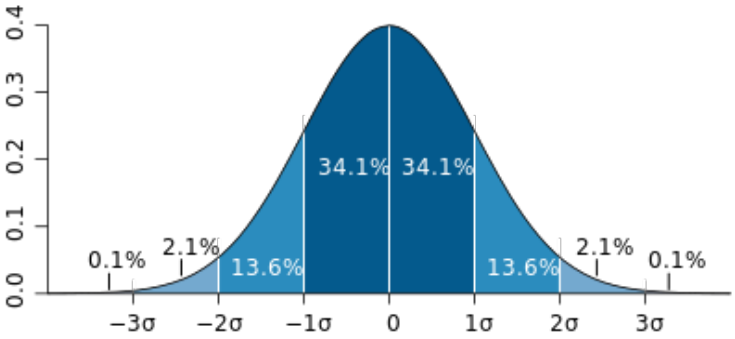


For a Gaussian with unknown σ, the estimated distribution follows the Student t-distribution.

Confidence interval for the mean

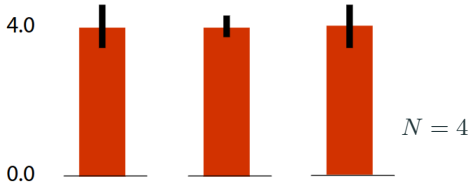
95% confidence interval for the unknown population mean: $\bar{x} \pm 1.96 \cdot \frac{\sigma_x}{\sqrt{N}}$

$1.96 \approx 97.5$ percentile point of the normal distribution.



Error bars

How do we interpret error bars?

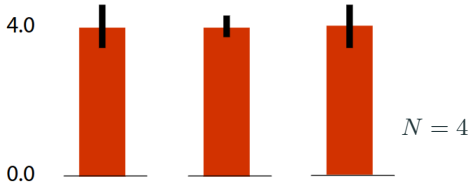


Error bars

How do we interpret error bars?

Standard deviation?
Standard error?
Confidence interval?

No agreed upon convention.

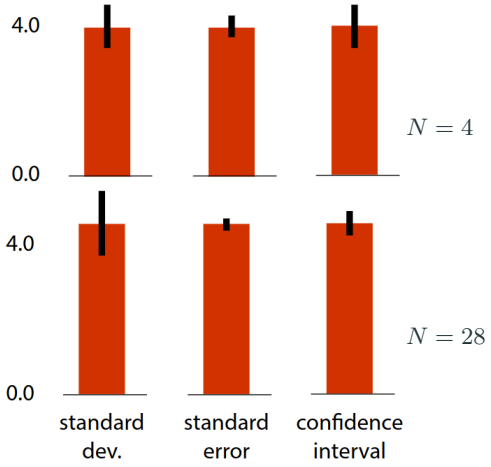


Error bars

How do we interpret error bars?

Standard deviation?
Standard error?
Confidence interval?

No agreed upon convention.



Data above: $3, 4, 6, 3$. Data below: $3, 4, 5, 2, 8, 7, 8, 2, 3, 5, 7, 0, 2, 4, 6, 7, 0, 4, 5, 1, 8, 7, 1, 2, 3, 5, 7, 4$.

Error bars

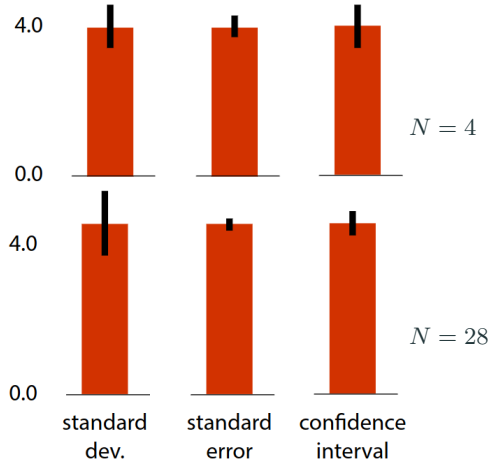
How do we interpret error bars?

Standard deviation?
Standard error?
Confidence interval?

No agreed upon convention.

Standard deviation: measure of **spread**
Standard error, CI: measure of **confidence**

More data → smaller CI, SE; more accurate SD.

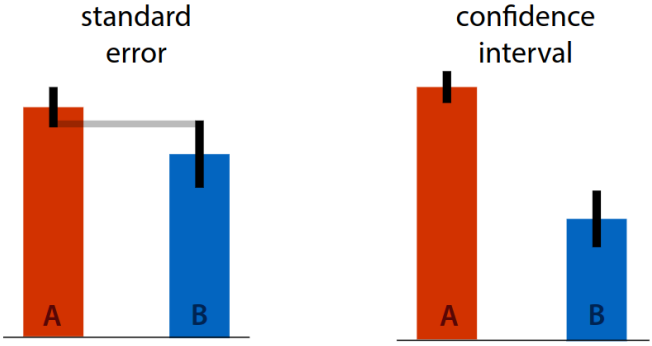


Data above: $3, 4, 6, 3$. Data below: $3, 4, 5, 2, 8, 7, 8, 2, 3, 5, 7, 0, 2, 4, 6, 7, 0, 4, 5, 1, 8, 7, 1, 2, 3, 5, 7, 4$.

Overlap and measures of confidence

Standard error: **overlap** implies **no significant** difference between A and B.

Confidence interval: **no overlap** implies **significant** difference between A and B.



Converse **not true** in both cases!

Showing spread

For a large enough test set, the CI for a measurement is usually small enough.

Still show **spread** (i.e. standard deviation) due to various sources of randomness:

- Data sampling, bootstrapping
- Data splitting, cross-validation
- Learning algorithm (e.g. initialization of gradient descent)

Outline

Metrics need context

Is accuracy of 99% good (enough)?

Metrics need context

Is accuracy of 99% good (enough)?

For a reasonable answer we need context:

- What is the application? Is it safety-critical?
- Is there **class imbalance**?

  Often (e.g. in medical settings) the distribution of labels is highly imbalanced.
  A test set with very few positives: only few ”levels of performance”.

Metrics need context

Is accuracy of 99% good (enough)?

For a reasonable answer we need context:

- What is the application? Is it safety-critical?

- Is there **class imbalance**?

  Often (e.g. in medical settings) the distribution of labels is highly imbalanced.
  A test set with very few positives: only few ”levels of performance”.

- Is there **cost imbalance**?

  How much worse is a mislabeled positive than a mislabeled negative?

  Disease diagnosis: miss a sick patient vs. invasive test to a healthy patient
  Spam classification: miss a valid email vs. show some spam
  Fraud detection: miss fraud vs. waste valuable expert time

Confusion matrix (contingency table)

Assume binary classification, $y \in \{0, 1\} = \{-1, 1\} = \{\text{neg}, \text{pos}\}$.

		predicted: pos	predicted: neg
true	pos	TP: true positive	FN: false negative
	neg	FP: false positive	TN: true negative

Generalizes to a $|\mathcal{Y}| \times |\mathcal{Y}|$ matrix where the entries on the diagonal are correct.

predicted

$$\text{true} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kk} \end{bmatrix}$$

Confusion matrix (contingency table)

Assume binary classification, $y \in \{0, 1\} = \{-1, 1\} = \{\text{neg}, \text{pos}\}$.

		predicted: pos	predicted: neg
true	pos	TP: true positive	FN: false negative
	neg	FP: false positive	TN: true negative

Generalizes to a $|\mathcal{Y}| \times |\mathcal{Y}|$ matrix where the entries on the diagonal are correct.

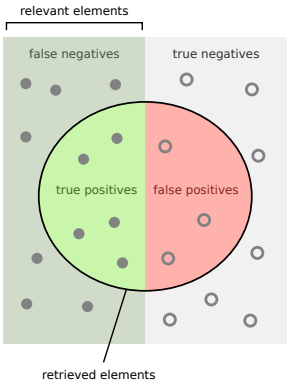
$$\begin{array}{c} \\ \text{true} \end{array} \begin{array}{c} \text{predicted} \\ \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kk} \end{bmatrix} \end{array}$$

Precision and Recall

$\frac{TP}{TP+FP}$ is **precision**: what proportion of returned positives are actually positive?

$\frac{TP}{TP+FN}$ is **recall**: what proportion of existing positives did we find?

		predicted pos	predicted neg
true	pos	TP	FN
	neg	FP	TN



Other metrics derived from the confusion matrix (wiki link)

Sources: [22][23][24][25][26][27][28][29][30] view · talk · edit

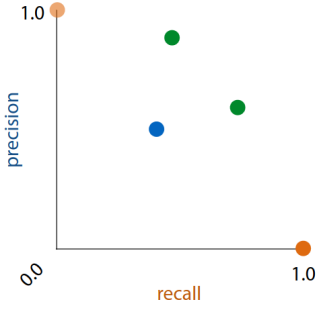
		Predicted condition			
	Total population = P + N	**Positive (PP)**	**Negative (PN)**	Informedness, bookmaker informedness (BM) = TPR + TNR − 1	Prevalence threshold (PT) = $\frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	**Positive (P)**	**True positive (TP),** hit	**False negative (FN),** type II error, miss, underestimation	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power = $\frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate = $\frac{FN}{P} = 1 - TPR$
	Negative (N)	**False positive (FP),** type I error, false alarm, overestimation	**True negative (TN),** correct rejection	False positive rate (FPR), probability of false alarm, fall-out = $\frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity = $\frac{TN}{N} = 1 - FPR$
	Prevalence = $\frac{P}{P+N}$	Positive predictive value (PPV), precision = $\frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) = $\frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) = $\frac{TPR}{FPR}$	Negative likelihood ratio (LR−) = $\frac{FNR}{TNR}$
	Accuracy (ACC) = $\frac{TP+TN}{P+N}$	False discovery rate (FDR) = $\frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) = $\frac{TN}{PN}$ = 1 − FOR	Markedness (MK), deltaP (Δp) = PPV + NPV − 1	Diagnostic odds ratio (DOR) = $\frac{LR+}{LR-}$
	Balanced accuracy (BA) = $\frac{TPR+TNR}{2}$	F_1 score = $\frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) = $\sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) = $\sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index = $\frac{TP}{TP + FN + FP}$

Comparing binary classifiers

Points in the corner are the most extreme options:

- call everything positive → recall of 1
- call only most likely positive → precision of 1

Green classifiers pareto dominate (improve on both metrics) the blue classifier.

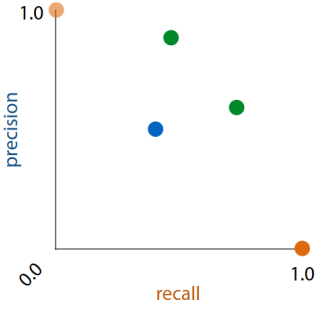


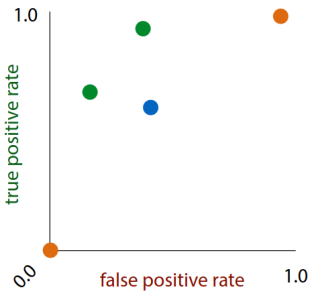
Comparing binary classifiers

Points in the corner are the most extreme options:

- call everything positive → recall of 1
- call only most likely positive → precision of 1

Green classifiers pareto dominate (improve on both metrics) the blue classifier.

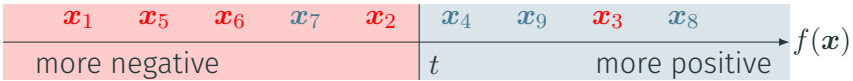




ROC curves to analyze a single binary classifer

Classifiers return a score $f(\boldsymbol{x})$ of how likely is an instance to be positive.

Predict positive everything above a default threshold t, e.g. $p(y = 1 \mid \boldsymbol{x}) > 0.5$.



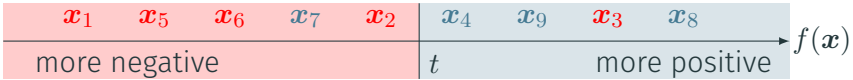
Different thresholds give different TPR / FPR trade-offs.

Try all thresholds to trace out the ROC curve.

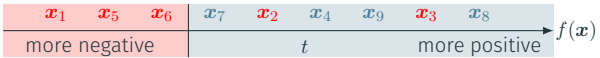
ROC curves to analyze a single binary classifer

Classifiers return a score $f(\boldsymbol{x})$ of how likely is an instance to be positive.

Predict positive everything above a default threshold t, e.g. $p(y = 1 \mid \boldsymbol{x}) > 0.5$.



Different thresholds give different TPR / FPR trade-offs.



Try all thresholds to trace out the ROC curve.

Area under the ROC curve

To obtain any classifier on the line between two points: pick the output at random, e.g. one with probability α and the other with $1 - \alpha$.

Area under the ROC curve

To obtain any classifier on the line between two points: pick the output at random, e.g. one with probability α and the other with $1 - \alpha$.

Area under the ROC curve is a good indicator of quality, called AUC-ROC.

AUC-ROC estimates the probability of a correctly ordered random pair.

Summary

Carefully set up your model evaluation pipeline – cross-validation and properly split data (e.g. ”future” data in the test when time is involved).

Model generalizability via early stopping and carefully select hyperparameters.

Downsides are even in well-established benchmarks.

Use statistical testing (CI, standard error and standard deviation) to compare evaluation performance between different models.

Do not consider *only* the accuracy of your model. Look at different metrics that capture possible settings of the task (class imbalance, ...).

Reading material

Main reading

- ”Probabilistic Machine Learning: An Introduction” by Murphy
  [ch. 4.7.1 - 4.7.4, ch 5.1.2 - 5.1.3]

Extra reading

- ”Combined 5×2 cv F Test for Comparing Supervised Classification Learning Algorithms” by Ethem Alpaydın
- ”Random Search for Hyper-Parameter Optimization” by Bergstra et al.