Performance evaluation of multivariate predictive models in neuroimaging

Jonas Richiardi

https://www.stanford.edu/~richiard

UNIVERSITÉ DE GENÈVE

Dept. of Neuroscience

Agenda

- Principles
- Algorithms for estimating performance
- Point estimates of classification performance
- Point estimates of regression performance
- Confidence intervals

Agenda

- Principles
- Algorithms for estimating performance
- Point estimates of classification performance
- Point estimates of regression performance
- Confidence intervals

Quick intro - learning and decision theory

- Classification - learn a function f that predicts one of C discrete categories (**class label**) from a feature vector:

$$\Omega = \{\omega_1, \ldots, \omega_C\}$$

$$f(\mathbf{x}) : \mathbb{R}^D \rightarrow \Omega$$

- Regression - learn a function f that predicts a scalar value (**target**) from a feature vector:

$$f(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^1$$

- In both cases these are approximation to an unknown “real-world” function F that generates features and labels/targets.

Empirical risk minimisation

- We want to learn model parameters that minimise the risk of making an erroneous predictions
  - but the real joint distribution of features and class labels or regression targets “in the world” is unknown
  - so we must approximate the “real” risk by the *empirical* risk: the average error or **loss** on the training sample:

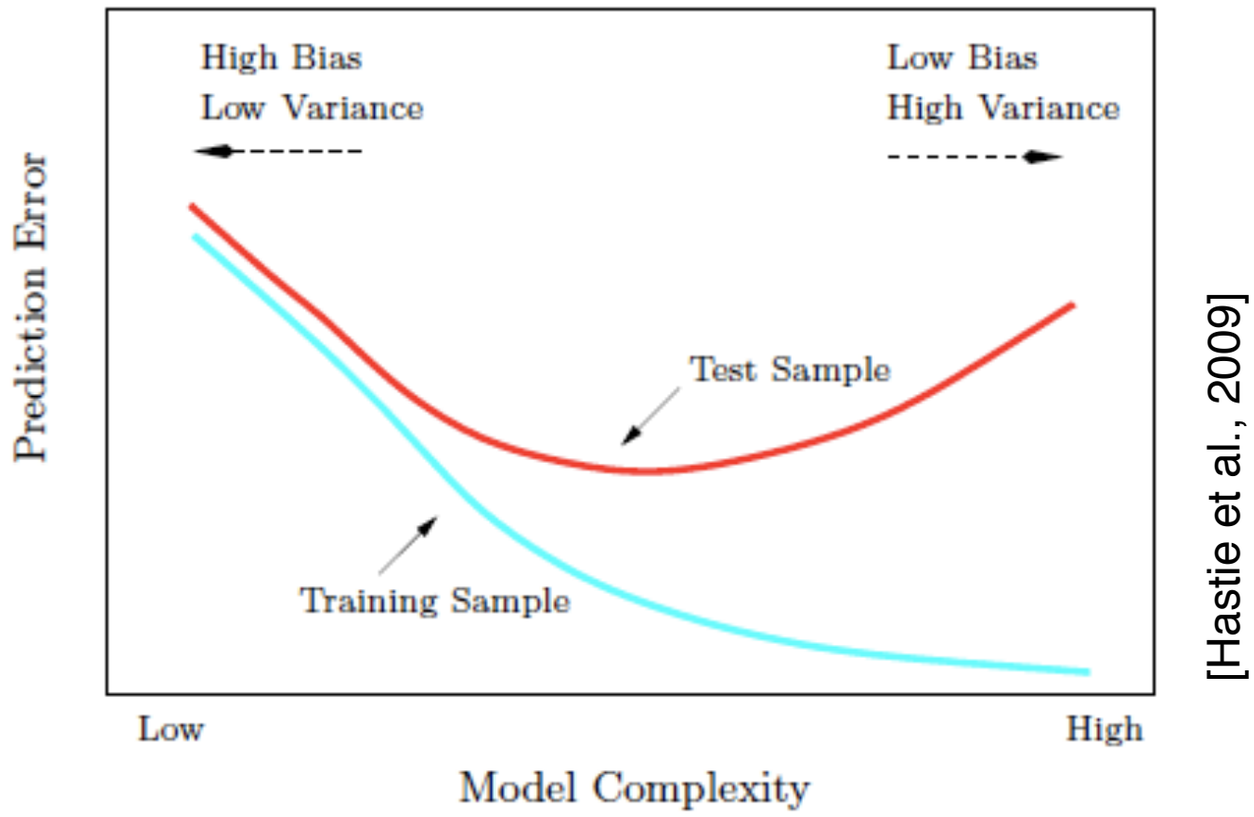
$$R(f, \mathbf{X}) = \frac{1}{N} \sum l(\omega_n, f(\mathbf{x}_n))$$

  - l() is a **loss function**, penalising bad predictions $\omega_n \neq f(\mathbf{x}_n)$
- Many loss functions can be used; a very common one for classification is 0-1 (or symmetrical) loss:

$$l_{01}(\omega_n, f(\mathbf{x}_n)) = \begin{cases} 0 & \omega_n = f(\mathbf{x}_n) \\ 1 & \omega_n \neq f(\mathbf{x}_n) \end{cases}$$

Training, testing, and overfitting

- Minimising the risk functional on the training set will encourage you to use more complex models and is not a guarantee of good **generalisation**



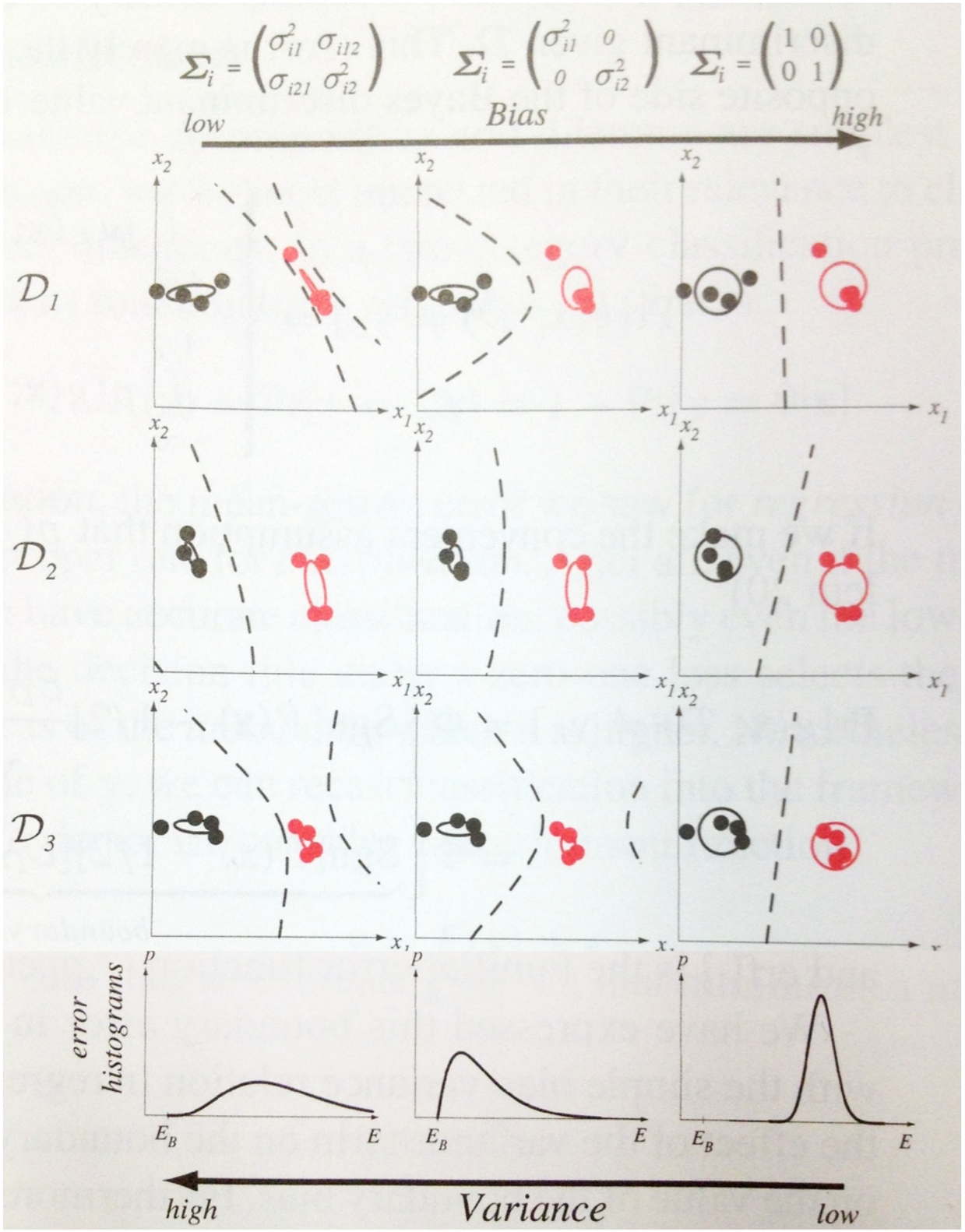
[Hastie et al., 2009]

- In practice, we minimise risk while penalising more complex models (**regularisation**):

$$R_r(f, \mathbf{X}, \lambda) = R(f, \mathbf{X}) + \lambda ||f||^2$$

Bias, variance, and stability

- The **bias-variance tradeoff** applies in predictive models:
  - We can decrease prediction error arbitrarily on a given dataset, thus yielding low **bias**.
- However, this systematically comes at a price in **variance**: the parameters of f can change a lot if the training set varies. This **instability** can actually be exploited in ensembling.



[Duda et al, 2001]

Agenda

- Principles
- Algorithms for estimating performance
- Point estimates of classification performance
- Point estimates of regression performance
- Confidence intervals

Empirical evaluation of performance

Empirical evaluation of performance

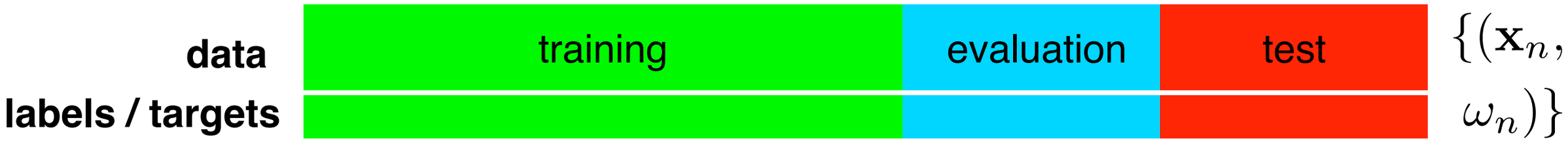
- To have some confidence in our performance estimate we must **learn (train)** the model on one dataset, and evaluate it on another dataset
  - If we don't do that, it is hard to substantiate claims of generalisation
  - The error rate on the training set (resubstitution error) can be made arbitrarily small... just use a complex model
- Remember our goal: predict **unseen** data.

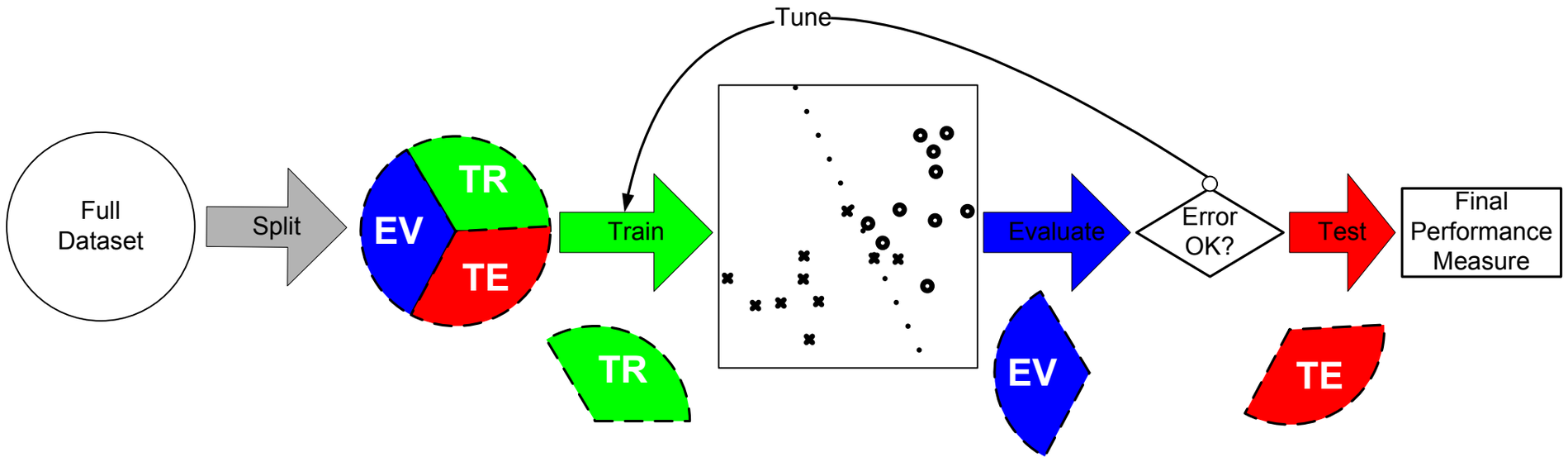
Empirical evaluation of performance

- To have some confidence in our performance estimate we must **learn (train)** the model on one dataset, and evaluate it on another dataset
  - If we don't do that, it is hard to substantiate claims of generalisation
  - The error rate on the training set (resubstitution error) can be made arbitrarily small... just use a complex model
  - Remember our goal: predict **unseen** data.
- Maintain strict train/test separation throughout the processing
  - Don't select voxels on the whole dataset !
  - Always ask yourself: **what would I do if I acquired data for one more subject after my model is trained**?

Hold-out technique

- Split the data into 2 or 3 parts





- This makes sub-optimal use of the data available
- Pessimistic bias

M-fold cross-validation technique

Split in folds, then train classifier...

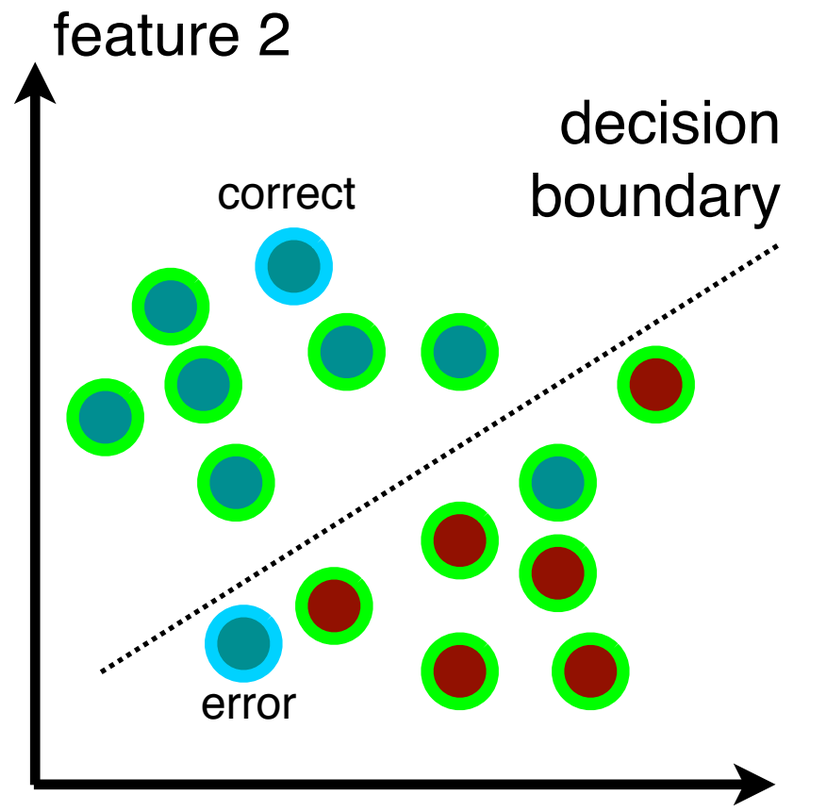
...and evaluate it

Rotate partitions and repeat

...

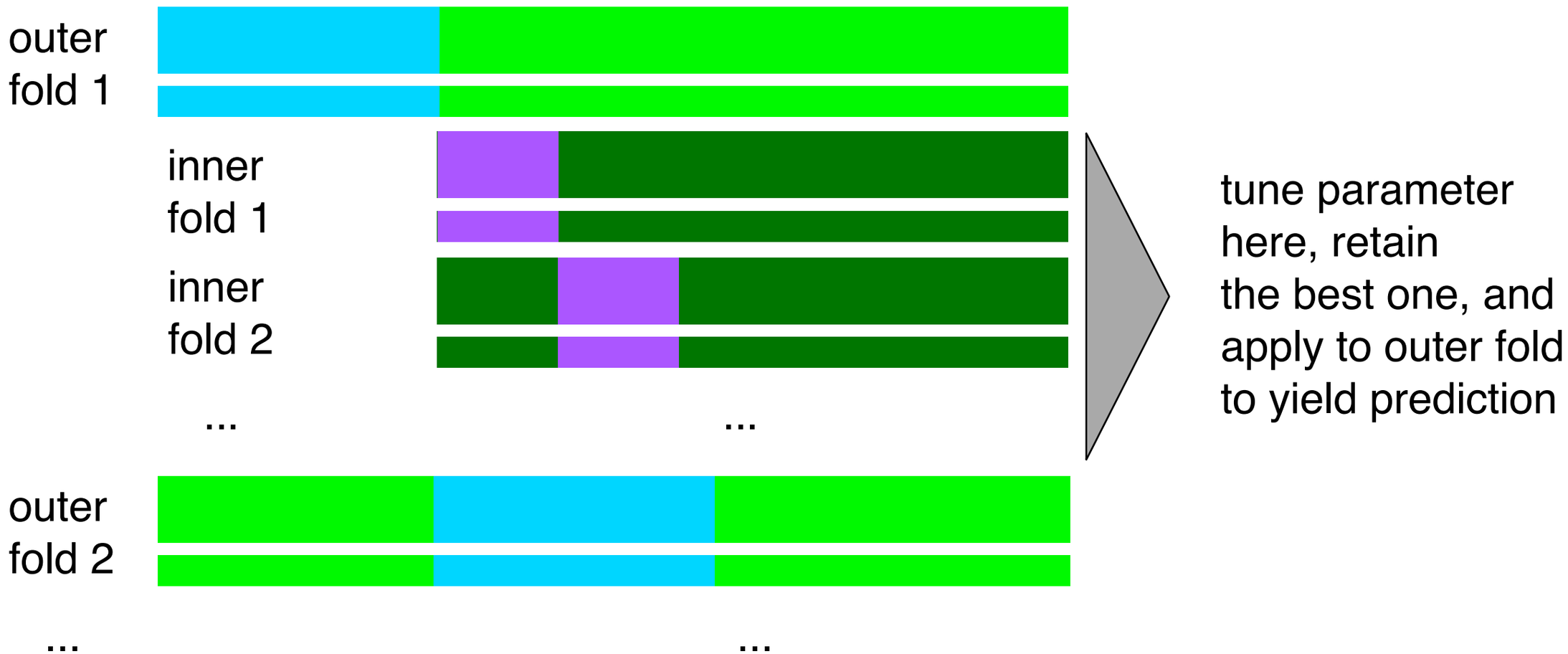
Ultimately test (once)

Applies to scans, blocks, subjects



Nested cross-validation

- If we need to tune the classifier (typically the regularisation parameter λ) or select features, we must maintain train/test separation!
  - Add a **nested cross-validation** loop within the main loop



- Not doing this is a surprisingly common mistake

Agenda

- Principles
- Algorithms for estimating performance
- Point estimates of classification performance
- Point estimates of regression performance
- Confidence intervals

Confusion matrix

Confusion matrix

- **Accuracy** statistics can be shown in a **confusion matrix** (summary table):
  - Class 0 accuracy (p_0) = A/(A+B)
  - Class 1 accuracy (p_1) = D/(C+D)
  - Accuracy (p) = (A+D)/(A+B+C+D)
  - With 0-1 loss, we have

truth \ $\hat{\omega}$	ω_0	ω_1
ω_0	A	B
ω_1	C	D

$$p = 1 - \frac{1}{N}\sum l_{01}(\omega_n, f(\mathbf{x}_n))$$

- Biostats (ω_0 =healthy, ω_1 =disease):
  - sensitivity (“What is the probability that the disease is PRESENT and that I detect it?”): D/(C+D) = TP/(FN+TP)
  - specificity: (“disease is ABSENT and I report no disease”: A/(A+B) = TN/(FP+TN)

Confusion matrix

- **Accuracy** statistics can be shown in a **confusion matrix** (summary table):
  - Class 0 accuracy (p_0) = A/(A+B)
  - Class 1 accuracy (p_1) = D/(C+D)
  - Accuracy (p) = (A+D)/(A+B+C+D)
  - With 0-1 loss, we have

truth \ $\hat{\omega}$	ω_0	ω_1
ω_0	A	B
ω_1	C	D

$$p = 1 - \frac{1}{N}\sum l_{01}(\omega_n, f(\mathbf{x}_n))$$

  - Biostats (ω_0 =healthy, ω_1 =disease):
    - sensitivity ("What is the probability that the disease is PRESENT and that I detect it?"): D/(C+D) = TP/(FN+TP)
    - specificity: ("disease is ABSENT and I report no disease": A/(A+B) = TN/(FP+TN)
- Perfect: B=C=0. **Be suspicious if this happens!**

Confusion matrix

- **Accuracy** statistics can be shown in a **confusion matrix** (summary table):
  - Class 0 accuracy (p_0) = A/(A+B)
  - Class 1 accuracy (p_1) = D/(C+D)
  - Accuracy (p) = (A+D)/(A+B+C+D)
  - With 0-1 loss, we have

truth \ $\hat{\omega}$	ω_0	ω_1
ω_0	A	B
ω_1	C	D

$$p = 1 - \frac{1}{N}\sum l_{01}(\omega_n, f(\mathbf{x}_n))$$

  - Biostats (ω_0 =healthy, ω_1 =disease):
    - sensitivity ("What is the probability that the disease is PRESENT and that I detect it?"): D/(C+D) = TP/(FN+TP)
    - specificity: ("disease is ABSENT and I report no disease": A/(A+B) = TN/(FP+TN)
- Perfect: B=C=0. **Be suspicious if this happens!**
- Random: A=B=C=D. Same as flipping a coin.

Balance in datasets

- If the classes don't all have the same number of examples
  - The classifier may sacrifice performance on the **minority classes** to improve on the **majority class**
  - Accuracy may seem to be above chance whereas the minority classes are sacrificed and below chance
- A common strategy is to subsample the majority class, but data is lost
- Reporting class accuracies (p_0,..., p_C) is good practice
- Balanced accuracy is computed as $p^{bal} = \frac{1}{C}\sum p_c$

‘Soft’ versus ‘hard’ classifier outputs

- Many classifiers first compute a “soft” function value

$$g(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^1$$

- Then “hard” decisions are taken by a simple function

$$f(g(\mathbf{x})) : \mathbb{R}^1 \rightarrow \Omega$$

$$f(g(\mathbf{x})) := sgn(g(\mathbf{x}))$$

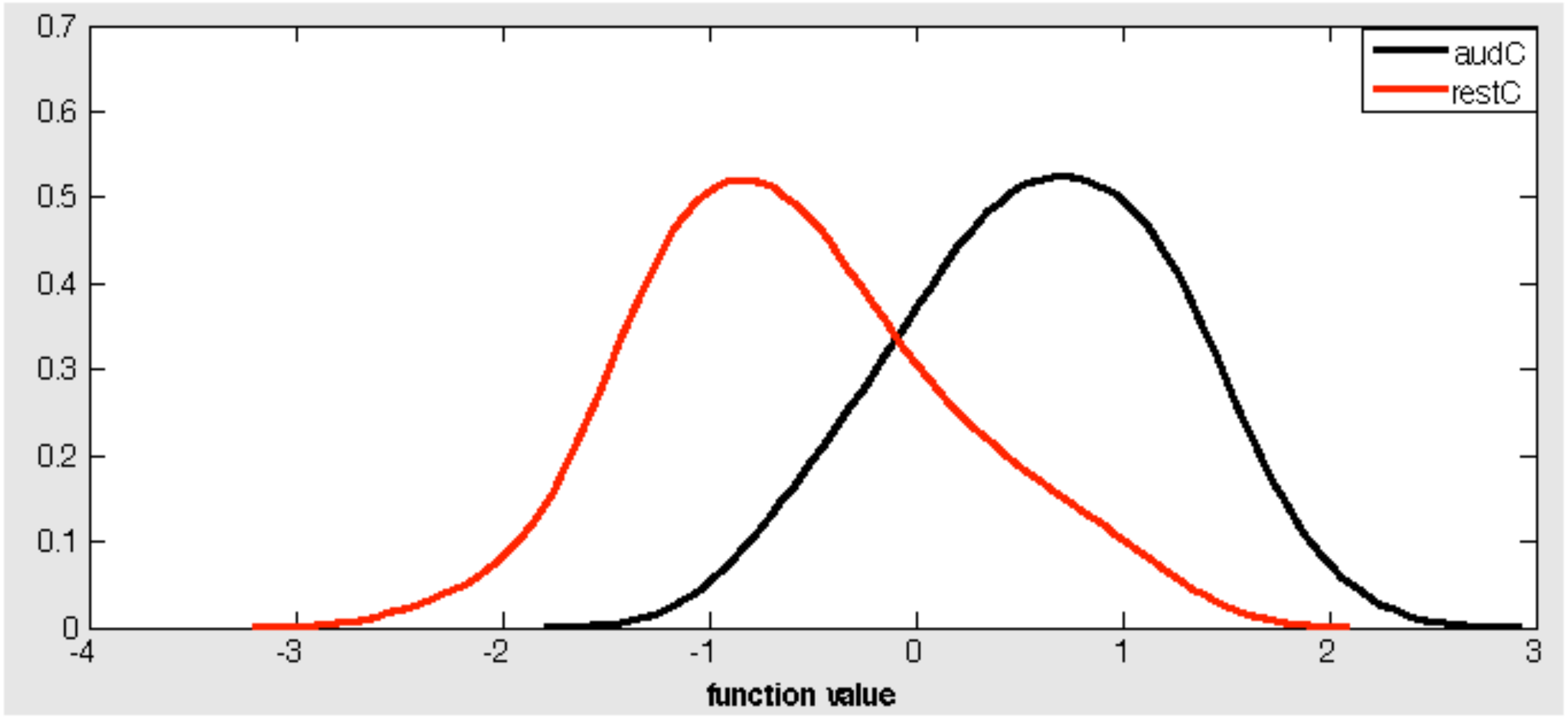
- Both g() and f() contribute to accuracy figure
  - f() defines the decision boundary or threshold
  - Typically not much choice - it’s built in
- looking at g() output is informative

Sensitivity / specificity trade-off

- For a fixed classifier, **increasing sensitivity can only come at the cost of decreasing specificity**, and vice-versa.

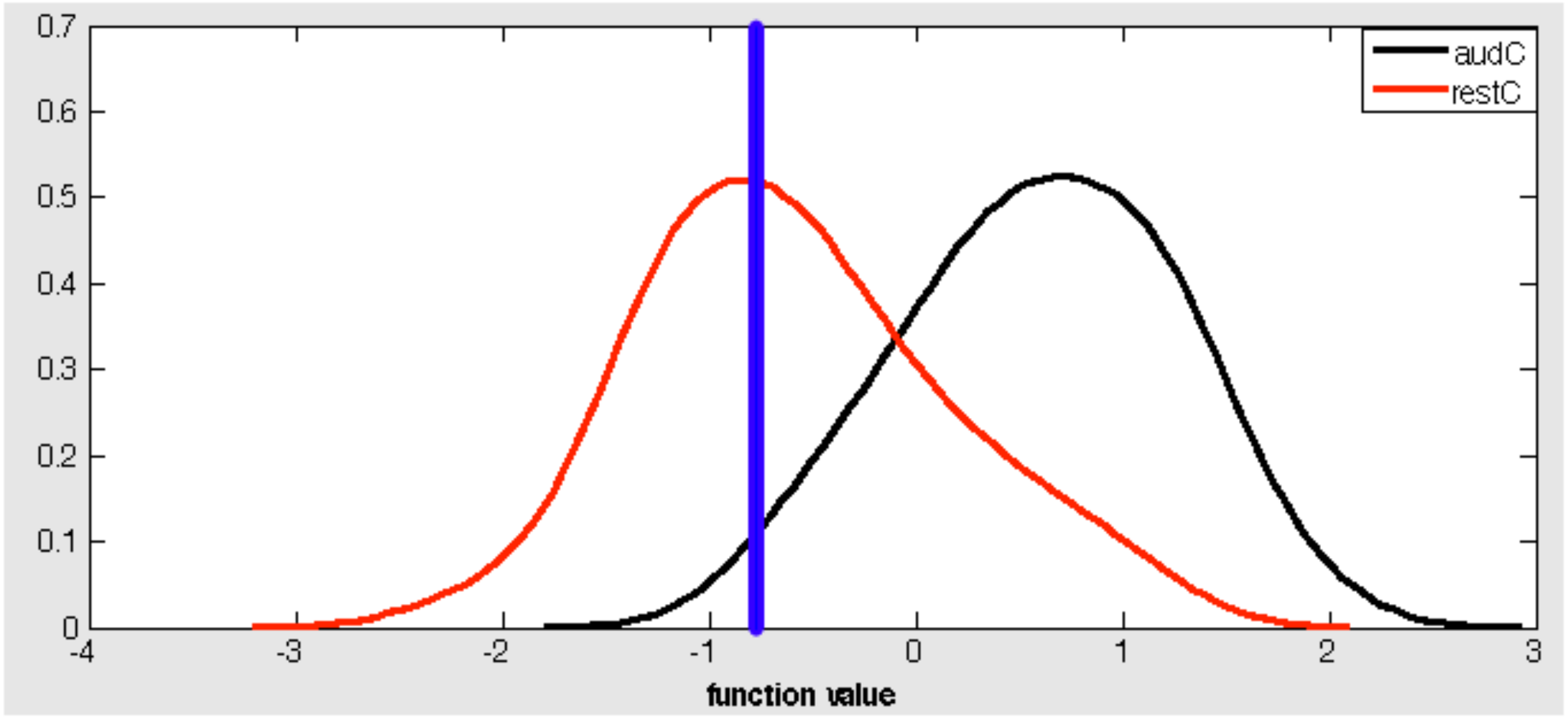
Sensitivity / specificity trade-off

- For a fixed classifier, **increasing sensitivity can only come at the cost of decreasing specificity**, and vice-versa.



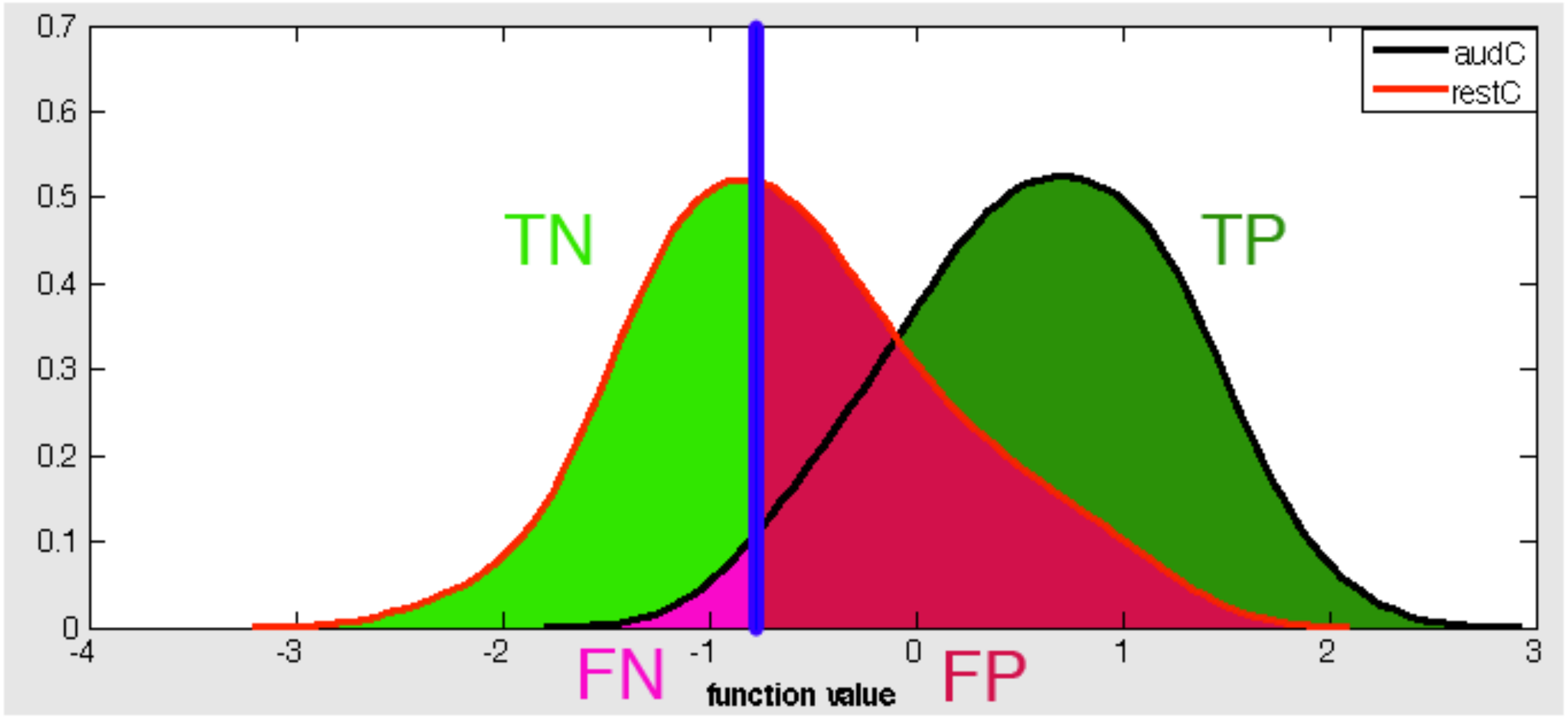
Sensitivity / specificity trade-off

- For a fixed classifier, **increasing sensitivity can only come at the cost of decreasing specificity**, and vice-versa.



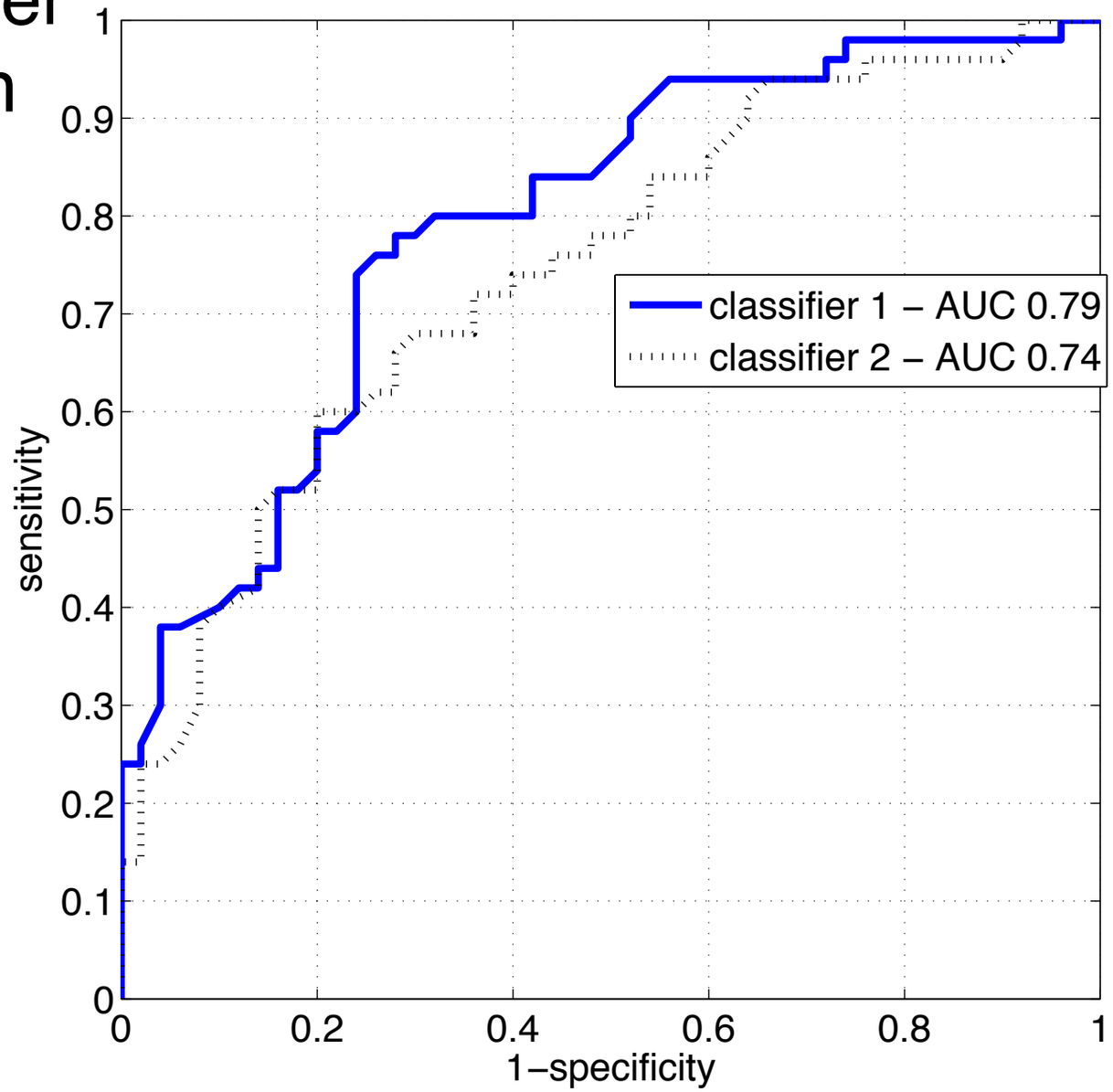
Sensitivity / specificity trade-off

- For a fixed classifier, **increasing sensitivity can only come at the cost of decreasing specificity**, and vice-versa.



Graphical representations: the ROC curve

- The Receiver Operating Characteristic (curve) is a good way of seeing the sens/spec tradeoff over the operating range of a classifier.
  - It is also used for classifier comparison (with caution if lines cross !)
- We can compute the Area Under Curve (AUC) as a summary measure of performance
  - AUC = 1.0: perfect
  - AUC = 0.5: chance



Agenda

- Principles
- Algorithms for estimating performance
- Point estimates of classification performance
- Point estimates of regression performance
- Confidence intervals

Mean squared error

- One of many ways to quantify error
- LOO error in one fold

$$SE_n = (y_n - f(\mathbf{x}_n))^2$$

- Across LOO folds:

$$R(f, \mathbf{X}) = MSE = \frac{1}{N}\sum_{n=1}^{N}(y_n - f(\mathbf{x}_n))^2$$

- This is exactly the same as used in classical inference (although we will have lost some efficiency due to the cross-validation), except we quantify **out-of-sample** MSE
- Models with lower MSE predict better

Correlation of prediction with targets

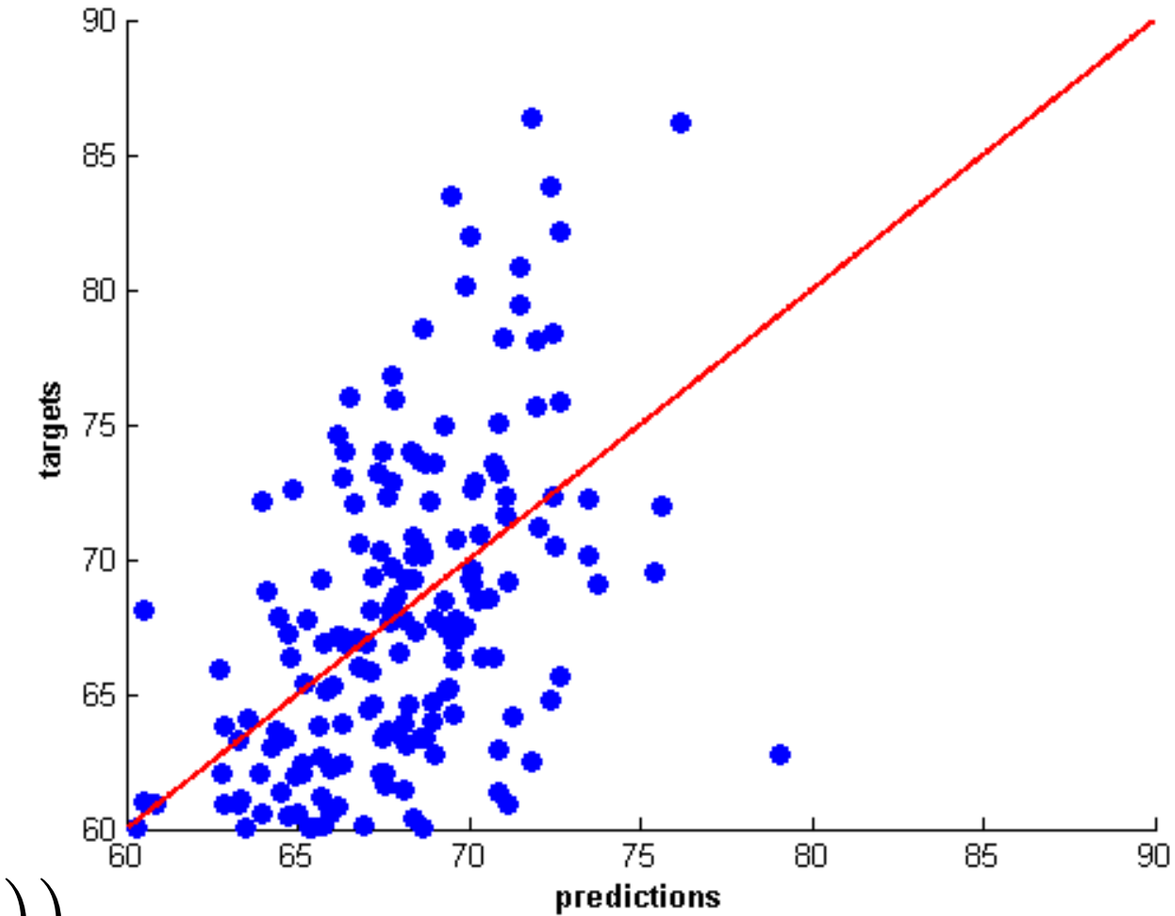
- Correlation coefficient r_{fy} (across folds, depends on CV)

$$S_{yy} = \sum_n (y_n - \bar{y})^2$$

$$S_{ff} = \sum_n (f(\mathbf{x}_n) - f(\bar{\mathbf{x}}))^2$$

$$S_{fy} = \sum_n (y_n - \bar{y})(f(\mathbf{x}_n) - \bar{f}(\mathbf{x}))$$

$$r_{fy} = \frac{S_{fy}}{\sqrt{S_{yy} S_{ff}}}$$



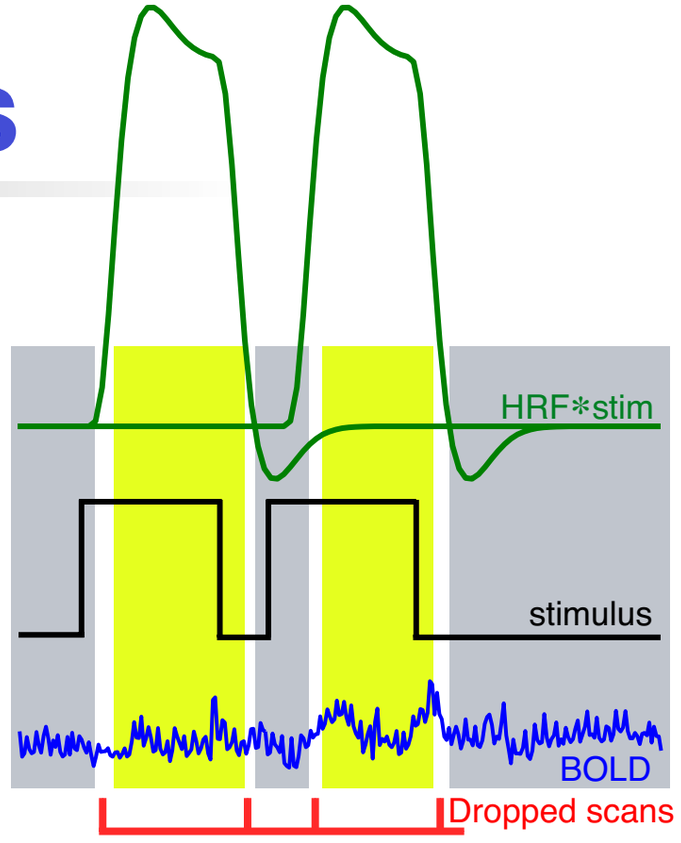
- note $r_{fy}^2 = R^2$

Agenda

- Principles
- Algorithms for estimating performance
- Point estimates of classification performance
- Point estimates of regression performance
- Confidence intervals

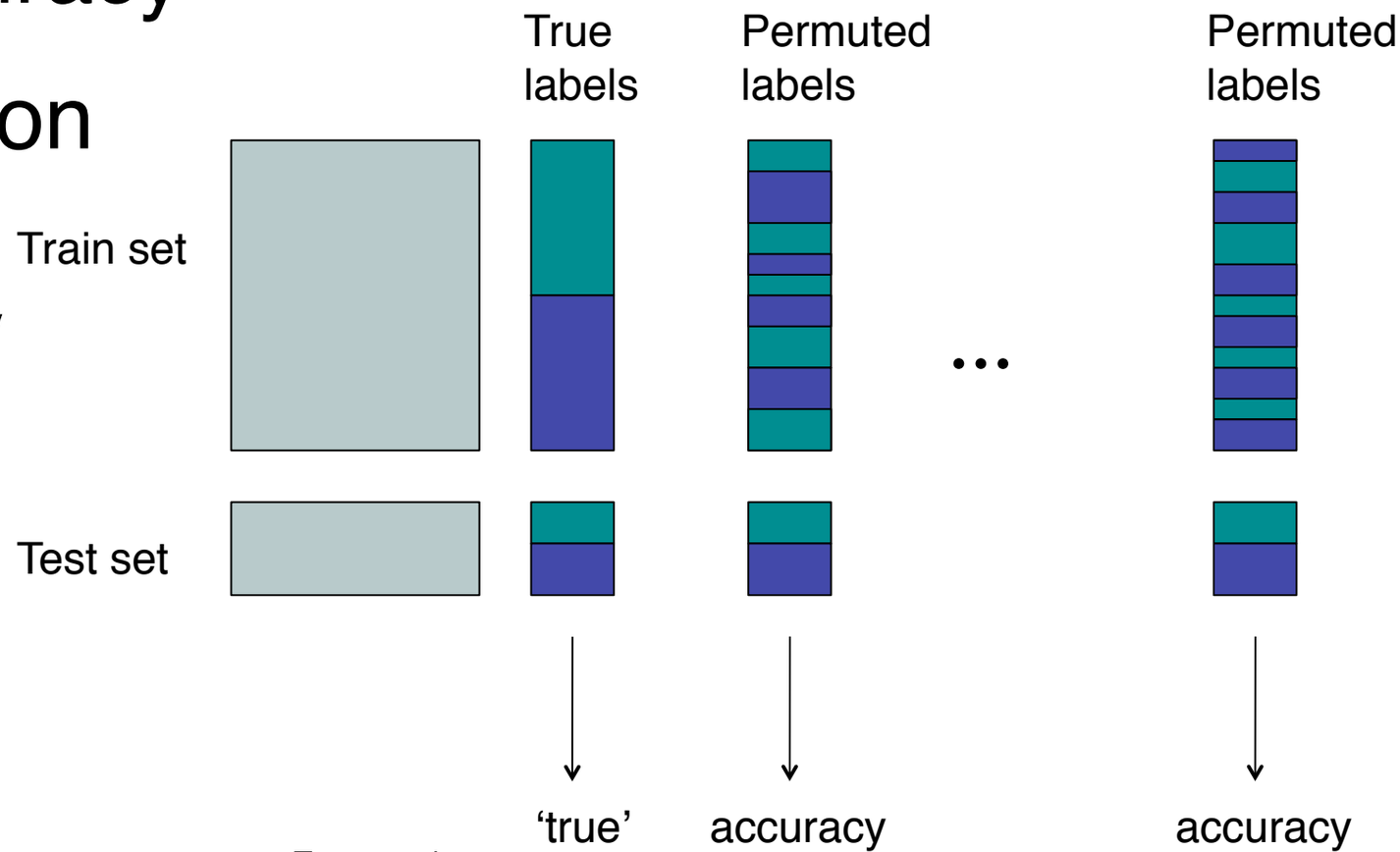
Parametric confidence intervals

- Parametric tests imply assumptions
  - IID samples
    - but! Correlation between scans
  - $\sim \mathcal{N}$
- Wilson Binomial score test
  - Model decisions in two-class classification as a binomial distribution. Accuracy = “probability of success”
  - Works well for small (<40) and large n.
  - Confidence intervals at 95% (but coverage probability can be impacted by violated assumptions)
  - Closed-form expression -> fast
  - gives upper and lower bounds, e.g. 40/60 subjects classified = 66.7% pointwise, 5% CI [54.1% 77.3%]



Non-parametric: permutation testing

- No hypotheses on data distribution
- H_0: “class labels are non-informative”
- Test statistic: CV accuracy
- Estimate the distribution of the test statistic under H_0 by randomly permuting labels M times, and running a full CV experiment



- p-value: $\frac{1}{M+1}\left(\left[\sum_{m}^{M}\delta(p_m^{perm} > p^{real})\right]+1\right)$

Comparing classifiers

- "Is classifier θ_A better than classifier θ_B?"
- Hypothesis testing framework! "Do the two sets of decisions of classifiers θ_A and θ_B represent two different populations?"

test sample	1	2	3	4	5	6	7	8	9	10
θ_A	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘
θ_B	✔	✔	✘	✔	✔	✔	✘	✔	✔	✘

- Careful: Samples (sets of decisions) are **dependent** (same evaluation/testing dataset), and measurement type is **categorical binary** (for a two-class problem)
- ⇒Use **McNemar** test. H_0: "there is no difference between the accuracies of the two classifiers"

McNemar Test

- Compute relationships between classifier decisions

	θ_B ✔	θ_B ✘
θ_A ✔	N_{11}	N_{10}
θ_A ✘	N_{01}	N_{00}

- If H_0 is true, we should have $N_{01} = N_{10} = 0.5(N_{01}+N_{10})$
- We can measure the discrepancy between this and observed counts using the statistic:

$$x^2 = \frac{(|N_{01} - N_{10}| - 1)^2}{N_{01} + N_{10}}$$

- Then compare x^2 to critical value of χ^2 at a level of significance α.

Recommended reading

- Duda et al., Pattern Recognition, Wiley, 2001
- Hastie et al., The elements of statistical learning, Springer, 2009
- Pereira et al., *Machine learning classifiers and fMRI: A tutorial overview*, NeuroImage 45, 2009
- Kriegeskorte et al., *Circular analysis in systems neuroscience: the dangers of double dipping*, Nature Neuroscience 12, 2009
- Kohavi, *A study of cross-validation and bootstrap for accuracy estimation and model selection*, IJCAI, 1995