COMPUTER SCIENCE
INDIANA UNIVERSITY
School of Informatics and Computing
Bloomington

Performance measures

Reminders/comments

- Assignment 4
  - For the first question, there *may not* be a solution
  - For the kernels question, if you are getting terrible performance, consider the choices for your kernel algorithm.

Experimental set-up

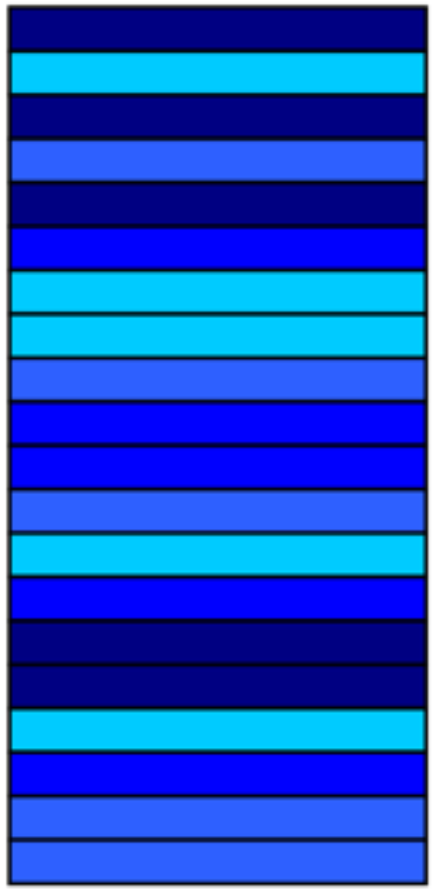
- Cross-validation? External or Internal?
- Repeated subsampling?
- Careful statistical work done on executing empirical studies; pros and cons to each (and some are also just wrong)
- My personal preference:
    - Repeated subsampling to generate training/validation splits (can choose as many as desired, say m)
    - k-fold cross-validation on training set to select parameters

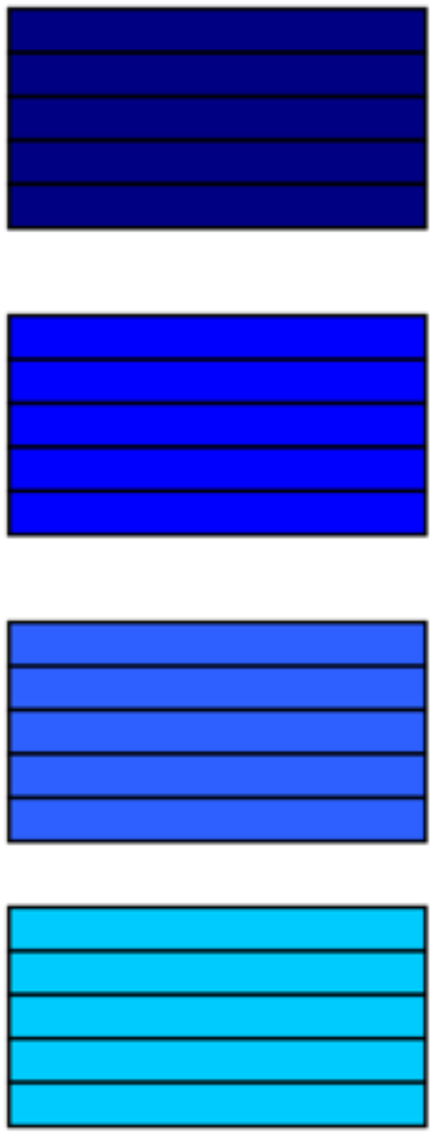
k-fold CV for training set i

Randomly and evenly split into 4 non-overlapping partitions

D

20 data points





Partition 1.

Data points: 1, 3, 5, 15, 16

Partition 2.

Data points: 6, 10, 11, 14, 17

Partition 3.

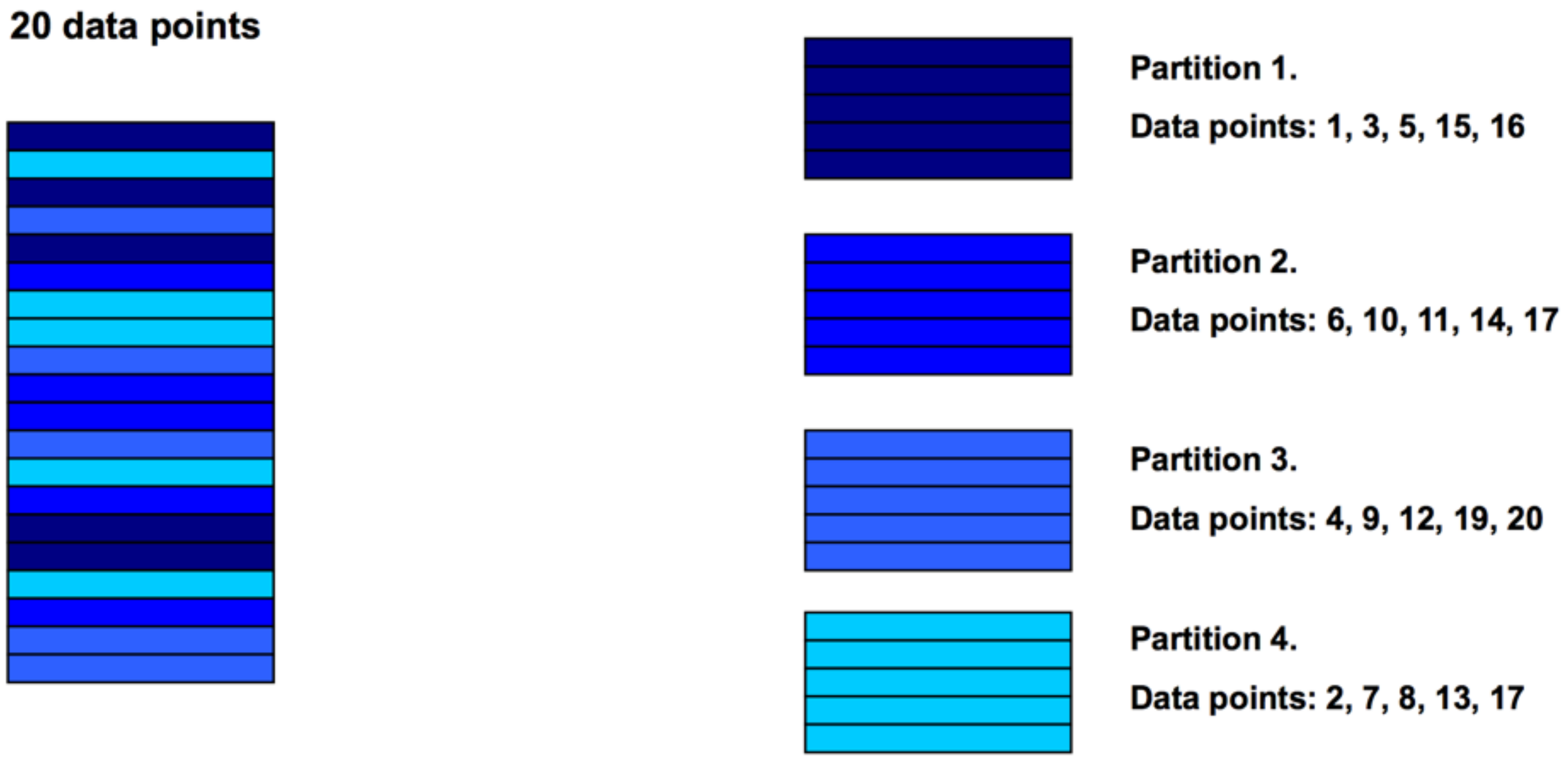
Data points: 4, 9, 12, 19, 20

Partition 4.

Data points: 2, 7, 8, 13, 17

k-fold CV for training set i

- For parameter setting j, learn model on k-1 folds and test on the hold-out fold (done k times); average k error estimates
- Select parameter setting j that gives lowest average error


20 data points
Partition 1.
Data points: 1, 3, 5, 15, 16
Partition 2.
Data points: 6, 10, 11, 14, 17
Partition 3.
Data points: 4, 9, 12, 19, 20
Partition 4.
Data points: 2, 7, 8, 13, 17

Training set i

- Once “best” parameters obtained for the training set, learn on the entire training set i with those parameters, and validate on validation set i
- This gives you an estimate of error for training/validation split i
- Over all m splits, get m estimates of error (different from k)

Experiments

- "What if I cannot find any difference between the algorithms?"
  - If you ran a fair experiment, with lots of repetitions (random splits of training and test) to get a large enough number of samples of the error, then that is a fine result
  - Remember that algorithms have many parameters that can strongly affect their performance
- "How do I select parameter ranges?"
  - the best is to provide a large enough range; this can be slow
- "Do I have to sweep all parameters in the CV?"
  - No, but remember that any choice of parameters affects your conclusion —> it is much less interesting to conclude that linear regression with regularization weight = 0.1 is outperformed by a NN with

Why CV?

- CV a reasonable strategy to allow data to pick your parameters: measures on folds give idea of generalization
- Also simulates how someone (or you) might use the model in practice:
  - given a dataset set, you might do 10-fold CV to select across your meta-parameters
  - learn one model with those meta-parameters and now can use for future test data (which we do not have access to)

What to avoid

- Data snooping: looking at test data and iteratively modifying your model
  - BAD: do CV with set of parameters on training data, then see if model did well on test; if did not do well, go back and change set of meta-parameters given to CV
  - GOOD: do CV with set of parameters on training data, see average errors on k-folds, refine set of meta-parameters and then finally complete the CV (all only using the training data)
- Data snooping will typically give you an overestimate of the performance of your model, since you “overfit” to your test
- Leave-one-out CV: higher variance, not usually a good choice

What if I have another approach to selecting meta-parameters?

- Imagine that you are comparing two algorithms, with proposed approaches for automatically selecting their meta-parameters
- These are typically called “meta-parameter free”
- How does this change the experiment?
- What are you trying to test?

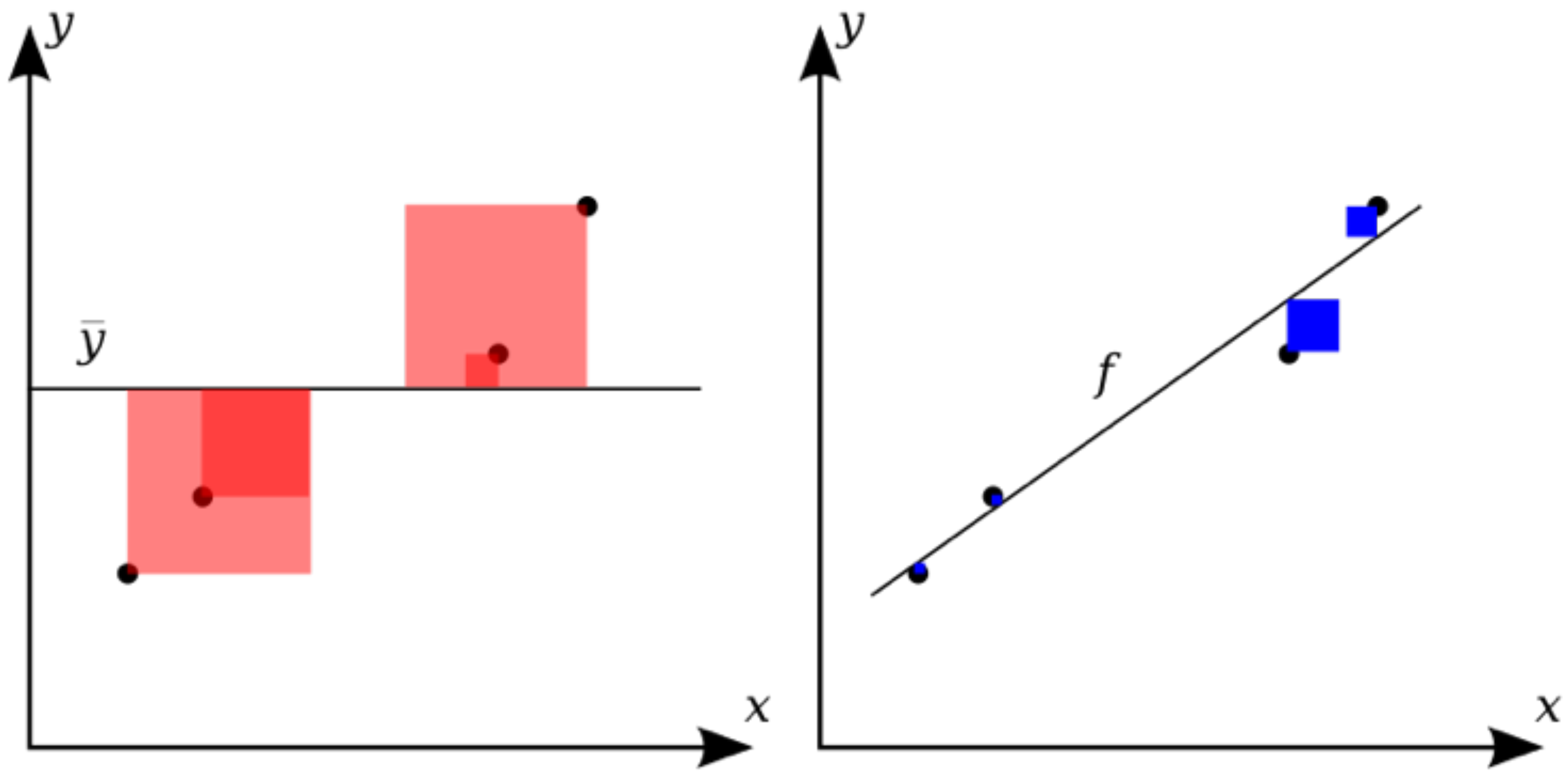
Regression objectives

- We have looked at l2 error for estimating parameters (i.e., as an objective) and to measure performance

- Other options:
  - l1 error — can be difficult to optimize, still a useful measure of error
  - smooth l1 — smooth and convex, easier to optimize, not usually used as a measure of error (unless reporting accuracy of optimizer)
  - R-squared — coefficient of determination
  - Variance unexplained
  - Percentage error — rescale by magnitude of values

R-squared measure

- Also called “coefficient of determination”



- The sum of squares of residuals, also called the residual sum of squares:

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2$$

- The total sum of squares (proportional to the variance of the data):

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2,$$

R-squared is monotone in number of features

- As add more features, the R-squared measure cannot decrease. Why?

$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}. \qquad SS_{\text{res}} = \sum_i (y_i - f_i)^2 \qquad SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2,$$

- Is this an issue?

- Alternative: adjusted R-squared — penalize the number of explanatory variables (features)

Variance explained

- Variance explained = variance of predictor to mean of y
- R-squared = Variance explained / Total variance
- FractionVarianceUnexplained(predictor)
  - = MSE[predictor] / Var[y]
  - = 1 - R-squared

Percentage error

- If use error || val1 - val2 ||, and get 0.1, is this good?
- One option: mean percentage error (issues?)
- Another option: mean absolute percentage error (MAPE)

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|,$$

- Another option: symmetric MAPE

$$\mathrm{SMAPE} = \frac{1}{n}\sum_{t=1}^{n}\frac{|F_t - A_t|}{(|A_t| + |F_t|)/2}$$

Classification terminology

- True positives — samples predicted by classifier to be positive that have true label positive
- False positives — samples predicted by classifier to be positive that have true label negative
- True negatives — samples predicted by classifier to be negative that have true label negative
- False negatives — samples predicted by classifier to be negative that have true label positive

Classification measures

Name	Symbol	Definition
Classification error	$error$	$error = \frac{fp+fn}{tp+fp+tn+fn}$
Classification accuracy	$accuracy$	$accuracy = 1 - error$
True positive rate	tpr	$tpr = \frac{tp}{tp+fn}$
False negative rate	fnr	$fnr = \frac{fn}{tp+fn}$
True negative rate	tnr	$tnr = \frac{tn}{tn+fp}$
False positive rate	fpr	$fpr = \frac{fp}{tn+fp}$
Precision	pr	$pr = \frac{tp}{tp+fp}$
Recall	rc	$rc = \frac{tp}{tp+fn}$

Why these specific values?

- These measures exist for multiple reasons
- Separate the importance of false positives and false negatives
  - In some cases, much more hazardous to have a false positive than a false negative (or vice versa) e.g.
- Avoid issues with imbalanced datasets

Confusion Matrix for binary classification

		True class	
		0	1
Predicted class	0	N_{00} 🙂	N_{01} ☹
	1	N_{10} ☹	N_{11} 🙂

Number of data points whose true class was 0 but predicted class was 1.

$$Accuracy = \frac{N_{00} + N_{11}}{N_{00} + N_{10} + N_{01} + N_{11}}$$

$$Error = 1 - Accuracy$$

Classification accuracy and error

$$Accuracy = \frac{N_{correct}}{N}$$

$N_{correct}$: number of correctly classified data points

N: total number of data points

$$Error = 1 - \frac{N_{correct}}{N}$$

Another naming convention:

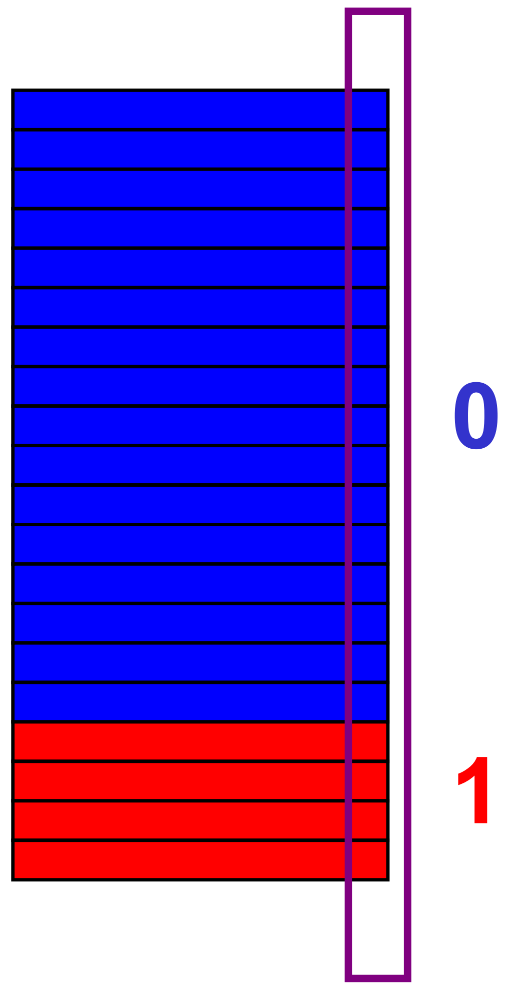
N_{00} = number of true negatives

N_{01} = number of false negatives

N_{10} = number of false positives

N_{11} = number of true positives

Example of importance of measures: imbalanced datasets



16 data points have class 0 (majority class)

4 data points have class 1 (minority class)

Trivial classifier: always predict majority class

Accuracy of a trivial classifier is: 16/20 = 80%

Random classifier: predict class 0 with probability 0.8 and class 1 with probability 0.2

Accuracy of the random classifier: 68%

($0.8^2 + 0.2^2 = 0.68$)

More on accuracy

$$sn = \frac{N_{11}}{N_{01} + N_{11}} \longleftarrow$$

Sensitivity or accuracy on data points whose class is 1. Also called true positive rate.

$$sp = \frac{N_{00}}{N_{10} + N_{00}} \longleftarrow$$

Specificity or accuracy on data points whose class is 0. Also called true negative rate.

$$1 - sn = 1 - \frac{N_{11}}{N_{01} + N_{11}} \longleftarrow$$

False negative rate.

$$1 - sp = 1 - \frac{N_{00}}{N_{10} + N_{00}} \longleftarrow$$

False positive rate.

$$\boxed{Accuracy_B = \frac{sn + sp}{2}} \longleftarrow$$

Balanced-sample accuracy

F-measure

$$F_\beta = \frac{(1+\beta^2)\cdot \text{true positive}}{(1+\beta^2)\cdot \text{true positive} + \beta^2 \cdot \text{false negative} + \text{false positive}}.$$

- Related measure, called F1 score has beta=1
  - best value at 1, worst at 0
- Weighted values of true positives and false negatives
- Focus on positives can be switched to negatives just by considering negatives as positive

More on accuracy

$$rc = \frac{N_{11}}{N_{01} + N_{11}} \quad \longleftarrow$$

Recall is accuracy on data points whose class is 1. Same as sensitivity or true positive rate.

$$pr = \frac{N_{11}}{N_{10} + N_{11}} \quad \longleftarrow$$

Precision is accuracy on data points that were predicted as 1. Also called positive predictive value.

$$1 - rc = 1 - \frac{N_{11}}{N_{01} + N_{11}} \quad \longleftarrow$$

False negative rate.

$$1 - pr = 1 - \frac{N_{11}}{N_{10} + N_{11}} \quad \longleftarrow$$

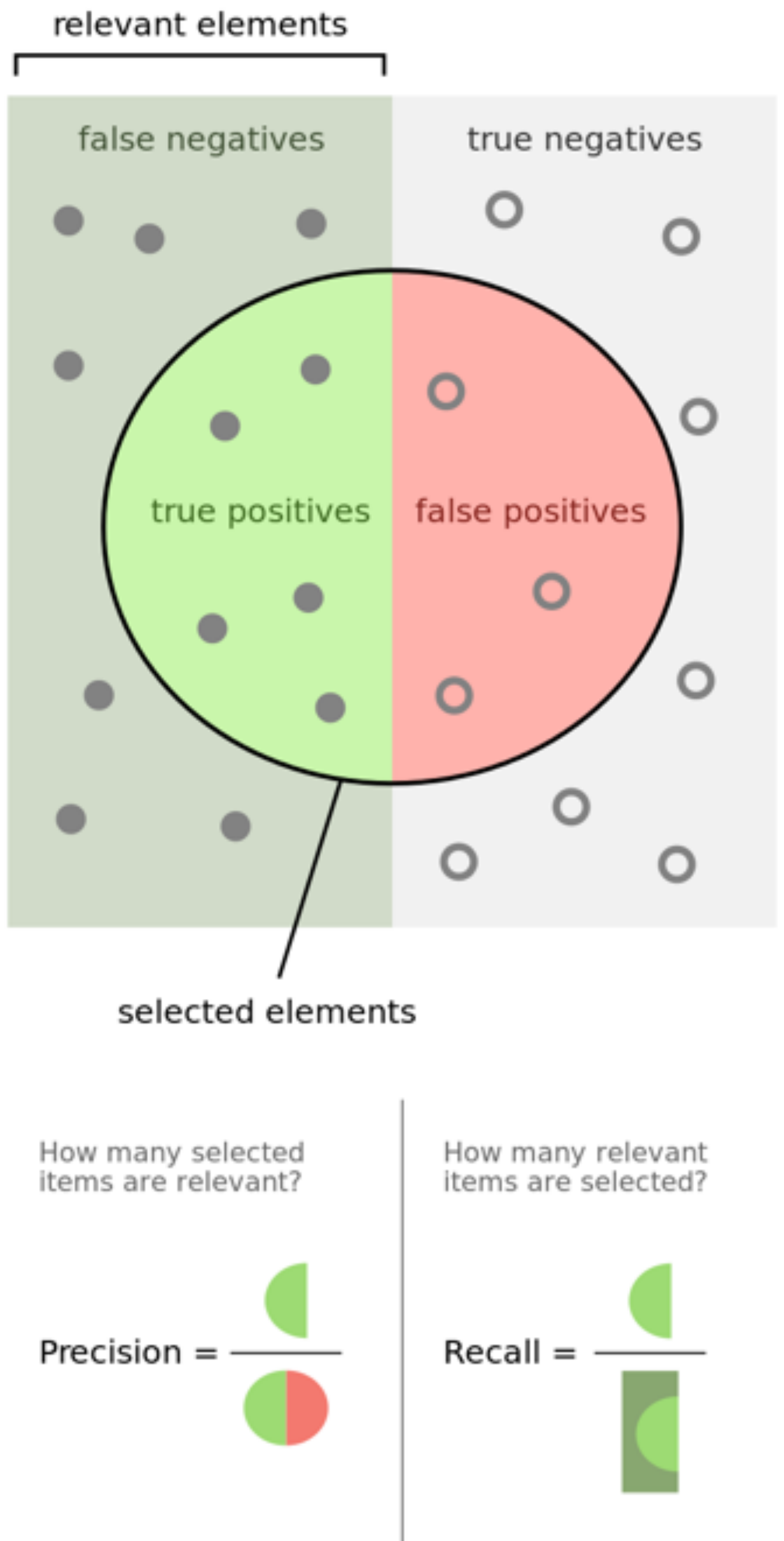
False discovery rate. Important: it is very different from the false positive rate!!!

$$\boxed{F_\beta = (1 + \beta^2) \cdot \frac{pr \cdot rc}{\beta^2 \cdot pr + rc}} \quad \longleftarrow$$

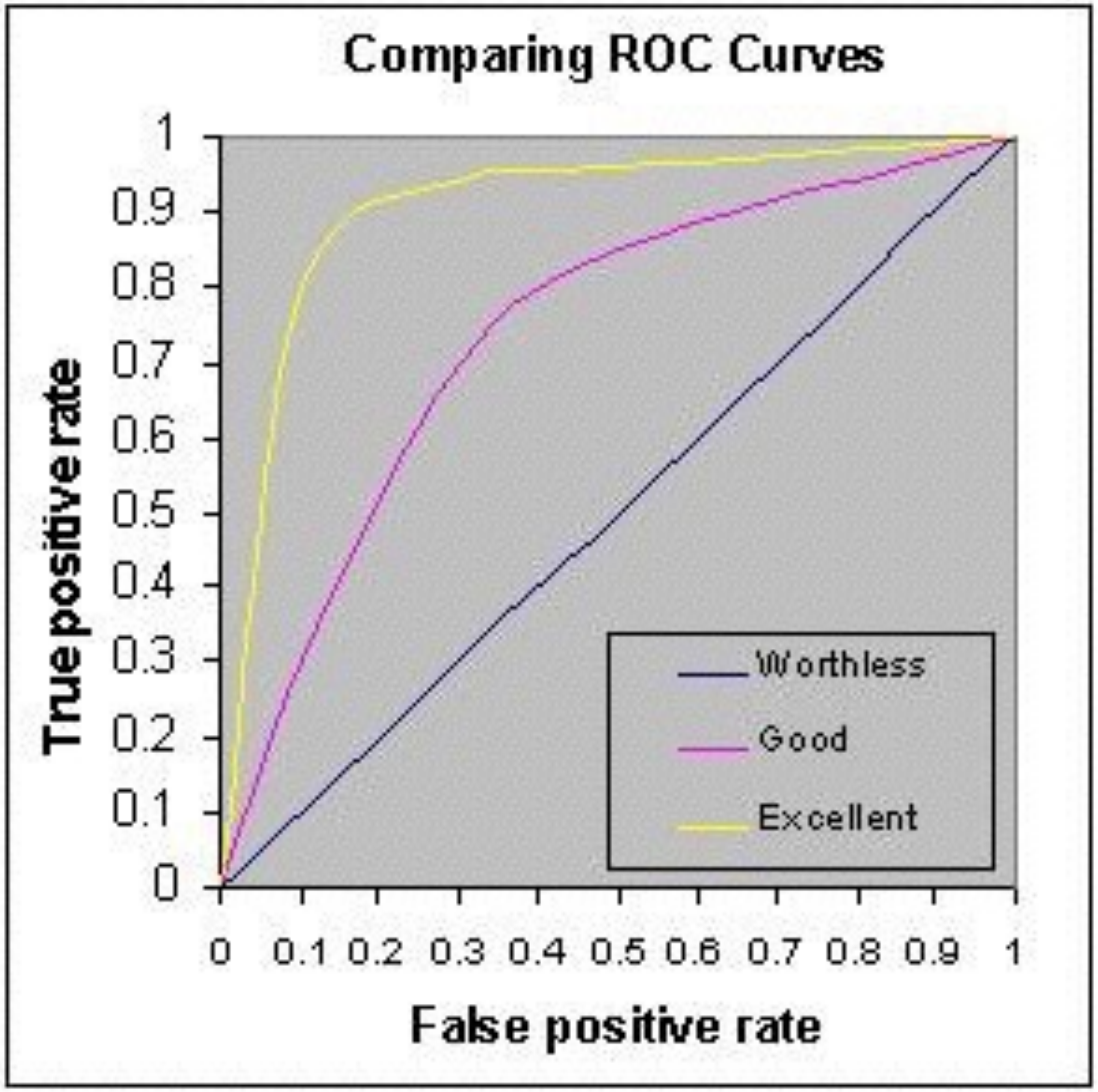
F-measure.

Precision and recall

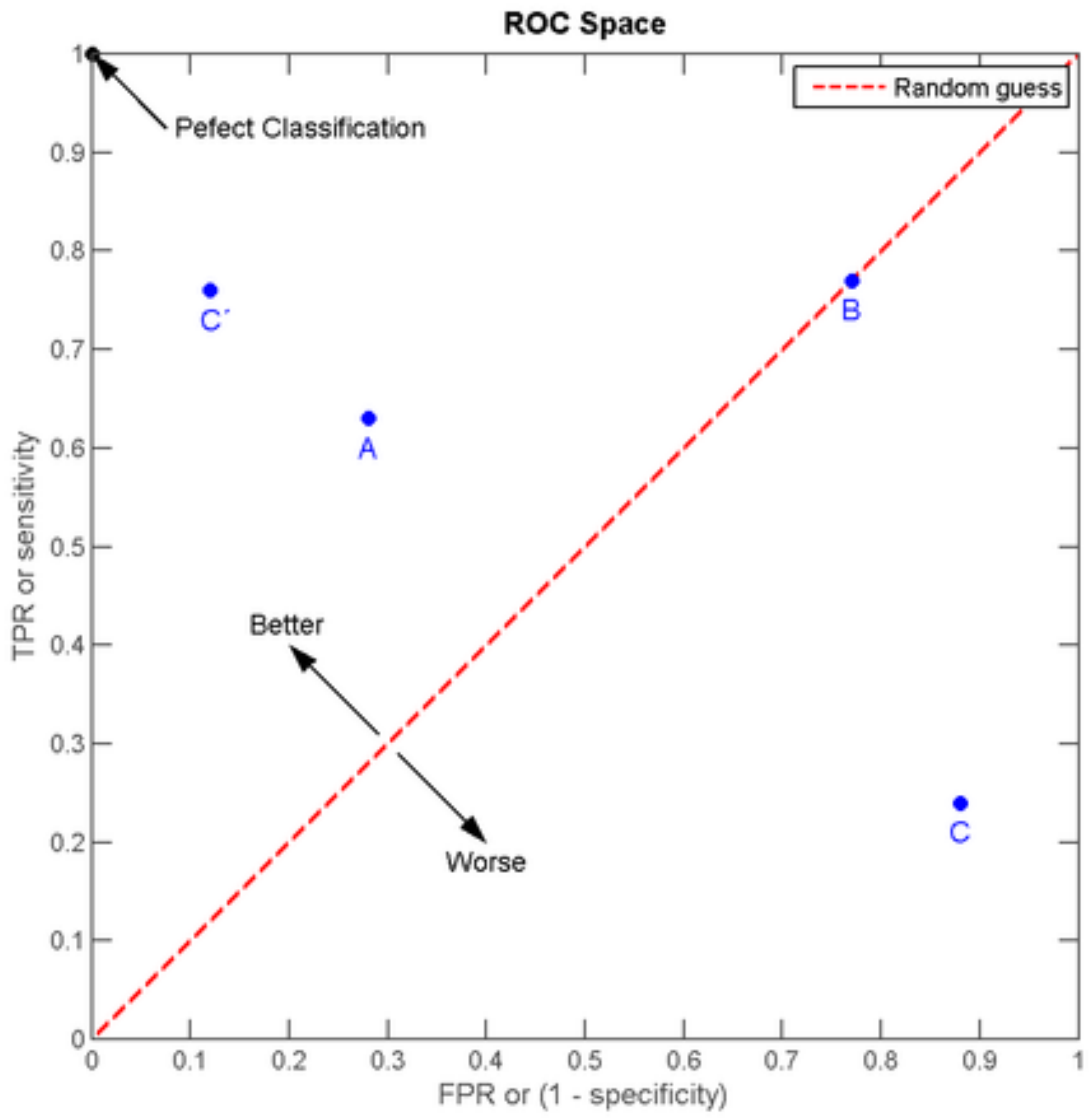
- Example: when a search engine returns 30 pages only 20 of which were relevant while failing to return 40 additional relevant pages, its precision is 20/30 = 2/3 while its recall is 20/60 = 1/3.



ROC Curve

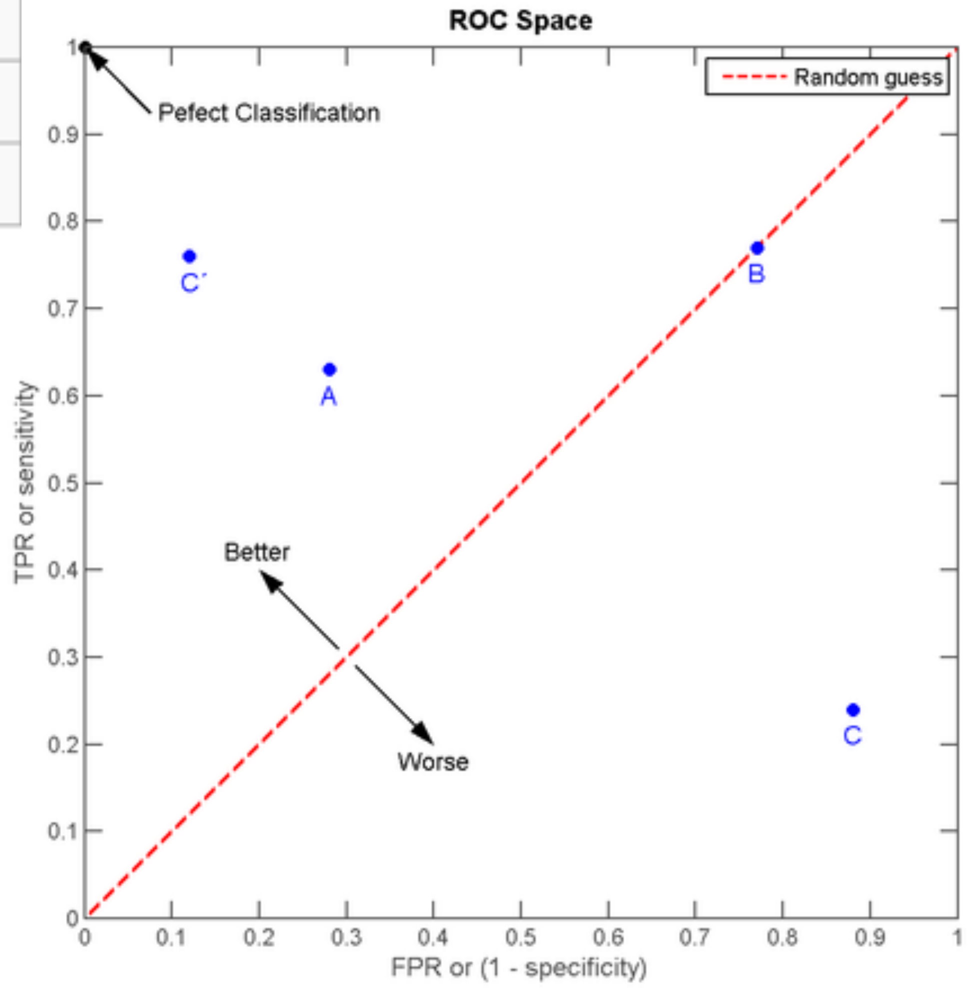

Comparing ROC Curves
True positive rate
False positive rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Worthless
Good
Excellent

ROC space


ROC Space
Pefect Classification
Random guess
C'
B
A
Better
Worse
C
TPR or sensitivity
FPR or (1 - specificity)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

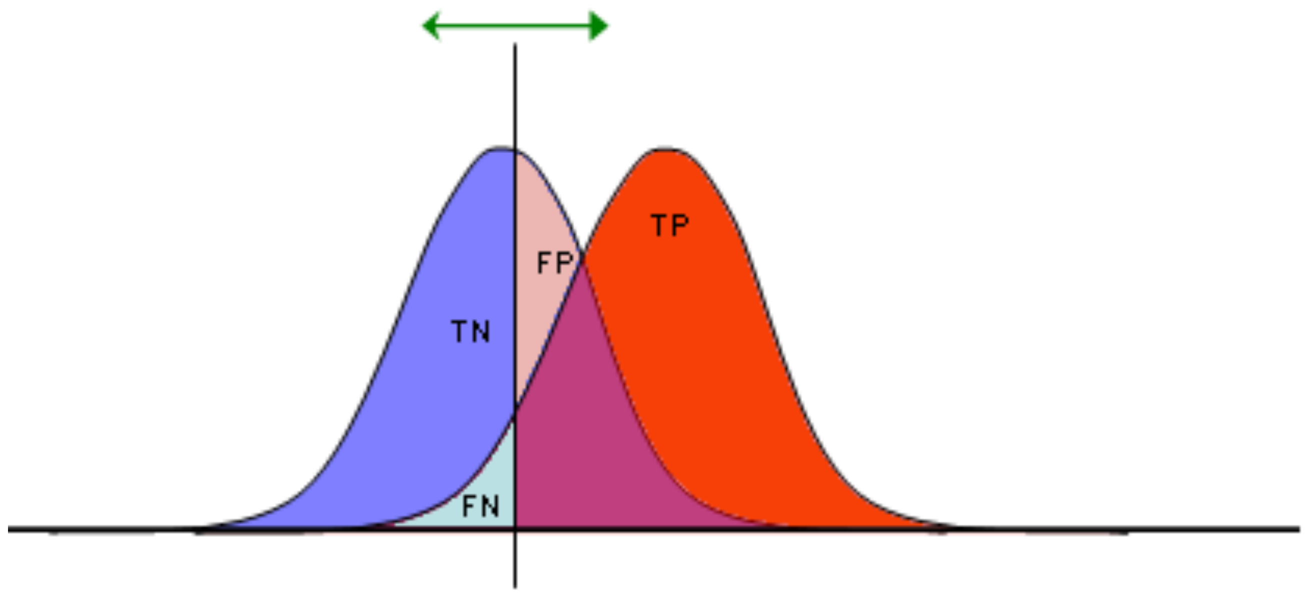
ROC space

A			B			C		
TP=63	FP=28	91	TP=77	FP=77	154	TP=24	FP=88	112
FN=37	TN=72	109	FN=23	TN=23	46	FN=76	TN=12	88
100	100	200	100	100	200	100	100	200
TPR = 0.63			TPR = 0.77			TPR = 0.24		
FPR = 0.28			FPR = 0.77			FPR = 0.88		
PPV = 0.69			PPV = 0.50			PPV = 0.21		
F1 = 0.66			F1 = 0.61			F1 = 0.22		
ACC = 0.68			ACC = 0.50			ACC = 0.18		

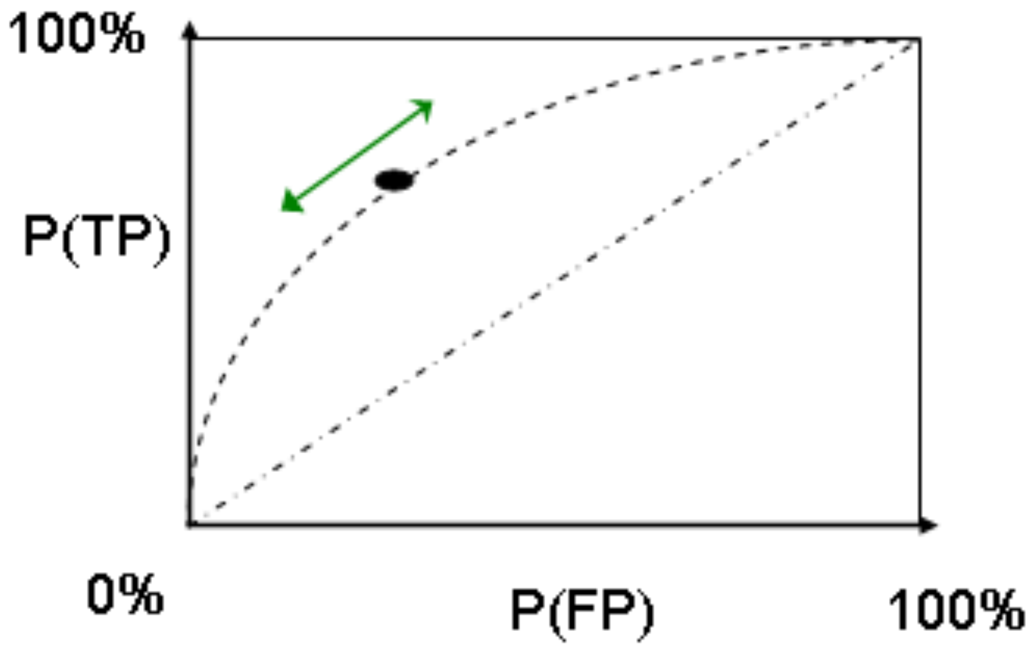


ROC Curve example

e.g., diseased people, healthy people
blood protein levels normally distributed
Parameter that changes: threshold



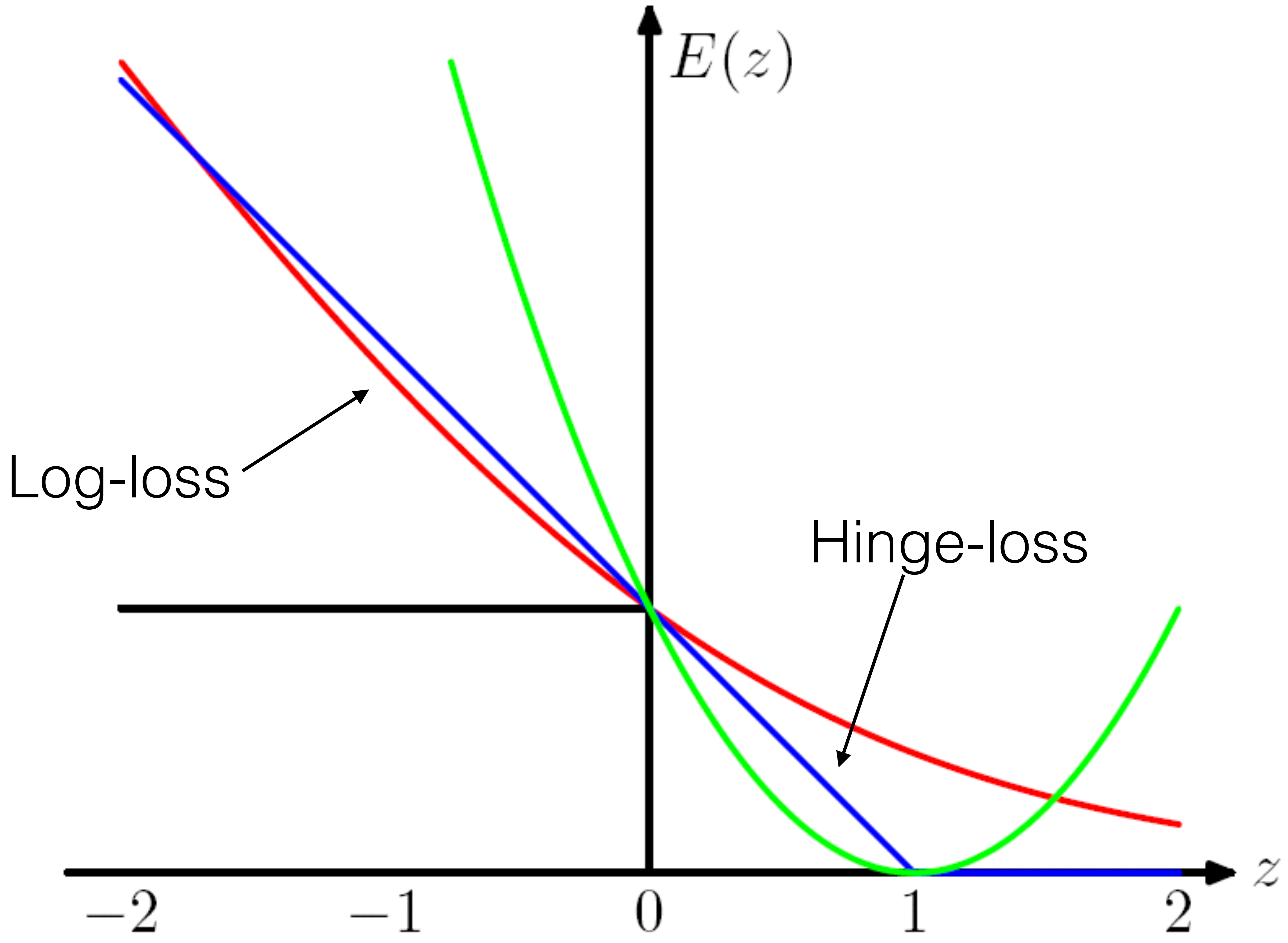
TP	FP
FN	TN
1	1



Area under the curve

- AUC or AUCROC gives the area under the ROC curve
- AUC is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one
- Some issues in using AUC to compare classifiers (see “Measuring classifier performance: a coherent alternative to the area under the ROC curve”, Hand, JMLR, 2009)
    - can give unequal important to a FPR or TPR for different classifiers

Classification objectives



Clipped objectives

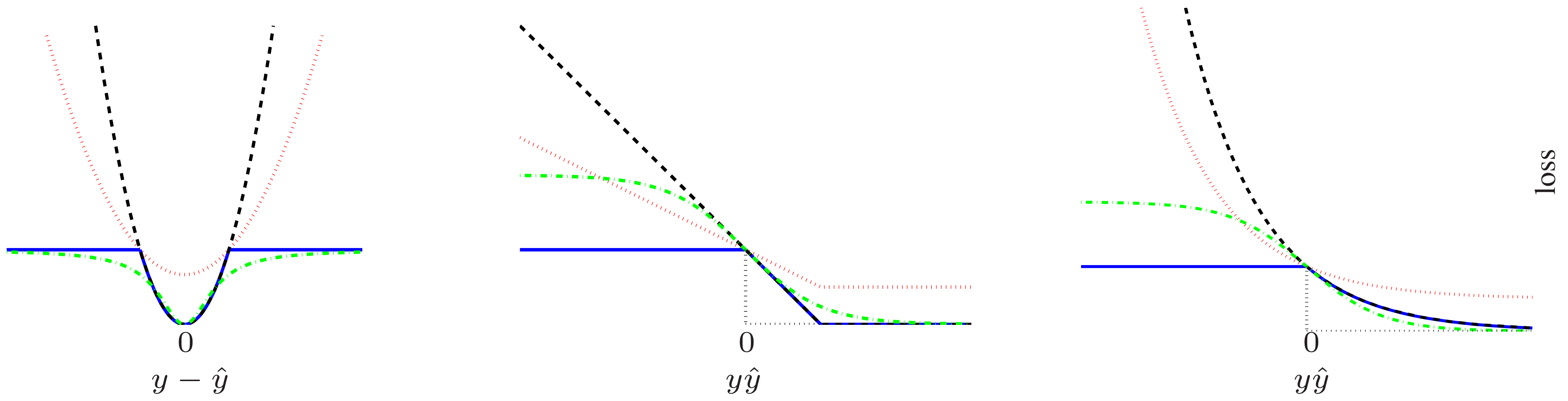


Figure 1: Comparing standard losses (dashed) with corresponding "clipped" losses (solid), ρ-relaxed losses (dotted), and non-convex robust losses (dash-dotted). **Left**: squared loss (dashed), clipped (solid), $1/3$-relaxed (dotted), robust Geman and McClure loss [2] (dash-dotted). **Center**: SVM hinge loss (dashed), clipped [27, 30] (solid), $1/2$-relaxed (upper dotted), robust $1 - \tanh(y\hat{y})$ loss [19] (dash-dotted). **Right**: Adaboost exponential loss (dashed), clipped (solid), $1/2$-relaxed (upper dotted), robust $1 - \tanh(y\hat{y})$ loss [19] (dash-dotted).

see: "Relaxed Clipping: A Global Training Method for Robust Regression and Classification", Yu et al, 2010

Miscellaneous

- Understand the expected properties of your estimator
- Example: give NN the same training data, in the same order, with the same starting point —> should it produce the same final set of weights?
- Example: with enough samples, should I get close to the true parameters? for any small random subset, how different might my learned parameters be?