IS 733 Lesson 8

Evaluation of Supervised Learning

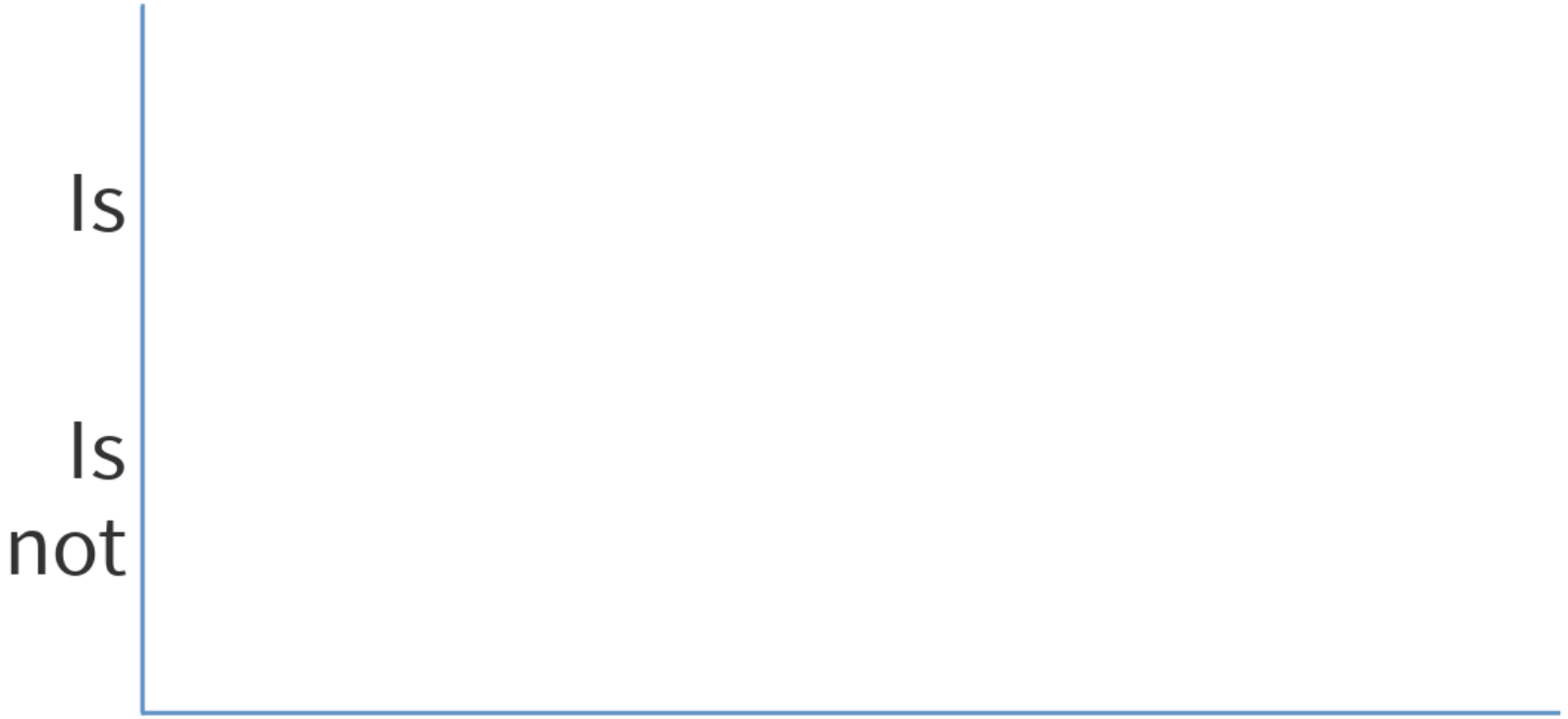
Slides based on those from Data Mining by I. H. Witten, E. Frank, M. A. Hall and C. J. Pal

Announcements

- Reminder: Homework 3 will be due on Friday midnight, 4/2/2021 (extended)

- You can submit it on Blackboard, under “Assignments.”

Performance on the training set _____ a good indicator of performance on an independent test set.



Which is NOT a benefit of cross-validation?

The computational cost on large datasets is low compared to the holdout method

In each fold, the training data is not used to test the model's performance

Helps ensure reliable estimates of a classifier's accuracy when there is limited data for training and testing

Every data point gets used to measure accuracy in a test set

The Receiver Operating Characteristic (ROC) curve is useful to measure classification performance when the classes are imbalanced, e.g. class 1 is much more frequent than class 2.

True

False

Learning outcomes

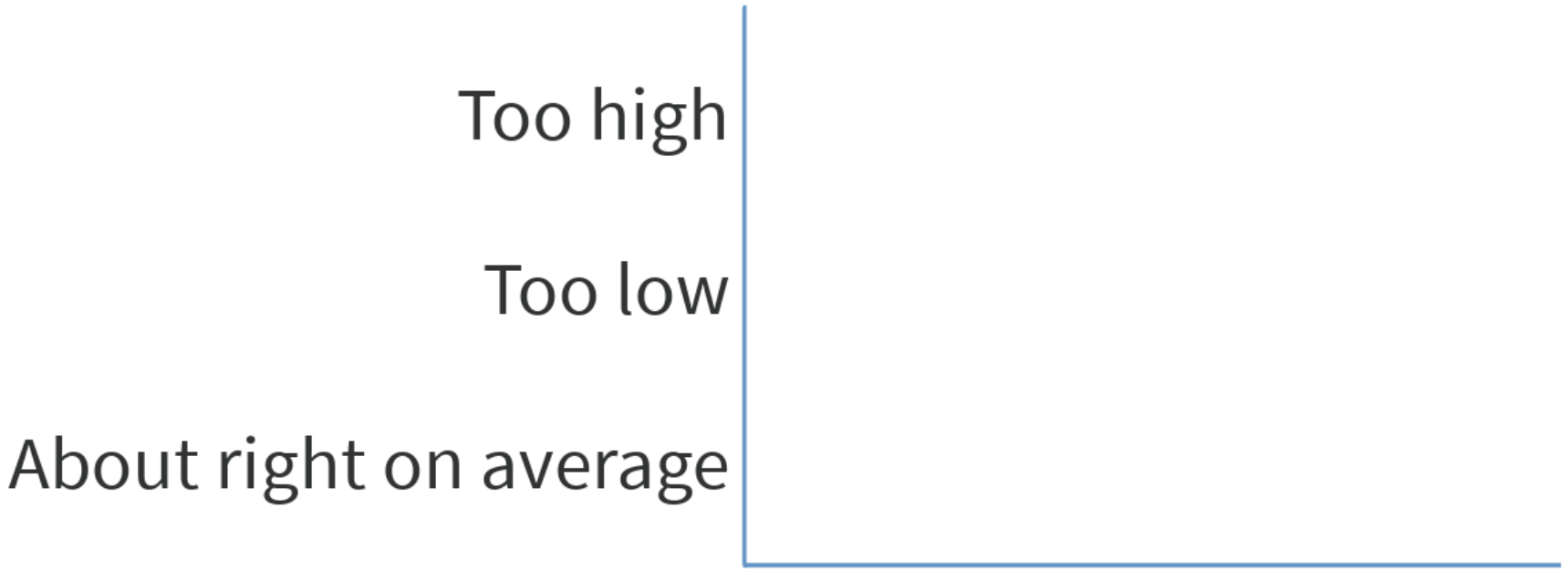
By the end of the lesson, you should be able to:

- **Explain** why classification accuracy on the training set may not be a good indicator of a classifier's performance
- **Describe** the steps of several evaluation methodologies for classifiers
- **Outline** valid strategies for hyperparameter selection which avoid "peeking" at the test data
- **Select** appropriate experimental methodologies when evaluating machine learning methods

Training and testing

- How predictive is the model we have learned?
- Natural performance measure for classification problems: ***error rate***
  - *Success*: instance's class is predicted correctly
  - *Error*: instance's class is predicted incorrectly
  - **Classification accuracy**: Percent of the whole set of instances that the classifier got right
  - **Error rate**: proportion of errors made over the whole set of instances

Suppose we estimate the error rate for a classifier on the same data that it was trained on. Our estimate of the error rate is likely to be _____.



Training and testing

- ***Resubstitution error (a.k.a. training error)****:*
  error rate obtained by evaluating model on training data

- This measures the model's performance on data that it got to see ahead of time.

- When it is deployed, it won't get to see the data that it has to predict on!

Evaluation: the key to success

- Error on the training data is ***not*** necessarily a good indicator of performance on future data
  - Otherwise 1-NN would be the optimal classifier!
  - Our model is potentially fitting to noisy, spurious patterns that occur in the training set, possibly by chance.
  - There is no guarantee that the error rate on the training data will correspond to its error rate "in the wild"

Resubstitution error is (hopelessly) optimistic!

The error rate estimated on the training set (resubstitution error) ____ .

gives an unfair advantage to more flexible models	**A**
gives an unfair advantage to more inflexible models	**B**
is similarly inaccurate regardless of whether a model is relatively flexible	**C**

Training and testing

- ***Test set***: independent instances that have played no part in formation of classifier
  - Assumption: both training data and test data are representative samples of the underlying problem

- Be careful to avoid a situation where test and training data could differ in nature
  - Example: classifiers built using customer data from two different towns *A* and *B*
  - To estimate performance of classifier from town *A* in completely new town, test it on data from *B*

Train/Test Split

- Given dataset X
- For each of K trials
  - Randomly divide X into training set (2/3) and testing set (1/3)
  - Learn classifier on training set
  - Test classifier on testing set (compute error)
- Compute average error over K trials
- Problem
  - Training and testing sets overlap between trials
  - Biases the results

K-fold Cross Validation

- Given dataset X
- Partition X into K disjoint sets X_1, …, X_K
- For i = 1 to K
  - Learn classifier on training set $X - X_i$
  - Test classifier on testing set X_i (compute error)
- Compute average error over K trials
- Testing sets no longer overlap
- Training sets still overlap

Cross-Validation

- Stratification
  - Distribution of classes in training and testing sets should be the same as in original dataset
  - Called “stratified cross validation”
- Leave-one-out cross validation
  - $K = N = |X|$
  - Used when classified data is scarce
    - Medical diagnosis

5x2 Cross-Validation

- Tom Dietterich, 1998
- For each of 5 trials (shuffling X each time)
  - Divide X **randomly** in two halves X_1 and X_2
  - Compute error using X_1 as training and X_2 as testing
  - Compute error using X_2 as training and X_1 as testing
- Compute average error of all 10 results
- 5 trials best number to minimize overlap among training and testing sets

Bootstrapping

- If not enough data for k-fold cross validation
- Generate multiple samples of size N from X by sampling with replacement
- Each sample has approximately 63% of the examples in X
- Compute average error over all samples

Bootstrapping

- Draw instances from a dataset *with replacement*
- Prob that we do not pick an instance after N draws $\left(1-\frac{1}{N}\right)^{N} \approx e^{-1} = 0.368$

that is, only 36.8% is new!

Measuring Classifier Performance

- Confusion matrix

	Predicted class		
True class	**Positive**	**Negative**	Total
Positive	tp: true positive	fn: false negative	p
Negative	fp: false positive	tn: true negative	n
Total	p’	n’	N

Performance Measures (2-class)

Name	Formula
error accuracy	(fp + fn)/N (tp + tn)/N
tp-rate fp-rate	tp/p fp/n
precision recall	tp/p' tp/p = tp_rate
sensitivity specificity	tp/p = tp_rate tn/n = 1 – fp_rate

F-measure:

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Predicting performance

- Assume the estimated error rate is 25%. How close is this to the true error rate?
  - Depends on the amount of test data

- Prediction is just like tossing a (biased!) coin
  - “Head” is a “success”, “tail” is an “error”
- In statistics, a succession of independent events like this is called a *Bernoulli process*
  - Statistical theory provides us with confidence intervals for the true underlying proportion

Holdout estimation

- What should we do if we only have a single dataset?

- The ***holdout* method** reserves a certain amount for testing and uses the remainder for training, after shuffling

- How much to hold out as a test set?
  - **More test data makes the estimated error rate more accurate**. But it means less training data!
  - **This will make the classifier less accurate!**

  - Typical choice: one third for testing, the rest for training
  - For big data, the test set can be a much smaller percentage

Repeated holdout method

- Holdout estimate can be made more reliable by repeating the process with different subsamples
  - In each iteration, a certain proportion is randomly selected for training (possibly with stratificiation)
  - The error rates on the different iterations are averaged to yield an overall error rate

- This is called the ***repeated holdout*** method

Making the most of the data

- Generally, the larger the training data the better the classifier (but returns diminish)

- The larger the test data the more accurate the error estimate

Dilemma: ideally both training set *and* test set should be large!

Cross-validation

- *K-fold cross-validation* avoids overlapping test sets
  - First step: split data into *k* subsets of equal size
  - Second step: use each subset in turn for testing, the remainder for training
  - This means the learning algorithm is applied to *k* different training sets

Test Train

Cross-validation

- *K-fold cross-validation* avoids overlapping test sets
  - First step: split data into *k* subsets of equal size
  - Second step: use each subset in turn for testing, the remainder for training
  - This means the learning algorithm is applied to *k* different training sets

Test Train

Cross-validation

- *K-fold cross-validation* avoids overlapping test sets
  - First step: split data into *k* subsets of equal size
  - Second step: use each subset in turn for testing, the remainder for training
  - This means the learning algorithm is applied to *k* different training sets

Cross-validation

- The error estimates are averaged to yield an overall error estimate
  - Standard deviation of the error rate can be computed

- Optionally, the subsets are stratified before the cross-validation is performed to yield **stratified *k*-fold cross-validation**

Hyperparameter selection

- ***Hyperparameter****:* parameter that can be tuned by hand to optimize the performance of a learning algorithm
  - Different from basic parameter that is part of a model, such as a coefficient in a linear regression model
  - E.g. *k* in the *k*-nearest neighbor classifier
  - whether to use Laplace smoothing for naïve Bayes
  - Regularization parameters to prevent overfitting…

Suppose we train our classifier on the training set, and evaluate it on the test set, and its error rate is not good. We change its hyperparameters and get a much better error rate on the test set. Is the new test set error rate a reasonable estimate of the classifier's performance "in the wild"?

Hyperparameter selection

- We are **not allowed to use the final test data** to choose the value of hyperparameters
  - Adjusting the hyperparameter to the test data will lead to *optimistic* performance estimates on this test data! This is "**peeking**" at the test set
  - Parameter tuning needs to be viewed as part of the learning algorithm and must be done using the **training data only**

Hyperparameter selection

- How to get a useful estimate of performance for different parameter values so that we can choose a value?
  - Answer: split the data into a smaller "training" set and a **validation set (a.k.a. development set)**
  - Build models using different values of *hyperparameters* on the new, smaller training set and **evaluate them on the validation set**
  - Pick the best value of *hyperparameter* and **rebuild** the model on the **full original training set**

Test Validation Train

Hyperparameters and cross-validation

- What to do when the training sets are very small, so that performance estimates on a validation set are unreliable?

- We can use ***nested cross-validation*** (expensive!)
  - For each training set of the "outer" *k*-fold cross-validation, **run "inner" *p*-fold cross-validations** to choose the best hyperparameter value
  - **Inner cross-validations are used to choose hyperparameter values**
  - Outer cross-validation is used to estimate quality of learning process
  - Inner cross-validations are part of the learning process!

Making the most of your data

- After comparing models and selecting hyperparameters using the hold-out method or CV, pick the best model/hyper-parameters

- You can then **retrain your model with all the data** (including the validation set) for deployment
  - This way you get to use all the data for the final model!

Counting the cost

- In practice, different types of classification errors often incur different costs
- Examples:
  - Terrorist profiling: “Not a terrorist” correct 99.99…% of the time
  - Loan decisions
  - Oil-slick detection
  - Fault diagnosis
  - Promotional mailing

Counting the cost

- The ***confusion matrix***:

		Predicted class	
		Yes	No
Actual class	Yes	True positive	False negative
	No	False positive	True negative

- Different misclassification costs can be assigned to **false positives** and **false negatives**

Classification with costs

- Two cost matrices:

(A)		Predicted Class	
		Yes	*No*
Actual class	*Yes*	0	1
	No	1	0

(B)		Predicted Class		
		a	*b*	*c*
Actual class	*a*	0	1	1
	b	1	0	1
	c	1	1	0

- In cost-sensitive evaluation of classification methods, success rate is replaced by average cost per prediction
  - Cost is given by appropriate entry in the cost matrix

Cost-sensitive classification

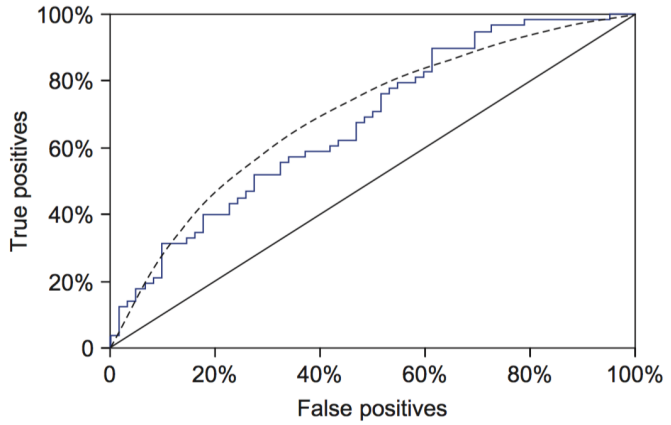
- Can take costs into account when making predictions
  - Basic idea: only predict high-cost class when very confident about prediction
- Given: predicted class probabilities
  - Normally, we just predict the most likely class
  - Here, we should make the prediction that minimizes the expected cost
    - Expected cost: dot product of vector of class probabilities and appropriate column in cost matrix
    - Choose column (class) that minimizes expected cost
- This is the minimum-expected cost approach to cost-sensitive classification

Cost-sensitive learning

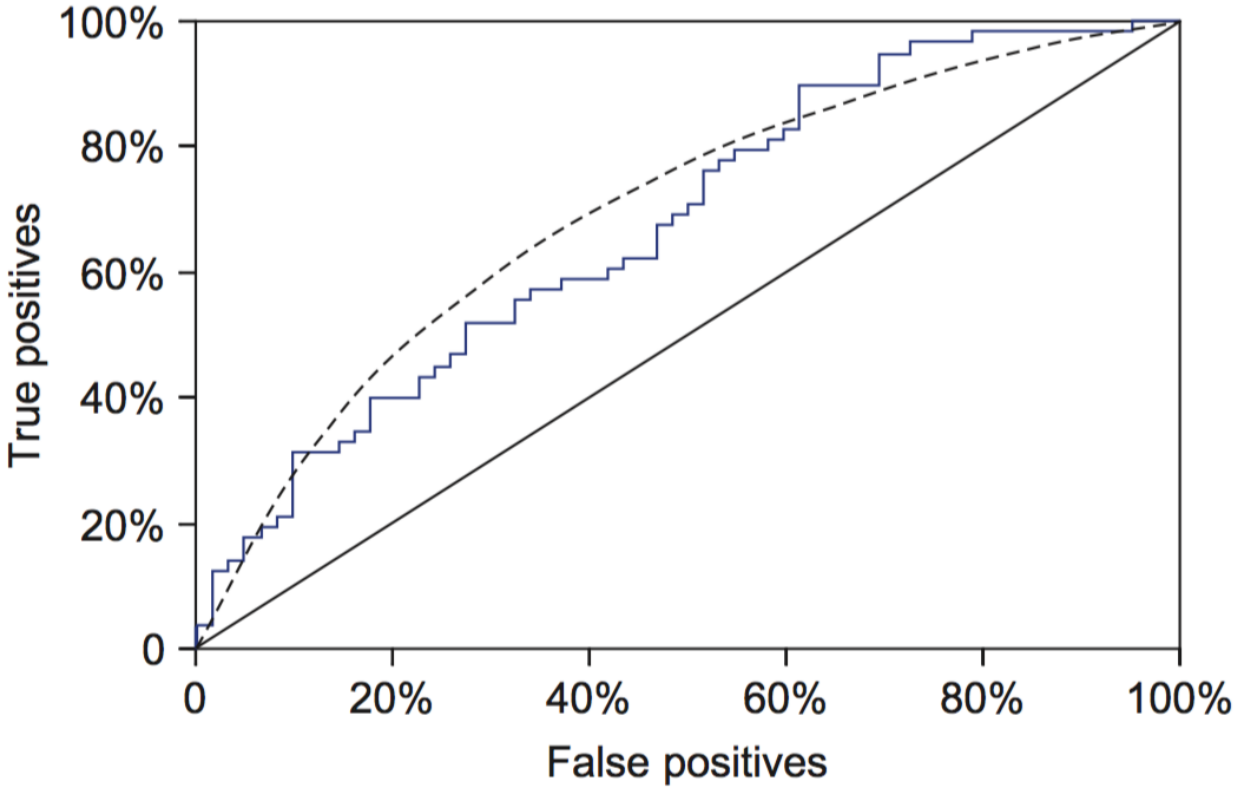
- So far we haven't taken costs into account at training time
- Most learning schemes do not perform cost-sensitive learning
  - They generate the same classifier no matter what costs are assigned to the different classes
  - Example: standard decision tree learner
- Simple methods for cost-sensitive learning:
  - Resampling of instances according to costs
  - Weighting of instances according to costs
- Some schemes can take costs into account by varying a parameter, e.g., naïve Bayes

ROC curves

- *ROC curves*
  - Stands for “receiver operating characteristic”
  - Originated in signal detection to show tradeoff between **hit rate** and **false alarm rate** over noisy channel
    - Shows behaviour of classifier as we shift a threshold (e.g. probability of class 1) to classify an instance as positive
    - E.g. predicting oil slicks, can vary detection threshold for oil slick, to show more or less candidates to a human analyst

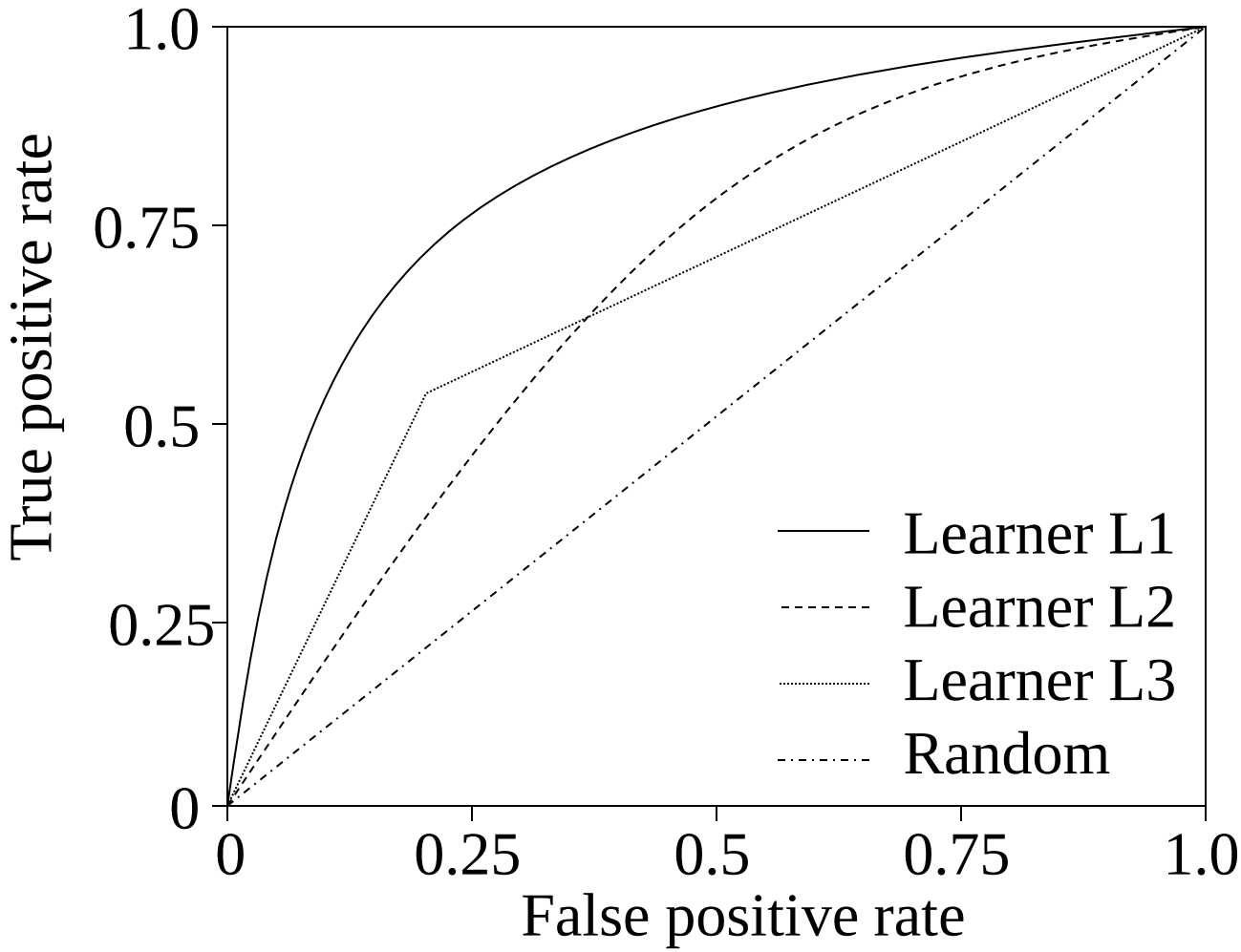


A sample ROC curve



- *x* axis: **percentage of false positives** in sample
- *y* axis: **percentage of true positives** in sample
- Jagged curve—one set of test data
- Smoother curve—use cross-validation

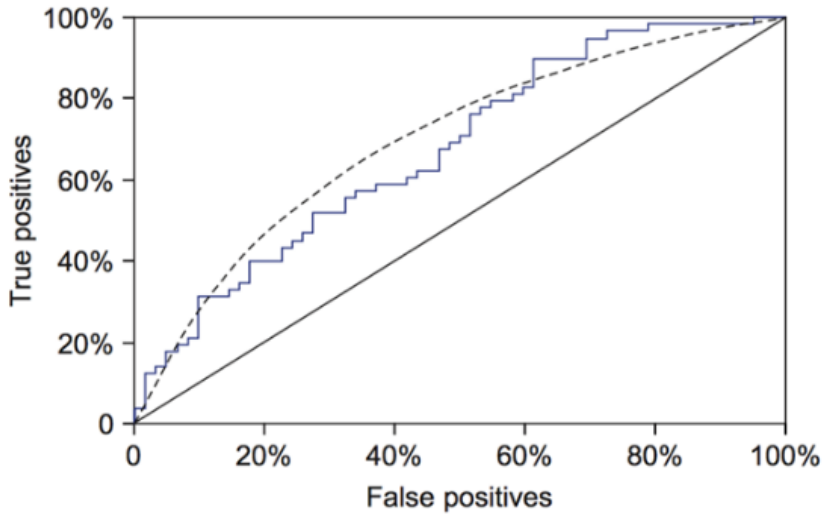
ROC Curve


1.0
0.75
0.5
0.25
0
True positive rate
0
0.25
0.5
0.75
1.0
False positive rate
Learner L1
Learner L2
Learner L3
Random

Domination in ROC Space

- Learner L1 dominates L2 if L1's ROC curve is always above L2's curve
- If L1 dominates L2, then L1 better than L2 for all possible error costs and class distributions
- If neither dominates (L2 and L3), then different classifiers are better under different conditions

It is better if the ROC curve passes through or near to the:



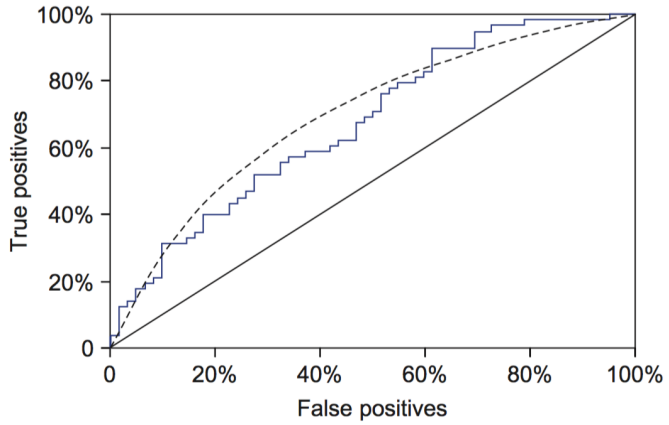
The top-left of the box

The bottom-right of the box

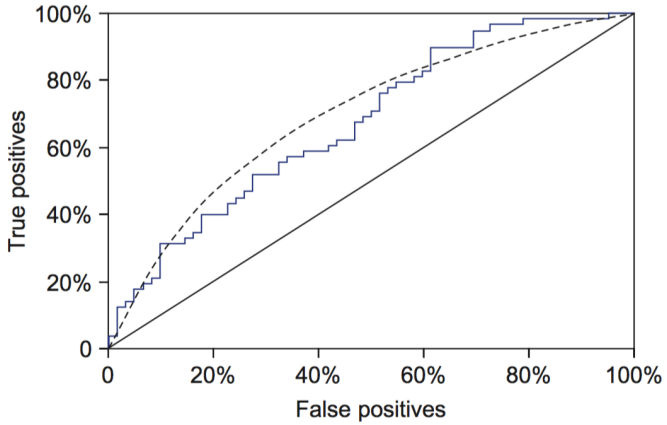
The diagonal line through the middle of the box

Cross-validation and ROC curves

- Simple method of getting a ROC curve using cross-validation:
  - Collect probabilities for instances in test folds
  - Sort instances according to class probabilities
  - Draw the figure from left to right, going up or across depending on whether the sorted instances are positive or negative
- This method is implemented in WEKA

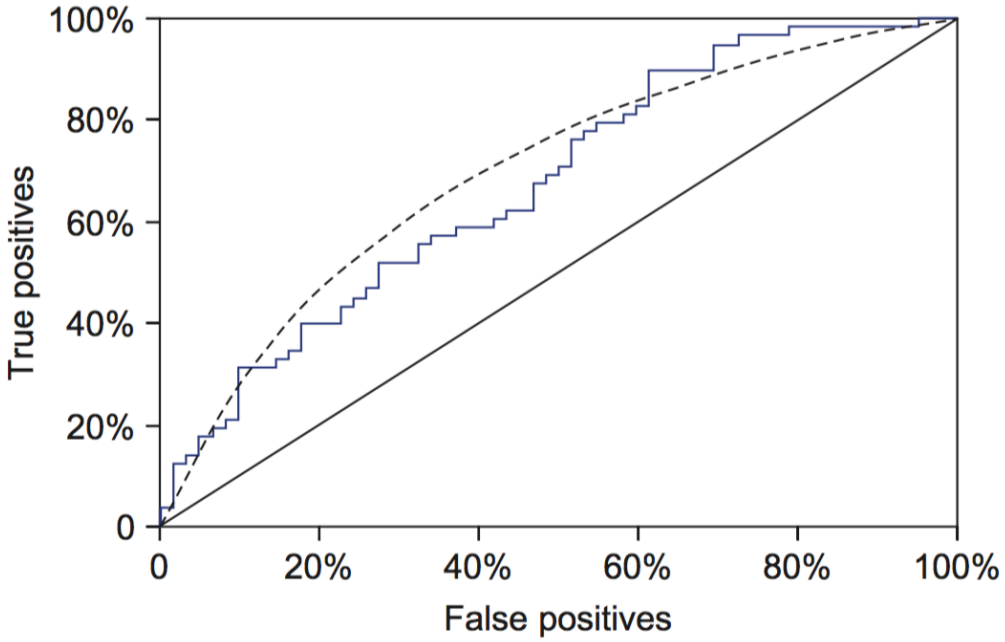


Area under the ROC curve



- The area under the ROC curve (ROC AUC) corresponds to the chance that a random positive instance is ranked higher than a random negative instance. Interpretation: suppose we
  - pick a random positive instance
  - pick a random negative instance
  - Check whether the model gave the positive instance a higher probability of being positive (i.e., got the ranking right)
  - Do this many times, and calculate the probability of a correct ranking by our model. This equals the ROC AUC

Area under the ROC curve



- ROC AUC = 1: perfect classification performance.
- ROC AUC = 0.5: random performance
- ROC AUC < 0.5: worse than random chance – should never happen!

Think-Pair-Share: Evaluation Procedures

- Choose an appropriate **evaluation procedure** (hold-out, repeated hold-out, cross-validation,nested cross-validation…) *and* a **performance metric** (accuracy/error rate, cost-sensitive classification, ROC curve, ROC-AUC) for each of the following tasks:

- Detecting oil slicks, given 1000 images where 1% are positive
- Predicting whether an image belongs to one of 20,000 categories, with 10 million training images
- Predicting whether or not a person will repay a loan, 10,000 people's data where 95% repaid the loan
- Training a support vector machine to predict whether a person will click on an online ad, 80,000 instances where 10% clicked on the ad