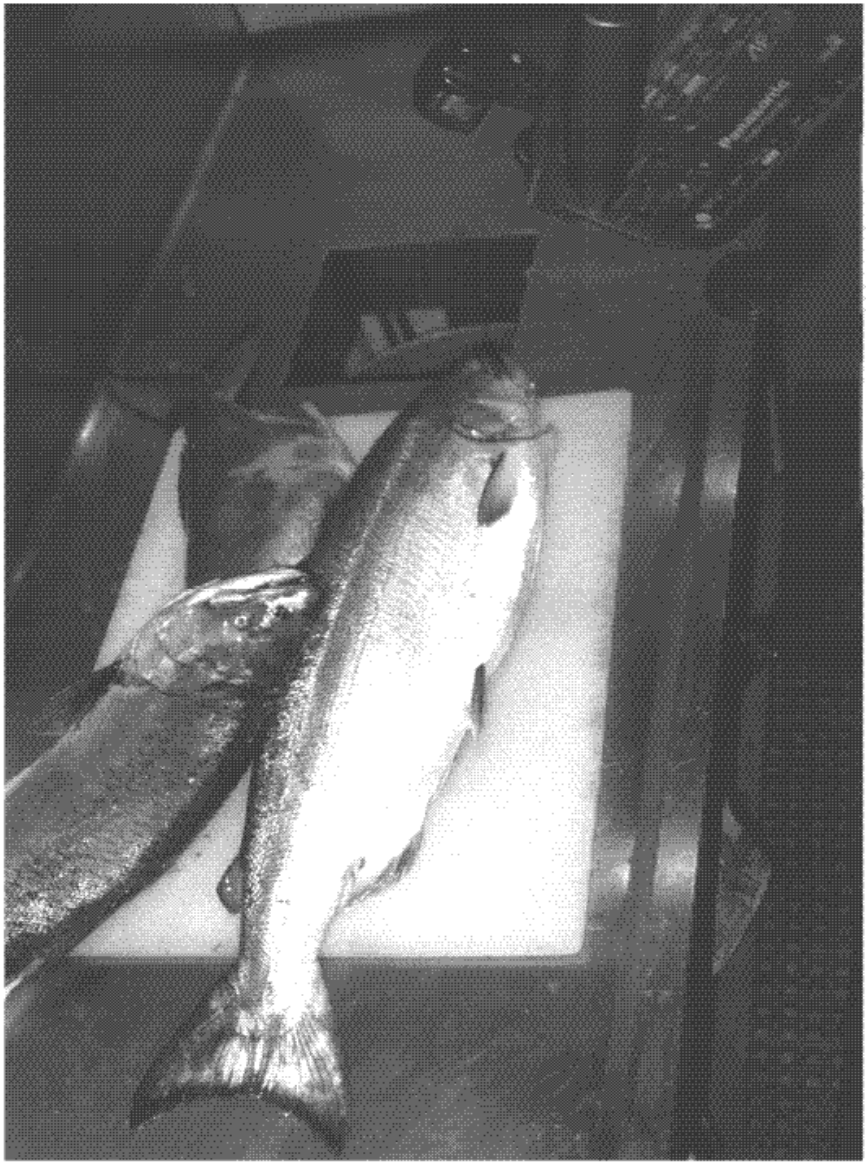
CHAPTER 14:

Assessing and Comparing Classification Algorithms

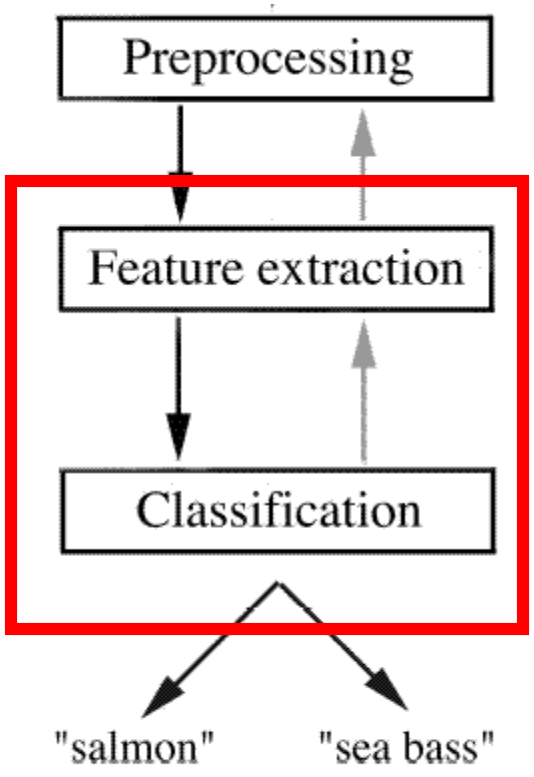
Classification example

- A fish-packing plant wants to automate the process of sorting incoming fish according to species
- As a pilot project, it is decided to try to separate sea bass from salmon using optical sensing



- Length
- Lightness
- Width
- Position of the mouth
- …

Features to explore for use in our classifier



- **Preprocessing**: Images of different fishes are isolated from one another and from background;
- **Feature extraction**: The information of a single fish is then sent to a feature extractor, that measure certain "features" or "properties";
- **Classification**: The values of these features are passed to a classifier that evaluates the evidence presented, and build a model to discriminate between the two species

- **Domain knowledge**:

  a sea bass is generally longer than a salmon

- **Feature**: *Length*

- **Model**:

  Sea bass have some typical length, and this is greater than the length of a salmon

Classification rule (Classification model):

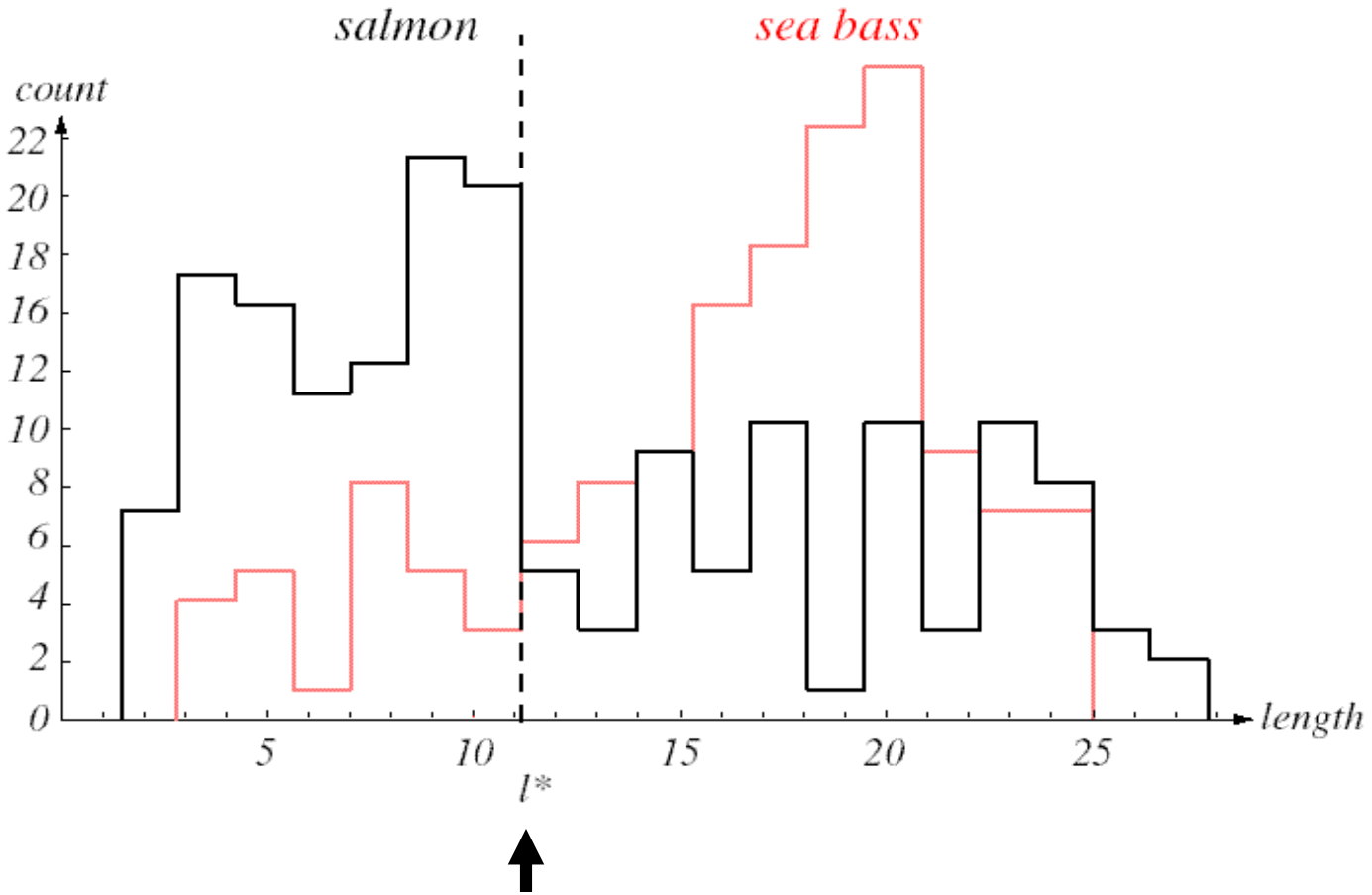
If *Length* >= *l** then *sea bass*

otherwise *salmon*

How to choose *l** ? – **parameter that we have to learn from the data!**

- Use length values of sample data (training data)

Histograms of the length feature for the two categories



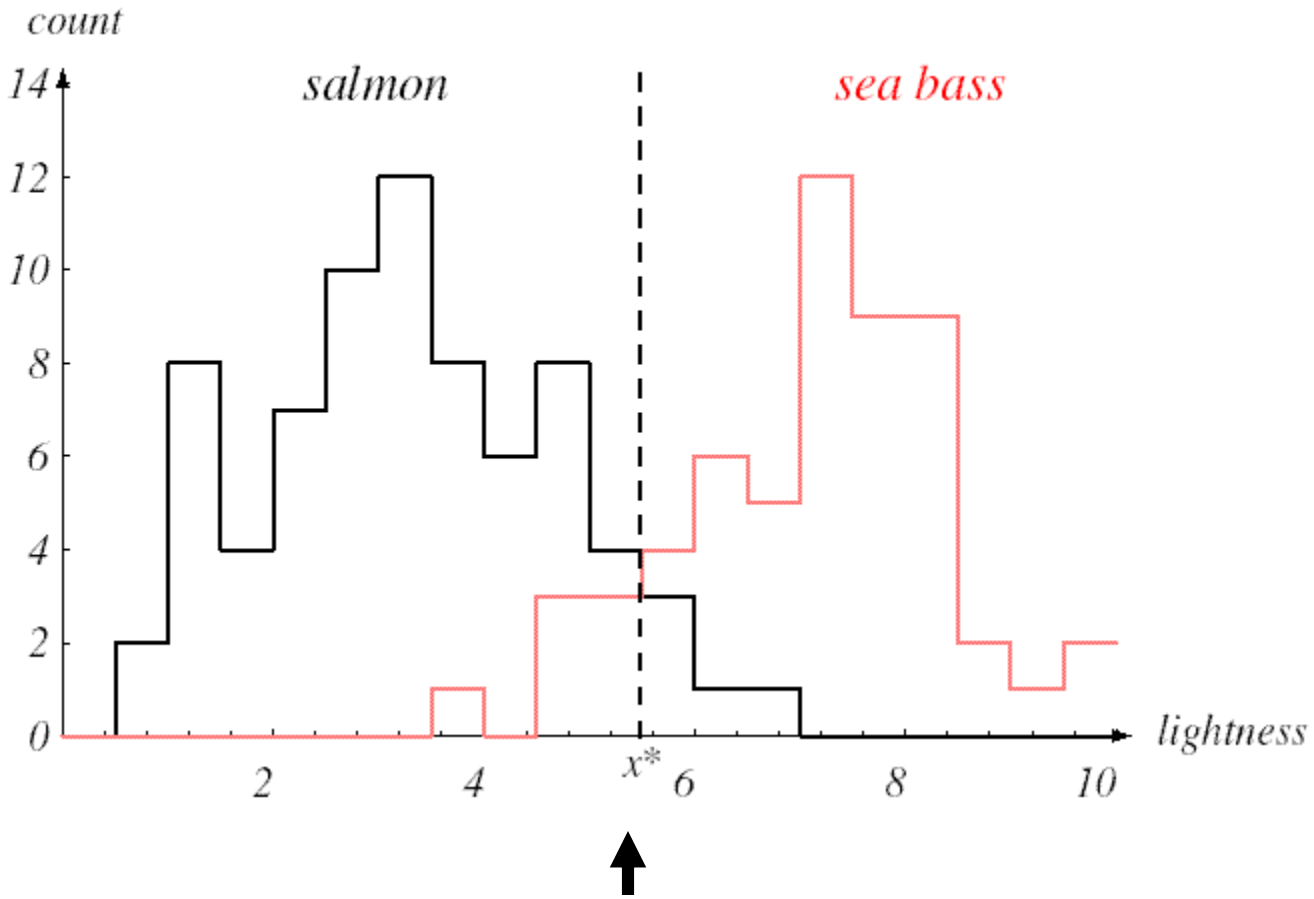
Leads to the smallest number of errors on average

We cannot reliably separate sea bass from salmon by length alone!

- **New Feature*:***

  Average lightness of the fish scales

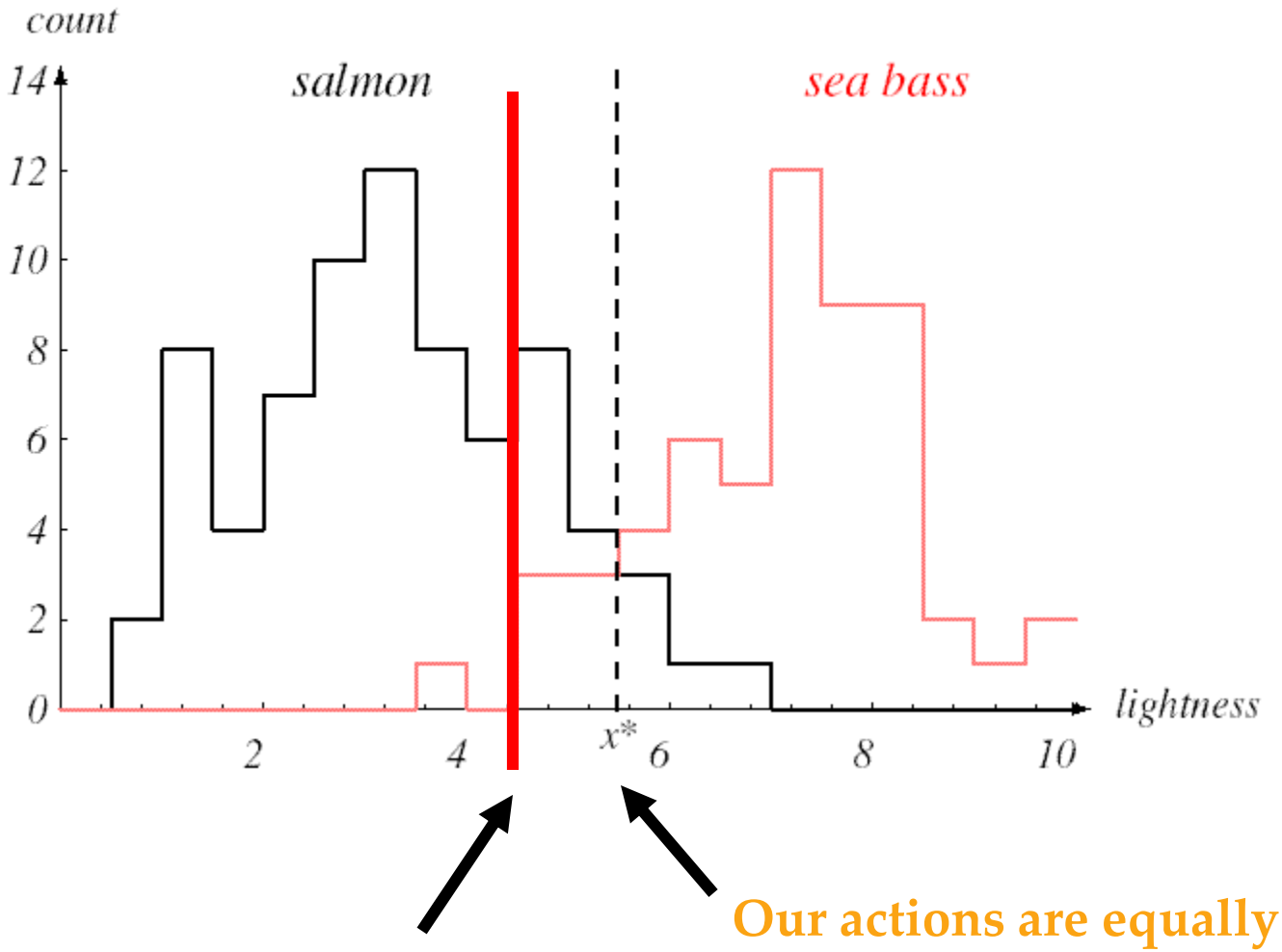
Histograms of the lightness feature for the two categories



Decision boundary: the particular x* leads to the smallest number of errors on average

The two classes are much better separated!

Histograms of the lightness feature for the two categories



Our actions are equally costly

Classifying a sea bass as salmon costs more. Thus we reduce the number of sea bass classified as salmon.

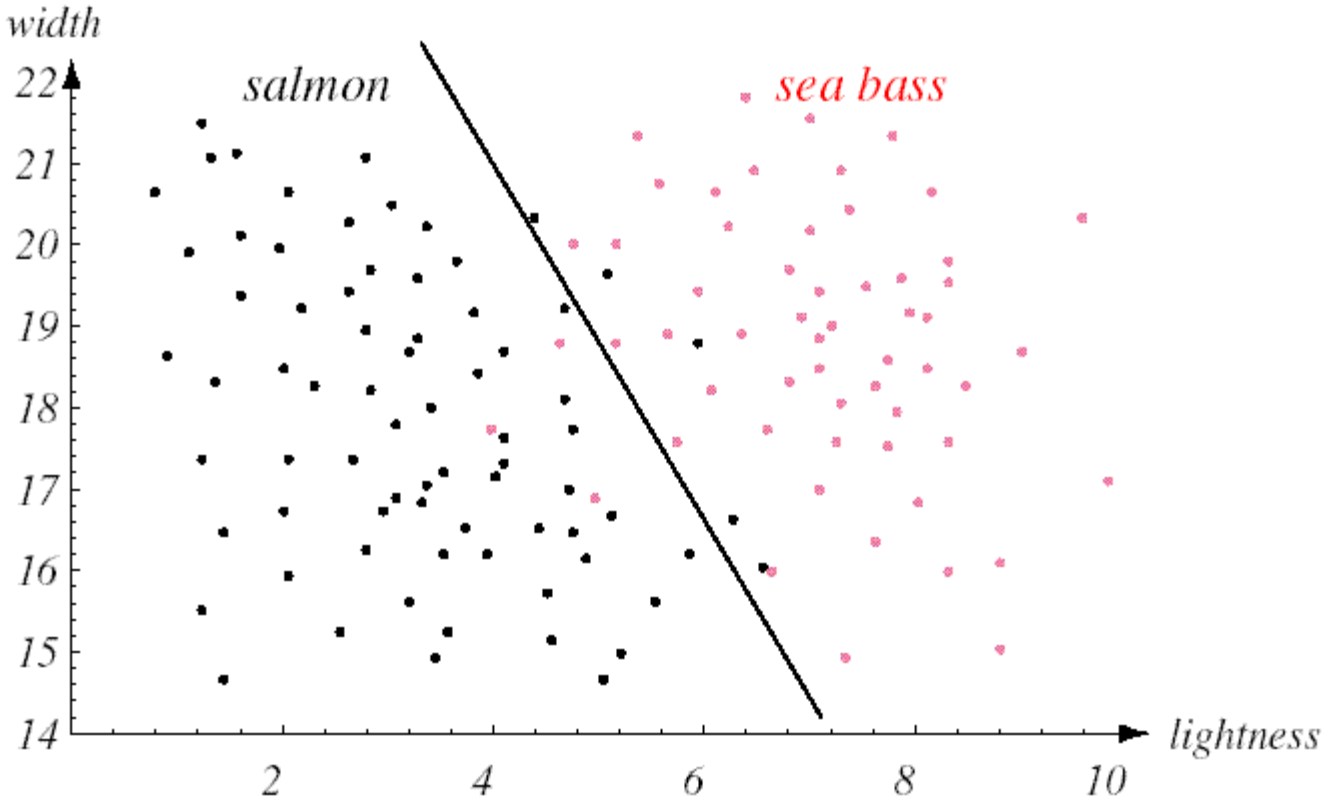
- **In Summary**:

  The overall task of classification is to come up with a decision rule (i.e., a **decision boundary**) so as **to minimize the cost** (which is equal to the average number of mistakes for equally costly actions).

- No single feature gives satisfactory results
- We must rely on using more than one feature
- We observe that:

  sea bass usually are wider than salmon
- Two features: *Lightness and Width*

- Resulting fish representation: $\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

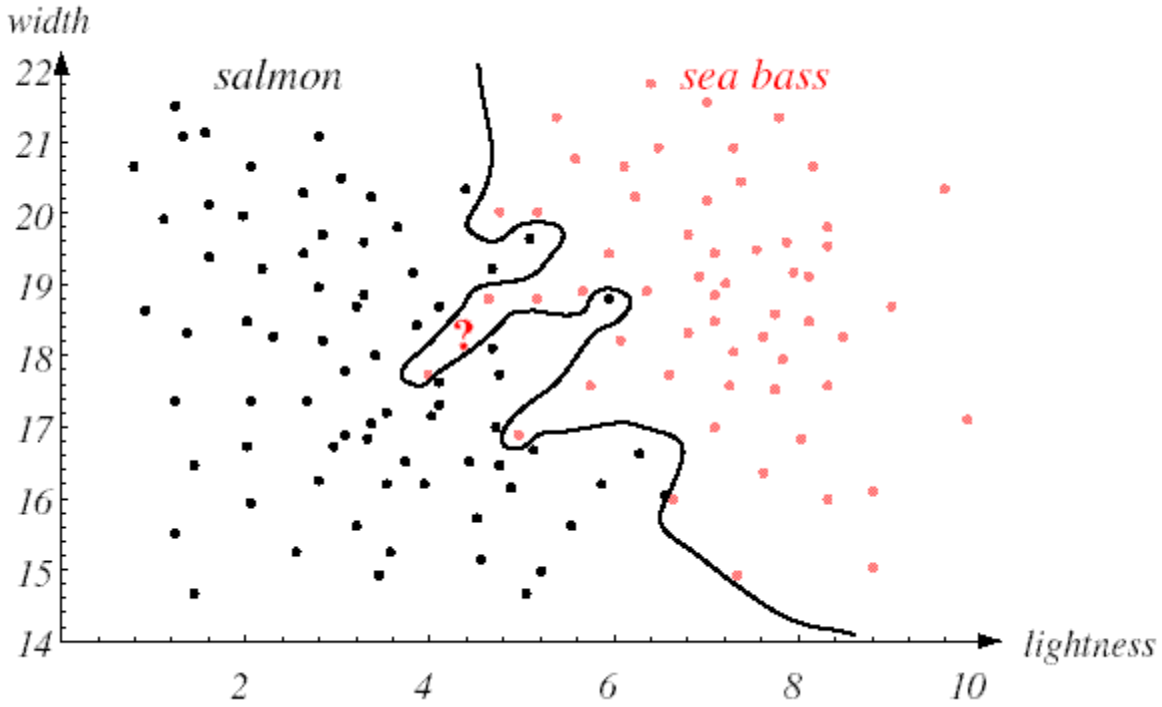


Decision rule: Classify the fish as a sea bass if its feature vector falls above the decision boundary shown, and as salmon otherwise

Should we be satisfied with this result?

Options we have:

- Consider additional features:
  - Which ones?
  - Some features may be redundant (e.g., if *eye color* perfectly correlates with *width*, then we gain no information by adding eye color as feature.)
  - It may be costly to attain more features
  - Too many features may hurt the performance!
- Use a more complex model



All training data are perfectly separated

Should we be satisfied now??

We must consider: Which decisions will the classifier take on novel patterns, i.e. fishes not yet seen? Will the classifier suggest the correct actions?

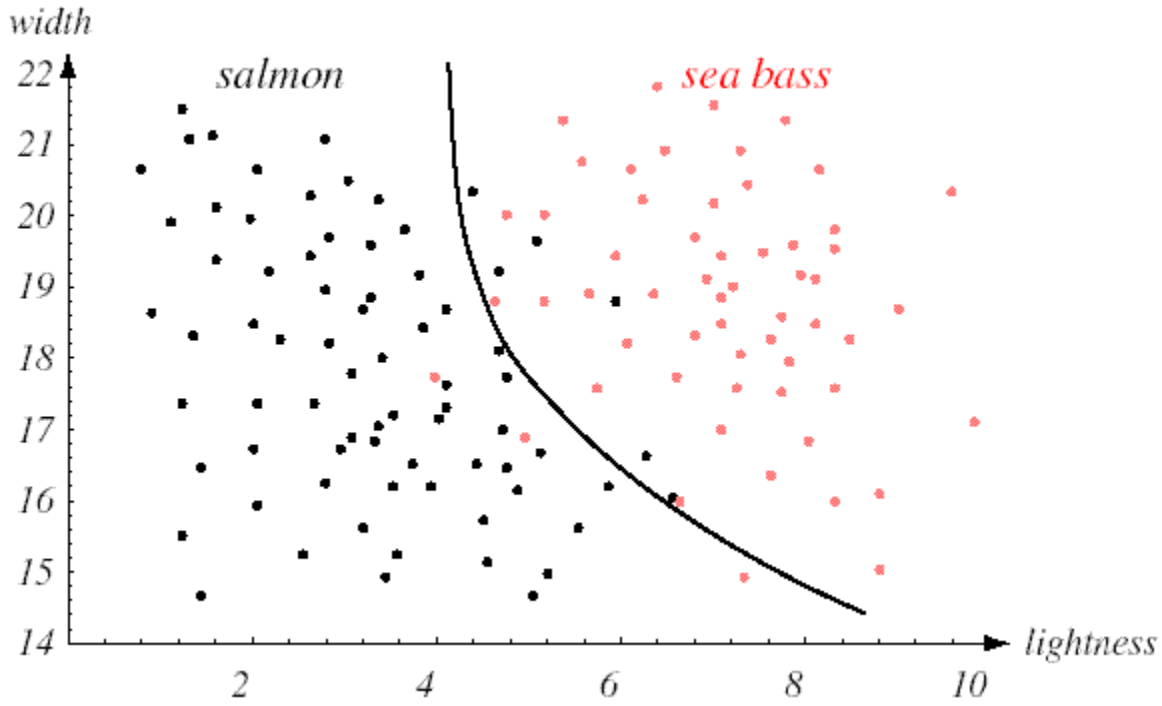
This is the issue of GENERALIZATION

Generalization

- A good classifier should be able to generalize, i.e. perform well on unseen data
- The classifier should capture the underlying characteristics of the categories
- The classifier should NOT be tuned to the specific (accidental) characteristics of the training data
- Training data in practice contain some noise

- **As a consequence:**

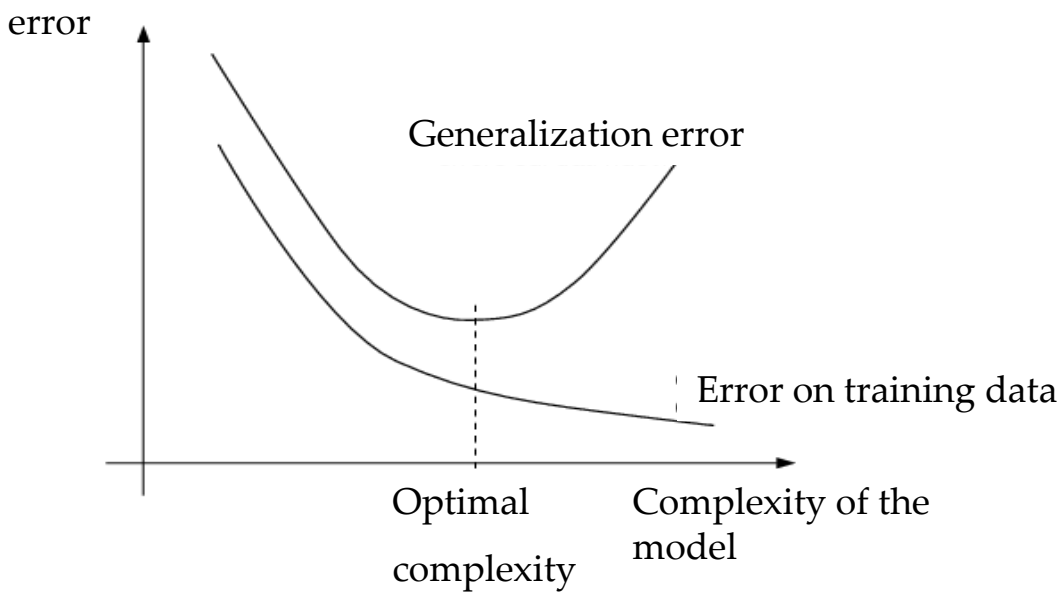
We are better off with a slightly poorer performance on the training examples if this means that our classifier will have better performance on novel patterns.



The decision boundary shown may represent the optimal tradeoff between accuracy on the training set and on new patterns

How can we determine automatically when the optimal tradeoff has been reached?

Tradeoff between performance on training and novel examples



Evaluation of the classifier on novel data is important to avoid *overfitting*

Summarizing our example:

- Classification is the task of recovering (approximating) the model that generated the patterns (generally expressed in terms of probability densities)

- Given a new vector of feature values, the classifier needs to determine the corresponding probability for each of the possible categories

- **Performance evaluation:**
  - Error rate: the percentage of new patterns that are assigned to the wrong class
  - Risk: total expected cost

Assessment of classification algortihms

- Questions:
  - Assessment of the expected error of a learning algorithm: Is the error rate of 1-NN less than 2%?
  - Comparing the expected errors of two algorithms: Is *k*-NN more accurate than MLP ?
- Training/validation/test sets

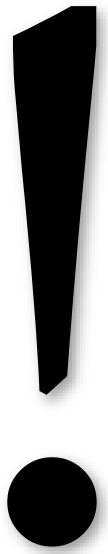
Algorithm Preference

- Criteria (Application-dependent):
  - Misclassification error, or risk (loss functions)
  - Training time/space complexity
  - Testing time/space complexity
  - Interpretability
  - Easy programmability
- Cost-sensitive learning

Measuring Error

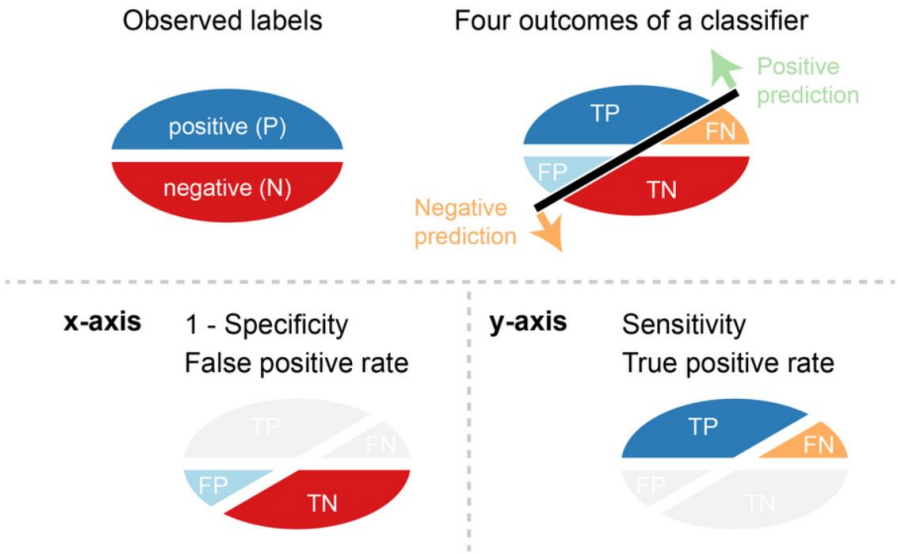
	Predicted class	
True Class	Yes	No
Yes	TP: True Positive	FN: False Negative
No	FP: False Positive	TN: True Negative

- Error rate = # of errors / # of instances = (FN+FP) / N
- Recall = # of found positives / # of positives
  = TP / (TP+FN) = sensitivity = hit rate
- Precision = # of found positives / # of found
  = TP / (TP+FP)
- Specificity = TN / (TN+FP)
- False alarm rate = FP / (FP+TN) = 1 – Specificity
- Multi-class confusion matrix



Measuring Error: ROC curve

The ROC plot is a model-wide evaluation measure that is based on two basic evaluation measures – specificity and sensitivity. Specificity is a performance measure of the whole negative part of a dataset, whereas sensitivity is a performance measure of the whole positive part.



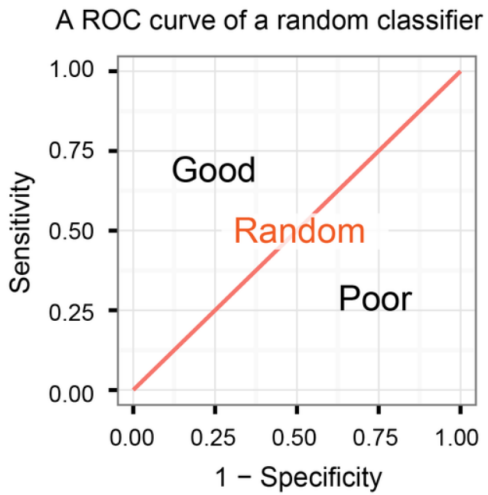
A dataset has two labels (P and N), and a classifier separates the dataset into four outcomes – TP, TN, FP, FN. The ROC plot is based on two basic measures – specificity and sensitivity that are calculated from the four outcomes.

https://classeval.wordpress.com/introduction/introduction-to-the-roc-receiver-operating-characteristics-plot/

Measuring Error: ROC curve

A ROC curve of a random classifier

A classifier with the random performance level always shows a straight line from the origin (0.0, 0.0) to the top right corner (1.0, 1.0). Two areas separated by this ROC curve indicates a simple estimation of the performance level. ROC curves in the area with the top left corner (0.0, 1.0) indicate good performance levels, whereas ROC curves in the other area with the bottom right corner (1.0, 0.0) indicate poor performance levels.



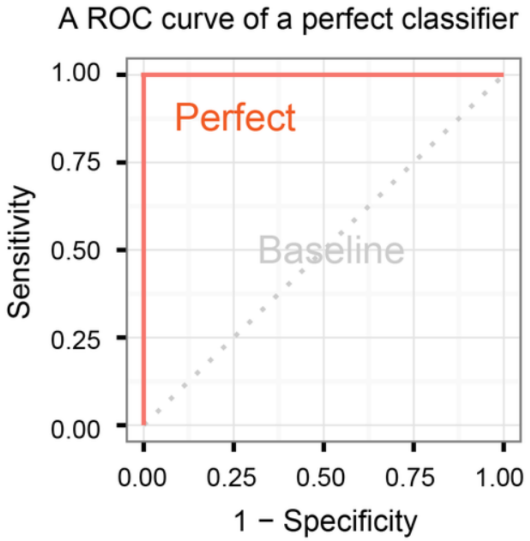
A ROC curve represents a classifier with the random performance level. The curve separates the space into two areas for good and poor performance levels.

https://classeval.wordpress.com/introduction/introduction-to-the-roc-receiver-operating-characteristics-plot/

Measuring Error: ROC curve

A ROC curve of a perfect classifier

A classifier with the perfect performance level shows a combination of two straight lines – from the origin (0.0, 0.0) to the top left corner (0.0, 1.0) and further to the top right corner (1.0, 1.0).



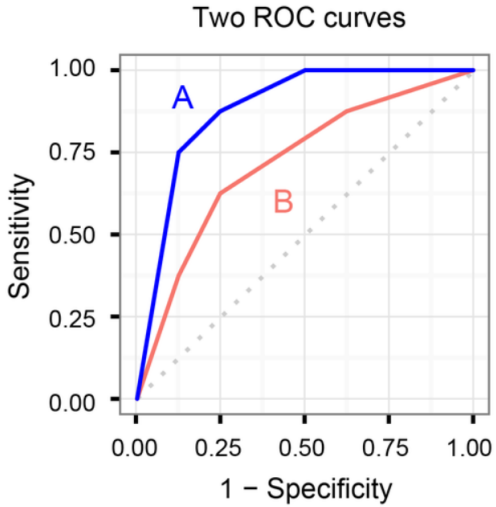
A ROC curve represents a classifier with the perfect performance level.

https://classeval.wordpress.com/introduction/introduction-to-the-roc-receiver-operating-characteristics-plot/

Measuring Error: ROC curve

ROC curves for multiple models

Comparison of multiple classifiers is usually straight-forward especially when no curves cross each other. Curves close to the perfect ROC curve have a better performance level than the ones closes to the baseline.



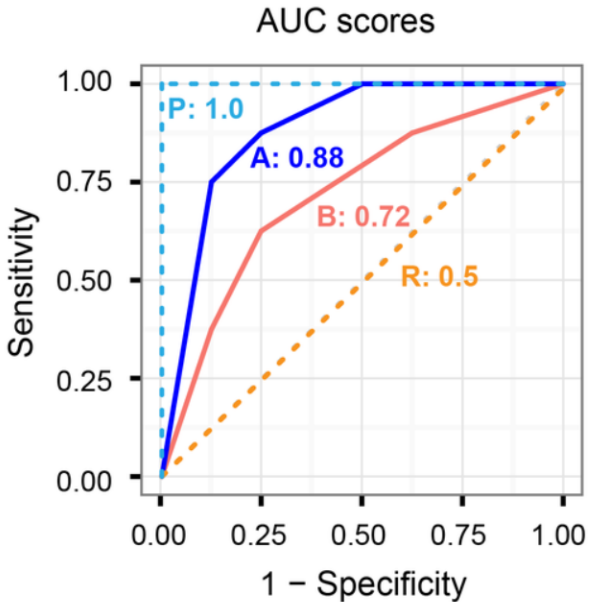
Two ROC curves represent the performance levels of two classifiers A and B. Classifier A clearly outperforms classifier B in this example.

https://classeval.wordpress.com/introduction/introduction-to-the-roc-receiver-operating-characteristics-plot/

Measuring Error: ROC curve

AUC (Area under the ROC curve) score

Another advantage of using the ROC plot is a single measure called the AUC (area under the ROC curve) score. As the name indicates, it is an area under the curve calculated in the ROC space. One of the easy ways to calculate the AUC score is using the trapezoidal rule, which is adding up all trapezoids under the curve.

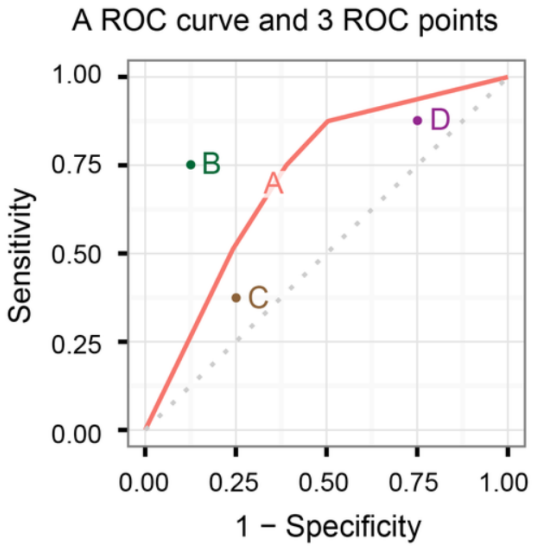


It shows four AUC scores. The score is 1.0 for the classifier with the perfect performance level (P) and 0.5 for the classifier with the random performance level (R). ROC curves clearly shows classifier A outperforms classifier B, which is also supported by their AUC scores (0.88 and 0.72).

Problems with ROC analysis

Use dissimilar datasets in one ROC plot

This is a common mistake when ROC is used for comparing multiple classifiers. One ROC curve with several ROC points are drawn in one plot. The comparison between them is valid only when the classifiers are evaluated on either a single dataset or multiple datasets that are almost identical among each other in terms of their data size and positive:negative ratio.



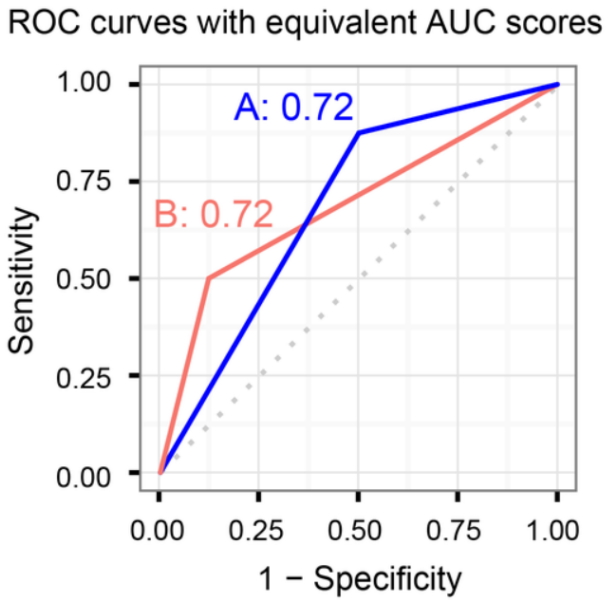
ROC curves and ROC points cannot be compared if they have been generated from different datasets. All classifiers, A, B, C, and D, should be tested on the same dataset for a valid comparison.

https://classeval.wordpress.com/introduction/introduction-to-the-roc-receiver-operating-characteristics-plot/

Problems with ROC analysis

Rely only on AUC scores

AUC scores are convenient to compare multiple classifiers. Nonetheless, it is also important to check the actual curves especially when evaluating the final model. Even when two ROC curves have the same AUC values, the actual curves can be quite different.



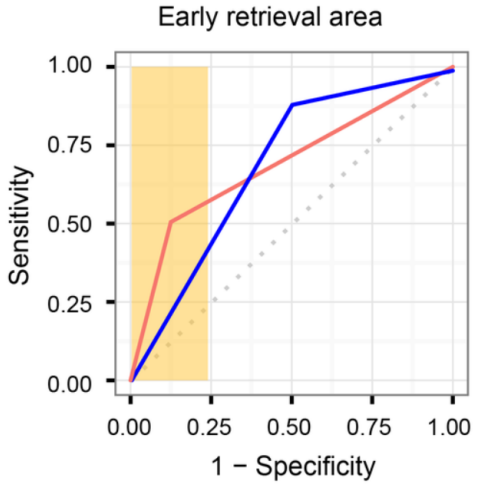
Two classifiers A and B have the same AUC scores, but their ROC curves are different.

https://classeval.wordpress.com/introduction/introduction-to-the-roc-receiver-operating-characteristics-plot/

Problems with ROC analysis

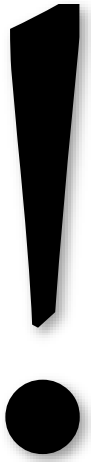
ROC analysis in conjunction with imbalanced datasets

ROC becomes less powerful when used with imbalanced datasets. One effective approach to avoid the potential issues with imbalanced datasets is using the early retrieval area, which is a region with high specificity values in the ROC space. Checking this area is useful to analyse the performance with fewer false positives (or small false positive rate).



The yellow region of the ROC plot indicates the early retrieval area. It is useful to check this area especially when the dataset is imbalanced.

Or use other error measures, e.g. F1 score



https://classeval.wordpress.com/introduction/introduction-to-the-roc-receiver-operating-characteristics-plot/

Hypothesis Testing

- Reject a null hypothesis if not supported by the sample with enough confidence
- $\mathcal{X} = \{ x^t \}_t$ where $x^t \sim \mathcal{N}(\mu, \sigma^2)$

$$H_0: \mu = \mu_0 \text{ vs. } H_1: \mu \neq \mu_0$$

Accept H_0 with level of significance α if μ_0 is in the 100(1- α) confidence interval

$$\frac{\sqrt{N}(m - \mu_0)}{\sigma} \in \left(-z_{\alpha/2}, z_{\alpha/2}\right)$$

Two-sided test

	Decision	
Truth	Accept	Reject
True	Correct	Type I error
False	Type II error	Correct (Power)

- One-sided test: H_0: $\mu \leq \mu_0$ vs. H_1: $\mu > \mu_0$
  Accept if
$$\frac{\sqrt{N}(m-\mu_0)}{\sigma} \in (-\infty, z_\alpha)$$

- Variance unknown: Use t, instead of z
  Accept H_0: $\mu = \mu_0$ if
$$\frac{\sqrt{N}(m-\mu_0)}{S} \in (-t_{\alpha/2,N-1}, t_{\alpha/2,N-1})$$

Resampling and K–Fold Cross–Validation

- The need for multiple training/validation sets $\{\mathcal{X}_i, \mathcal{V}_i\}_i$: Training/validation sets of fold i
- K-fold cross-validation: Divide $\mathcal{X}$ into k, $\mathcal{X}_i, i=1,\ldots,K$

$$\mathcal{V}_1 = \mathcal{X}_1 \quad \mathcal{T}_1 = \mathcal{X}_2 \cup \mathcal{X}_3 \cup \Lambda \cup \mathcal{X}_K$$
$$\mathcal{V}_2 = \mathcal{X}_2 \quad \mathcal{T}_2 = \mathcal{X}_1 \cup \mathcal{X}_3 \cup \Lambda \cup \mathcal{X}_K$$
$$\text{M}$$
$$\mathcal{V}_K = \mathcal{X}_K \quad \mathcal{T}_K = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \Lambda \cup \mathcal{X}_{K-1}$$

- $\mathcal{T}_i$ share K-2 parts

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!

Comparing Classifiers: $H_0: \mu_0 = \mu_1$ vs. $H_1: \mu_0 \neq \mu_1$

- Single training/validation set: McNemar's Test

e_{00}: Number of examples misclassified by both	e_{01}: Number of examples misclassified by 1 but not 2
e_{10}: Number of examples misclassified by 2 but not 1	e_{11}: Number of examples correctly classified by both

- Under H_0, we expect $e_{01} = e_{10} = (e_{01} + e_{10})/2$

$$\frac{(|e_{01} - e_{10}| - 1)^2}{e_{01} + e_{10}} \sim \chi_1^2$$

Accept if $< \chi^2_{\alpha,1}$

K–Fold CV Paired t Test

- Use K-fold cv to get K training/validation folds
- p_i^1, p_i^2: Errors of classifiers 1 and 2 on fold i
- $p_i = p_i^1 - p_i^2$: Paired difference on fold i
- The null hypothesis is whether p_i has mean 0

$$H_0 : \mu = 0 \text{ vs. } H_0 : \mu \neq 0$$

$$m = \frac{\sum_{i=1}^{K} p_i}{K} \qquad s^2 = \frac{\sum_{i=1}^{K} (p_i - m)^2}{K-1}$$

$$\frac{\sqrt{K}(m-0)}{s} = \frac{\sqrt{K} \cdot m}{s} \sim t_{K-1} \text{ Accept if in } \left(-t_{\alpha/2,K-1}, t_{\alpha/2,K-1}\right)$$