Measuring accuracy

To evaluate our model (evaluation, test) or to choose a model (model selection, validation) we need a metric to measure how well our model is performing.

Depending on the task, different metrics may be appropriate

- Binary classification
- Multiclass classification
- Regression
- Detection

Evaluating classification accuracy

Types of errors for binary classifiers

- **Type 1 error:** predict true when false
- **Type 2 error:** predict false when true

Can categorize predictions wrt ground truth as:

1. True positives (TP): $y_{true} = 1, y_{pred} = 1$
2. True negatives (TN): $y_{true} = 0, y_{pred} = 0$
3. False positives (FP): $y_{true} = 0, y_{pred} = 1$
4. False negatives (FN): $y_{true} = 1, y_{pred} = 0$

		y_{PRED}	
		T	**F**
y_{TRUE}	**T**	TP	FN
	F	FP	TN

Metrics for classification

Classification accuracy: proportion of correctly classified results

```
np.mean(y_true == y_pred)
```

Can also be written in terms of TP, FP, TN, FN

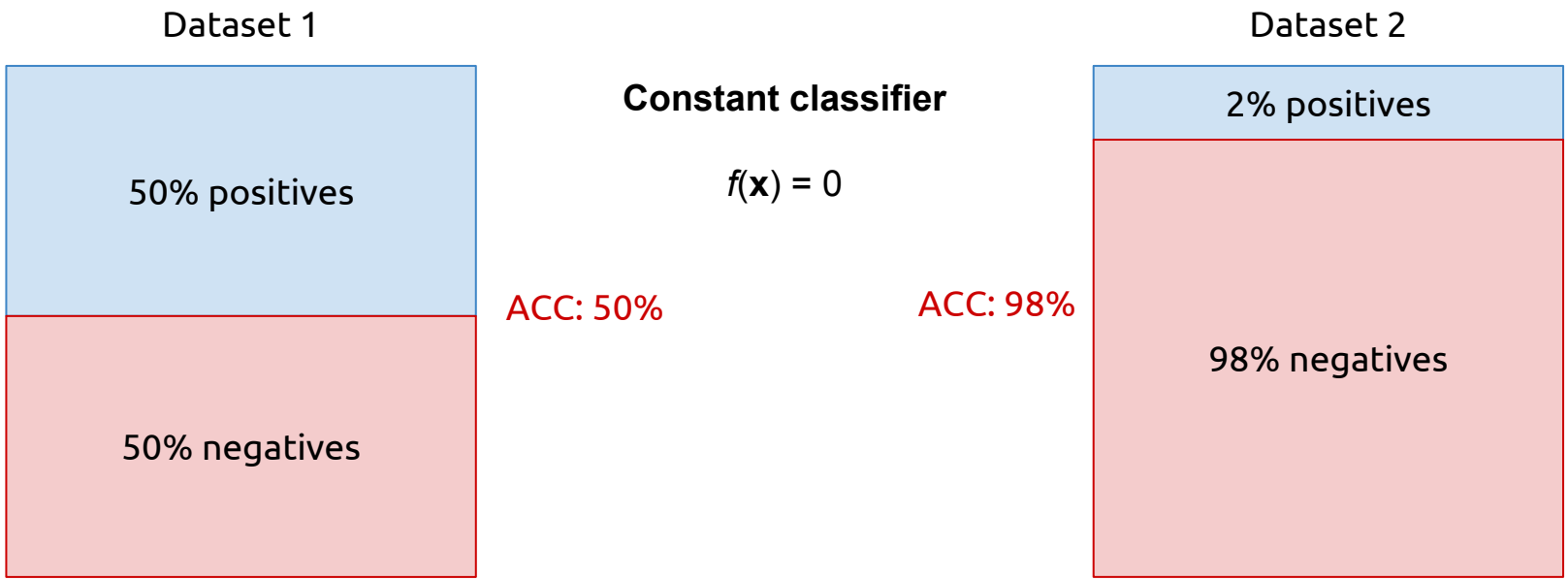
```
ACC = (TP + TN) / (TP + TN + FP + FN)
```

Also known as **zero-one accuracy.** Zero-one error = `1 - ACC`

		y_{PRED}	
		T	**F**
y_{TRUE}	**T**	TP	FN
	F	FP	TN

Considerations for using accuracy

If the the dataset is not balanced, then the value of accuracy may be misleading!



Balanced accuracy

Can be beneficial to use balanced accuracy when you have many more of one class than the other.

Proportion of total negatives marked as negative

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{TP}{P} + \frac{TN}{N}\right)$$

Proportion of total positives marked as positive

Same as accuracy when dataset is balanced

		y_{PRED}	
		T	**F**
y_{TRUE}	**T**	TP	FN
	F	FP	TN

Precision and recall

Consider a **search engine** that classifies documents as {relevant, non-relevant} based on a query

In addition to accuracy, would also be useful to know:

- What proportion of returned results are actually relevant
- What proportion of all relevant results are returned

Two metrics you can use for this are **precision** and **recall**

Precision

The proportion of elements marked as positive that are correct.

$$\text{Precision} = \frac{TP}{P} = \frac{TP}{TP + FP}$$

Information retrieval: proportion of documents returned that are relevant

Also known as PPV: positive predictive value

		y_{PRED}	
		T	**F**
y_{TRUE}	**T**	TP	FN
	F	FP	TN

Recall

The proportion all true events that are marked as true

$$\text{Recall} = \frac{TP}{T} = \frac{TP}{TP + FN}$$

Information retrieval: proportion of all relevant documents that were found by the system

AKA: true positive rate (Sensitivity)

		y_{PRED}	
		T	**F**
y_{TRUE}	**T**	TP	FN
	F	FP	TN

Precision recall curve

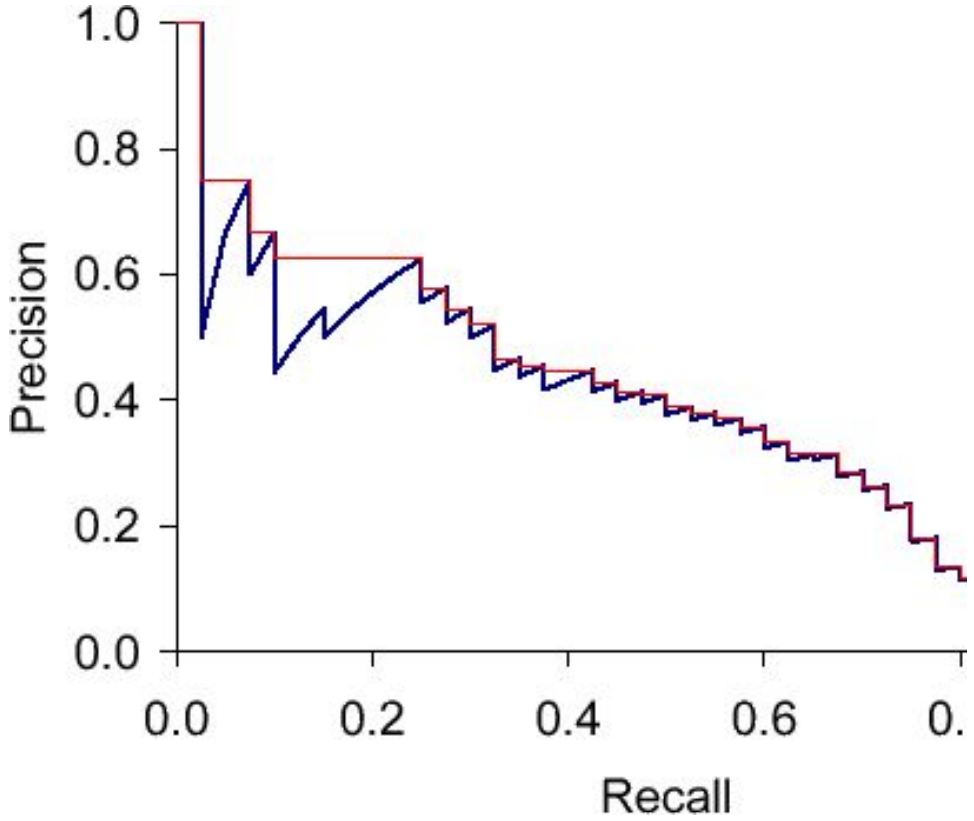
Easy to get perfect precision. $f(\mathbf{x}) = 0$ (return no document)

Easy to get perfect recall. $f(\mathbf{x}) = 1$ (return every document)

Often our classifier produces a score. Thresholding the score at different values will give different precision and recall values.

The curve traced out by all possible thresholds is called the **precision recall curve**

Precision recall curve



F_1 scores

Sometimes it is useful to have a single metric that combines precision and recall. The F_1 score is the harmonic mean of precision and recall.

$$F_1 = \frac{2(\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

Can also be calculated at many points on the PR curve (i.e. as a function of threshold)

Average precision

Precision is a function of the threshold you set: p(t)

Integrating over all possible thresholds [0..1] gives the area under the precision curve, known as average precision.

$$\text{AveP} = \int_0^1 p(t)dt$$

Can be computed by ranking by classifier score and computing the average value of precision@k

Notes on precision and recall based metrics

Precision and recall both use **true positives** proportions.

Assumes there is something special about the +1 class. Often true, e.g. for:

- Document retrieval
- Face detection
- Intrusion detection

But may not always be true. E.g. classify image as {dog, cat}. Which is the true case?

$$\text{Precision} = \frac{TP}{P} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{T} = \frac{TP}{TP + FN}$$

True and false positive rates

True positive rate (same as Recall):

```
TPR = TP / (TP + FN)
```

False positive rate:

```
FPR = FP / (FP + TN)
```

Using these to draw a curve is not biased toward the +1 class…

		y_{PRED}	
		T	**F**
y_{TRUE}	**T**	TP	FN
	F	FP	TN

True and false positive rates

Example: intrusion detection system

- **True positive rate** tells you how often your system will raise the alarm when there is indeed a burglar present P(Alarm|Burglar)
- **False positive rate** tells you how often your system will raise the alarm when there is no burglar present P(Alarm|No burglar)

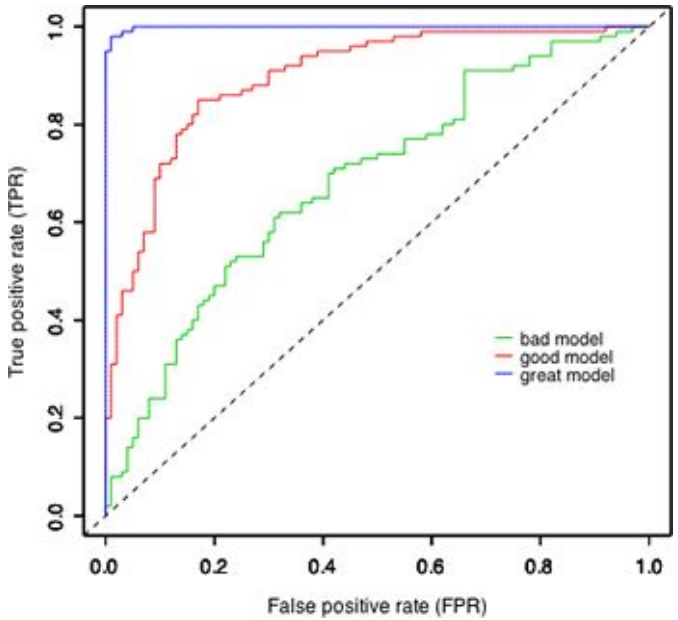
The receiver operator characteristic

ROC curve. Graph of TPR vs FPR as you vary the threshold.

- Sensitivity vs. 1-Specificity
- P(true|true) vs. P(true|false)

Called ROC for historical reasons (radar)

Different applications may prefer different points on the ROC



AUC

To reduce the ROC curve to a single metric, you can measure the **area under the curve**

This is called the AUC metric (or ROC-AUC)

AUC represents performance averaged over all possible cost ratios

AUC	Interpretation
1.0	perfect prediction
0.9	excellent prediction
0.8	good
0.7	mediocre
0.6	poor
0.5	random
<0.5	bug in code!

Multiclass classification and error analysis

For multiclass problems we can compute overall classification accuracy.

Can also compute binary metrics for each class, treating all other classes as negatives

In this case, the positive class *is* special, so it makes sense to use precision and recall

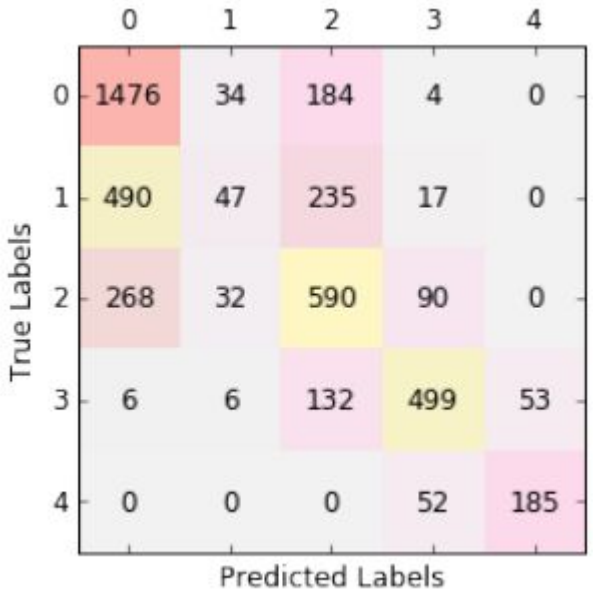
```
import sklearn.metrics as metrics

metrics.classification_report(y_true, y_pred)
```

class	precision	recall	f1-score	support
0	1	1	1	6451190
1	0.9	0.91	0.9	616014
2	0.94	0.95	0.94	89966
3	0.93	0.93	0.93	609634
4	0.62	0.52	0.57	6956
5	0.85	0.88	0.87	24057
6	0.95	0.96	0.96	122025
7	0.92	0.91	0.92	36384
8	0.93	0.9	0.91	456774
avg/tot	0.98	0.98	0.98	8413000

Multiclass classification and error analysis

The **confusion matrix** can be used to analyse which classes are most often mixed up by the classifier

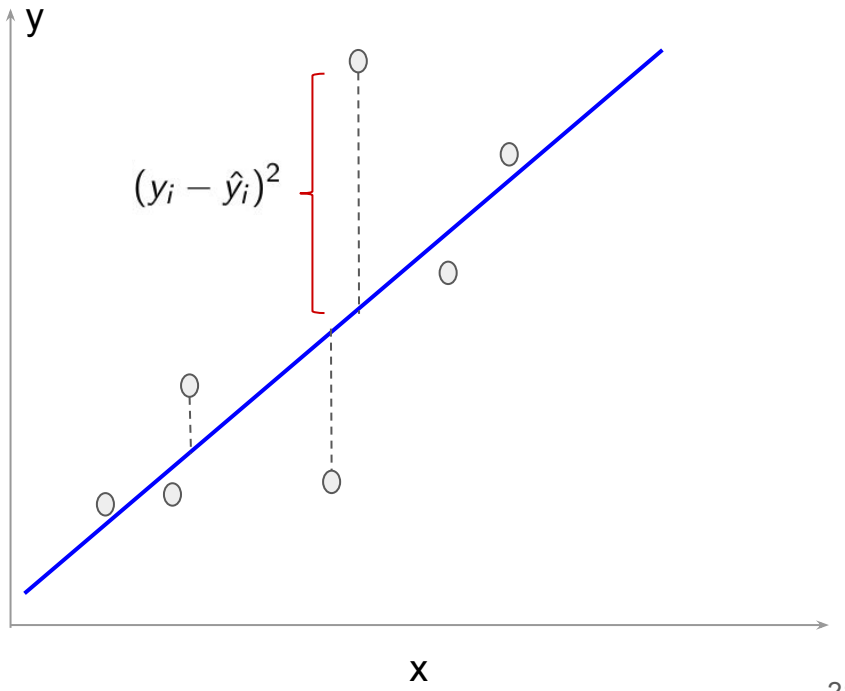
Off diagonal elements represent errors.



Metrics for regression

Mean squared error (MSE) is standard for regression problems.

$$E = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$



Mean absolute error

Mean squared error heavily penalizes points that are far from the prediction

Mean absolute error (MAE) should be used in situations where you want to be more robust to outliers (e.g. noisy ground truth)

$$E = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|$$