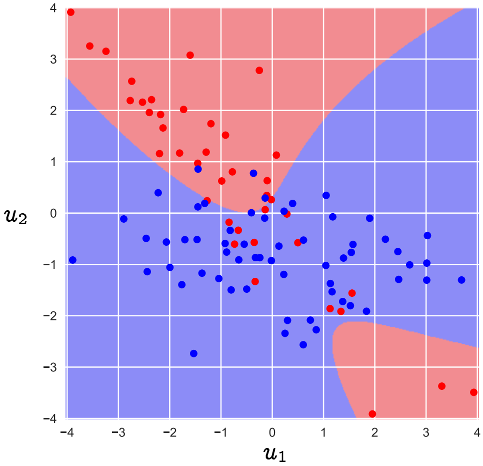
Categorical outputs

- we consider categorical raw outputs, $v \in \mathcal{V}$, $\mathcal{V}$ a finite set
- $\mathcal{V} = \{v_1, \ldots, v_K\}$ is the *label set*; v_i are called *classes* or *labels* or *categories*
- called *Boolean* for $K = 2$, *e.g.*,
    - $\mathcal{V} = \{\text{TRUE}, \text{FALSE}\}$
    - $\mathcal{V} = \{\text{POSITIVE}, \text{NEGATIVE}\}$
- called *multi-class* for $K > 2$, *e.g.*,
    - $\mathcal{V} = \{\text{YES, MAYBE, NO}\}$
    - $\mathcal{V} = \{\text{ALBANIA, AZERBAIJAN}, \ldots\}$
    - $\mathcal{V} = \{\text{HINDI, TAMIL}, \ldots\}$
    - $\mathcal{V} =$ set of English words in some dictionary
    - $\mathcal{V} =$ set of $m!$ possible orders of m horses in a race
- we often take $\mathcal{V} = \{1, \ldots, K\}$

Classifiers

- predicting a categorical raw output $v \in \mathcal{V}$ given a raw input $u \in \mathcal{U}$ is called *classification*
- called *Boolean classification* when $K = 2$
- called *multi-class classification* when $K > 2$
- predictor has form $G : \mathcal{U} \to \mathcal{V}$
- $\hat{v} = G(u)$ is our prediction of v, given u
- in this context, G is called a *classifier*
- roughly speaking, classifier classifies all $u \in \mathcal{U}$ into those with predictions $G(u) = v_i$, $i = 1, \ldots, K$

Example



- $\mathcal{U} = \mathbf{R}^2$, $\mathcal{V} = \{-1, 1\}$
- classifier shown with data set $u^1, \ldots, u^n$, $v^1, \ldots, v^n$, red $= -1$ and blue $= 1$

Applications

- medical diagnosis
  - u contains patient attributes, test results
  - Boolean v encodes disease status (has disease or not), or multi-class, *e.g.*, $\mathcal{V} = \{\textsc{covid19}, \textsc{flu}, \textsc{cold}\}$
- advertising
  - u contains attributes of a person and an ad shown to them
  - v encodes whether they buy the item, click on the ad, *etc.*.
- fraud detection
  - u contains attributes of a proposed transaction
  - $v \in \mathcal{V} = \{\textsc{fraud}, \textsc{valid}\}$
- image classification
  - u is an image
  - $v \in \mathcal{V} = \{\textsc{lion}, \textsc{tree}, \textsc{bus}, \ldots\}$

Applications

- spam filter
  - u contains attributes of an email message
  - $v \in \mathcal{V} = \{\textsc{spam}, \textsc{ham}\}$
- sports forecasting
  - u contains attributes of a game or match, team A versus team B
  - v encodes game winner, $\mathcal{V} = \{\mathrm{A}, \mathrm{B}, \textsc{tie}\}$
- topic detection
  - u is an article or news item
  - v encodes topic, *e.g.*$\mathcal{V} = \{\textsc{politics}, \textsc{sports}, \textsc{business}, \ldots\}$
- sentence parsing
  - u is a sentence
  - v encodes grammatical parsing of sentence (a labeled tree)

Performance metrics for Boolean classification

Error rate

- we are given a data set $u^1, \ldots, u^n$, $v^1, \ldots, v^n$
- predictions are $\hat{v}^i = G(u^i)$, $i = 1, \ldots, n$
- prediction is *correct* if $\hat{v} = v$, *wrong* or *error* if $\hat{v} \neq v$
- *error rate* E is fraction of errors,
$$E = \frac{1}{n} \left| \{ i \mid \hat{v}^i \neq v^i \} \right|$$
  ($|A|$ is the number of elements of a finite set A)
- error rate is the simplest performance metric for a classifier
- we can *validate* a classifier by evaluating its error rate on unseen or held back (test) data

The two types of errors in Boolean classification

- consider Boolean classification with $\mathcal{V} = \{-1, 1\}$
- class $v = -1$ is called *negative*, $v = 1$ is called *positive*
- only four possible values for the data pair $\hat{v}$, v:
  - *true positive* if $\hat{v} = 1$ and $v = 1$
  - *true negative* if $\hat{v} = -1$ and $v = -1$
  - *false negative* or *type II error* if $\hat{v} = -1$ and $v = 1$
  - *false positive* or *type I error* if $\hat{v} = 1$ and $v = -1$

Boolean confusion matrix

- for a predictor and a data set the confusion matrix is

$$C = \begin{bmatrix} \#\ \text{true negatives} & \#\ \text{false negatives} \\ \#\ \text{false positives} & \#\ \text{true positives} \end{bmatrix} = \begin{bmatrix} C_{\mathsf{tn}} & C_{\mathsf{fn}} \\ C_{\mathsf{fp}} & C_{\mathsf{tp}} \end{bmatrix}$$

- $C_{\mathsf{tn}} + C_{\mathsf{fn}} + C_{\mathsf{fp}} + C_{\mathsf{tp}} = n$ (total number of examples)
- $N_{\mathsf{n}} = C_{\mathsf{tn}} + C_{\mathsf{fp}}$ is number of negative examples
- $N_{\mathsf{p}} = C_{\mathsf{fn}} + C_{\mathsf{tp}}$ is number of positive examples
- diagonal entries give numbers of correct predictions
- off-diagonal entries give numbers of incorrect predictions of the two types

Some Boolean classification performance metrics

- confusion matrix $\begin{bmatrix} C_{\mathrm{tn}} & C_{\mathrm{fn}} \\ C_{\mathrm{fp}} & C_{\mathrm{tp}} \end{bmatrix}$

- the basic error measures:
  - *false positive rate* is C_{fp}/n
  - *false negative rate* is C_{fn}/n
  - *error rate* is $(C_{\mathrm{fn}} + C_{\mathrm{fp}})/n$

- error measures some people use:
  - *true positive rate* or *sensitivity* or *recall* is $C_{\mathrm{tp}}/N_{\mathrm{p}}$ (fraction of true positives we correctly guess)
  - *false alarm rate* is $C_{\mathrm{fp}}/N_{\mathrm{n}}$ (fraction of true negatives we incorrectly guess as positive)
  - *specificity* or *true negative rate* is $C_{\mathrm{tn}}/N_{\mathrm{n}}$ (fraction of true negatives we correctly guess)
  - *precision* is $C_{\mathrm{tp}}/(C_{\mathrm{tp}} + C_{\mathrm{fp}})$ (fraction of our positive guesses that really are positive)

Neyman-Pearson metric

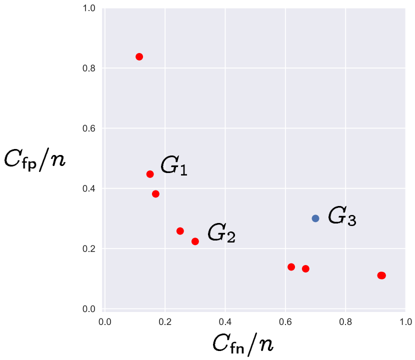
- we have *two* metrics or objectives for a Boolean classifier: false positive and false negative rate
- we want both small
- to obtain a single (number) metric, we combine them with a weight to get the *Neyman-Pearson metric*

$$E^{\mathsf{NP}} = \kappa C_{\mathsf{fn}}/n + C_{\mathsf{fp}}/n$$

- $\kappa > 0$ sets how much we care about false negatives, compared to false positives
  - for $\kappa > 1$, false negatives upset us more than false positives
  - for $\kappa < 1$, false negatives upset us less than false positives
  - for $\kappa = 1$, $E^{\mathsf{NP}} = E$, the overall error rate

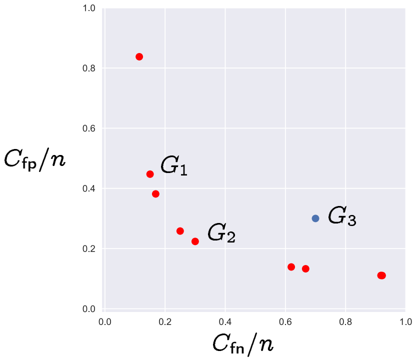
False positive and false negatives

- Boolean classifier has two objectives: false positive rate and true positive rate
- plot the performance of each classifier
- G_3 is worse than G_2 (more false positives and more false negatives)
- G_1 has fewer false negatives than G_2, but more false positives



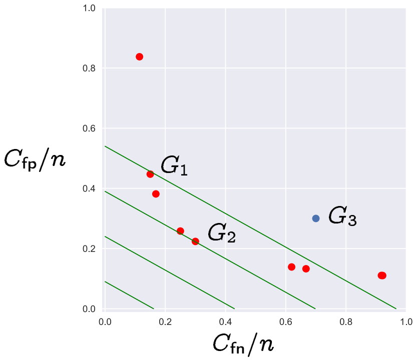
ROC curve

- red points are *Pareto optimal*; no other classifier is better in *both* C_{fp} and C_{fn}
- set of all Pareto optimal points is called the *ROC* or *operating characteristic*
- ROC stands for Receiver Operating Characteristic (from WWII, never spelled out)
- it is common to develop multiple classifiers, which trade off these two error rates



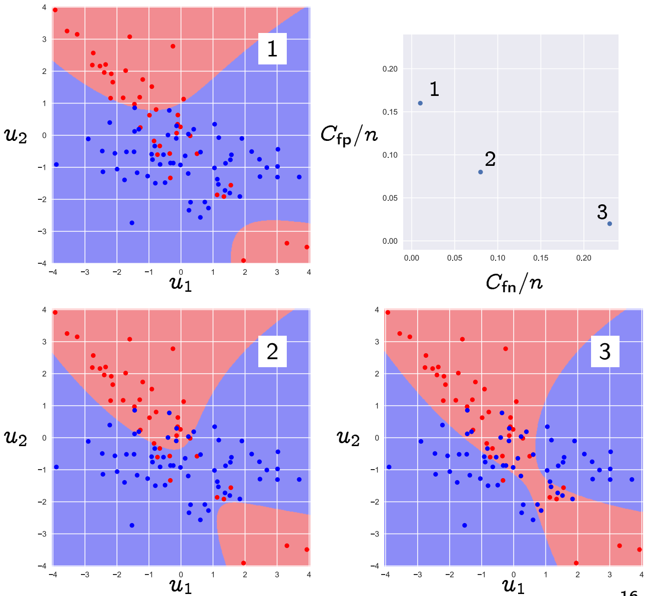
Neyman-Pearson error

- we can measure performance in different directions in this plane
- let $\kappa > 0$ be how much more false negatives irritate us than false positives
- instead of using the error-rate as a performance metric, use the weighted-sum
$$\kappa C_{\mathsf{fn}}/n + C_{\mathsf{fp}}/n$$
- a *scalarization* of two objectives called the *Neyman-Pearson error*
- when $\kappa = 1$, the Neyman-Pearson error is the *error rate*
- each green line shows points where $\kappa C_{\mathsf{fn}}/n + C_{\mathsf{fp}}/n$ is constant; slope of dashed lines is $-\kappa$



Example

- red points have $v = -1$, blue have $v = 1$
- false negative are blue points for which the classifier would predict red
- plot 1 has $C = \begin{bmatrix} 24 & 1 \\ 16 & 59 \end{bmatrix}$
- plot 2 has $C = \begin{bmatrix} 32 & 8 \\ 8 & 52 \end{bmatrix}$
- plot 3 has $C = \begin{bmatrix} 38 & 23 \\ 2 & 37 \end{bmatrix}$



Performance metrics for multiclass classification

Error types

- there are K^2 possible values of $(\hat{v}, v)$, since $\hat{v}, v \in \{v_1, \ldots, v_k\}$
- $\hat{v} = v_i$, $v = v_j$ means the true value is v_j, and we predict v_i
- prediction is correct when $v_i = v_j$, and an error when $v_i \neq v_j$
- we further distinguish $K(K-1)$ *types of errors*, one for each pair i, j with $i \neq j$
- for $i \neq j$, $\hat{v} = v_i$, $v = v_j$ means we mistook v_j for v_i
- *i.e.*, the value is v_j, but we guess v_i

Confusion matrix

- $K \times K$ *confusion matrix* is defined by

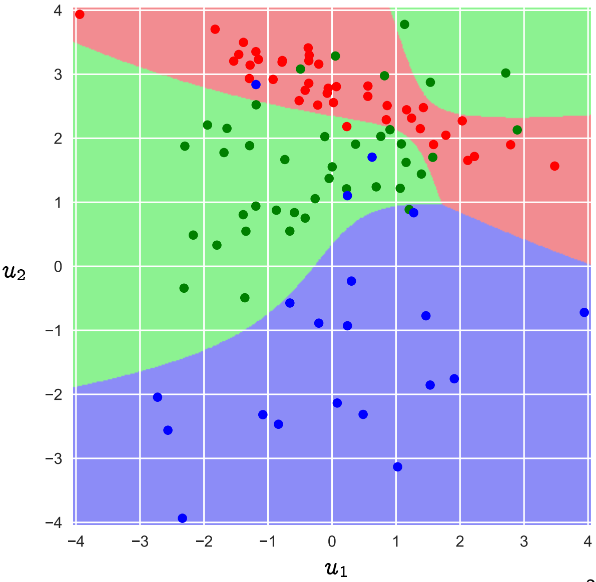
$$C_{ij} = \#\text{ records with } \hat{v} = v_i \text{ and } v = v_j$$

  (warning: some people use the transpose of C)

- entries in C add up to n
- column sums of C give number of records in each class in the data set
- C_{ii} is the number of times we predict v_i correctly
- C_{ij} for $i \neq j$ is the number of times we mistook v_j for v_i
- there are $K(K-1)$ different *error rates*, $E_{ij} = C_{ij}/n$, $i \neq j$
- the overall error rate is $E = \sum_{i \neq j} C_{ij}/n = \sum_{i \neq j} E_{ij}$

Example

- red = 1, green = 2, blue = 3
- confusion matrix $C = \begin{bmatrix} 39 & 5 & 1 \\ 1 & 34 & 2 \\ 0 & 1 & 17 \end{bmatrix}$
- error rates $E = \begin{bmatrix} 0 & 0.05 & 0.01 \\ 0.01 & 0 & 0.02 \\ 0 & 0.01 & 0 \end{bmatrix}$
- error rate = 10%



Neyman-Pearson error

- $E_j = \sum_{i \neq j} C_{ij}$ is number of times we mistook v_j for another class
- E_j/n is the error rate of mistaking v_j
- we will scalarize these K error rates using a weighted sum
- the *Neyman-Pearson error* is
$$\sum_{j=1}^{K} \kappa_j E_j = \sum_{i \neq j} \kappa_j C_{ij}/n$$
where κ is a weight vector with nonnegative entries
- κ_j is how much we care about mistaking v_j
- for $\kappa_j = 1$ for all i, Neyman-Pearson error is the *error rate*