AdaBoost for binary classification

Data set $(x_i, y_i)_{i=1,\dots,N}$ — $x_i \in \mathbb{R}^n$, $y_i \in \{-1,1\}$. Two classes: $-1, 1$.

- $C_0(x) = 0$
- $C_m(x_i) = C_{m-1}(x_i) + \alpha_m k_m(x_i)$.

α_m: constant; k_m: classifier with values in $\{-1,1\}$

5.5 - grade
find some interesting project connected with ML, [redacted] share with me, if accepted, carry out and get 5.5

How to choose α_m, k_m? To minimise the total error E of C_m:

$$E = \sum_{i=1}^{N} e^{-y_i C_m(x_i)} = \sum_{i=1}^{N} \underbrace{e^{-y_i C_{m-1}(x_i)}}_{w_i^{(m)}} e^{-y_i \alpha_m k_m(x_i)} = \sum_{i=1}^{N} w_i^{(m)} e^{-y_i \alpha_m k_m(x_i)} =$$

$$= \sum_{i: y_i = k_m(x_i)} w_i^{(m)} e^{-\alpha_m} + \sum_{i: y_i \neq k_m(x_i)} w_i^{(m)} e^{\alpha_m} = \underbrace{\left(\sum_{i=1}^{N} w_i^{(m)}\right) e^{-\alpha_m}}_{\text{does not depend on } k_m} + \sum_{i: y_i \neq k_m(x_i)} w_i^{(m)} \underbrace{\left(e^{\alpha_m} - e^{-\alpha_m}\right)}_{\text{does not depend on } i \text{ nor on } k_m}$$

So k_m should be chosen to minimise $\sum_{i: y_i \neq k_m(x_i)} w_i^{(m)}$, i.e., weighted error.

Then, given k_m, choose α_m to minimize $\bar{E}$:

$$\alpha_m = \frac{1}{2} \ln \frac{\sum_{i: y_i = k_m(x_i)} w_i^{(m)}}{\sum_{i: y_i \neq k_m(x_i)} w_i^{(m)}}$$

After repeating for $m = 1, 2, \ldots, M$ we obtain some C_M.

classification is done by $\operatorname{sgn}(C_M(x))$

Drawbacks:

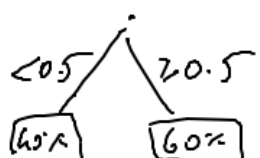
excessive weight is assigned to outliers

(there are some other versions)

Handling missing data

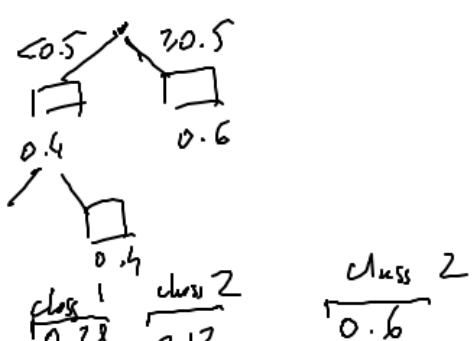
$(x_1, \ldots, x_n, y)$

- ignore records with missing values (it could result in a bias)
- 'guess' the missing attribute value
- treat missing values as a special category
- split the sample to "subsamples" (used in e.g. C4.5 — Ross Quinlan)



(from the samples with the attr. value)

for the classification:



there are different methods to treat missing data in decision tree, but mostly is scipy

weka

Imbalanced data

classification problem, in which one class is substantially smaller than one of the others

e.g. testing for a (rare) disease

one method to tackle this problem: use bootstrap to make the classes balanced

Confusion matrix

(binary classification)

$p^* = \frac{1}{9+1} = \frac{1}{10}$

9

	actual −	actual +
predicted − (0)	60%	10%
predicted + (1)	10%	20%

1 1

C(0,1) ← cost of misclassifying 1

C(1,0)

accuracy = 60% + 20% = 80%

Often, one wants to avoid one of the types of the error, usually that one

* cost-sensitive classification:

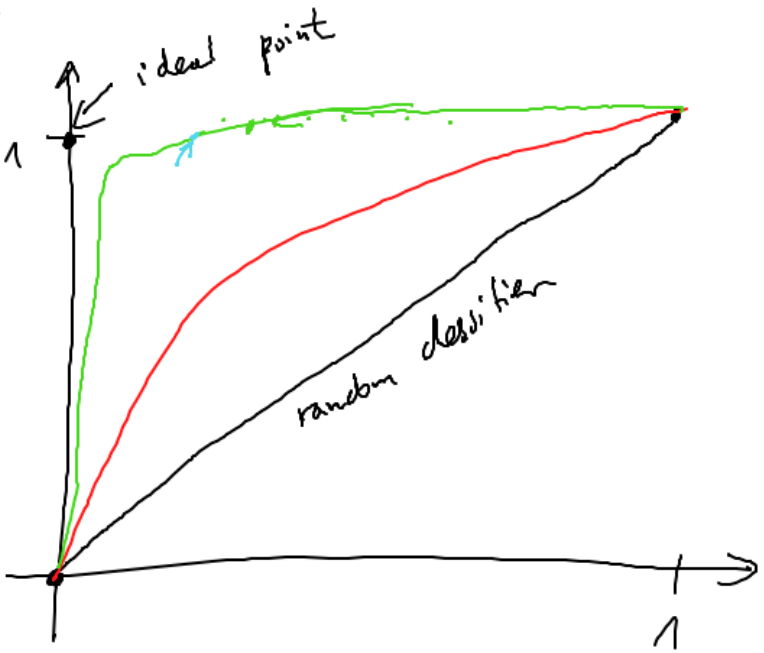
introduce a threshold $p^* = \frac{C(1,0)}{C(0,1)+C(1,0)}$

classify x as of class 1 (i.e., +), if

$P(1|x) \geq p^*$

↑ estimated prob. of being in class 1

ROC curve ← (receiver operating characteristic)



true positive rate

$$\frac{TP}{TP+FN}$$

Confusion matrix	actual −	actual +
predicted −	TN	FN
predicted +	FP	TP

avoid that error

−	1−α	α
+	0	0

false positive rate $= \frac{FP}{FP+TN}$

	−	+
−	0	0
+	π−α	α

random classifier: randomly classify each sample as being in class 1 (with prob. β) or −1 (with prob. $1-\beta$)

$$\frac{TP}{TP+FN} = \frac{\beta\alpha}{\alpha} = \beta \qquad \frac{FP}{FP+TN} = \frac{\beta(1-\alpha)}{1-\alpha} = \beta$$

$(1-\alpha)$	α
$(1-\beta)(1-\alpha)$	$(1-\beta)\alpha$
$\beta(1-\alpha)$	$\beta\alpha$

There is some tradeoff between FPR and TPR to be made

Logistic regression

binary classification problem — X - predictors ($\in \mathbb{R}^n$), $Y \in \{0,1\}$ class

model for $p(x) = \Pr(Y=1 \mid x)$:

$$p(x) = \varphi(\underbrace{\beta_0 + \beta_1 x}_{\text{linear regression}}) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}$$

$(\beta_0 \in \mathbb{R},\ \beta_1 \in \mathbb{R}^n)$

we take logistic function:

$$\varphi(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$$

$$(1+e^{\beta_0+\beta_1 x})\, p(x) = e^{\beta_0+\beta_1 x}$$

$$p(x) = e^{\beta_0+\beta_1 x}(1-p(x))$$

$$e^{\beta_0+\beta_1 x} = \frac{p(x)}{1-p(x)}$$

$$\beta_0 + \beta_1 x = \log\left(\frac{p(x)}{1-p(x)}\right)$$

fitting: maximising likelyhood function:

$$\ell(\beta_0, \beta_1) = \prod_{i:\, y_i=1} p(x_i) \cdot \prod_{i:\, y_i=0} (1-p(x_i))$$