Until now : how to use a model to make optimal decisions

Machine learning -> how to acquire a model from data/experience

- learning paramaters (eg. prob.)
- learning structure (eg. BN graphs)
- learning hidden concepts (eg. clustering, neural nets)

Classification

example : spam filter

- input: email
- output : spam/ham
- setup:
  - get large collection of emails labeled spam or ham, hand labeled!
  - want to learn to predict labels of future emails
- features: attributes on which base the decision
  - words, eg. FREE!
  - text patterns
  - non text : senderInContacts, WidelyBroadcast
  - ...

decision given inputs x, predict labels(classes) y

Important in commercial technology.

Model based classification

- build a model (eg. BN) where both the output label and input features are random variables
- instantiate any observed features
- query for the distribution of the label conditioned on the features

what structure should the BN have?

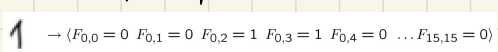
How should we learn its parameters?

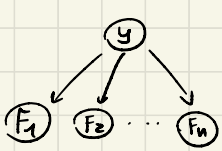
Naive Bayes

Assume all features are indipendent effects of the label

example digit recognition

- one feature (variable) F_{ij} for each grid position <i,j>
- feature values off/on based on intensity > or < 0.5 in image
- each input maps to a feature vector





lots of features with binary values

Naive Bayes model: $P(y|F_{0,0},\dots F_{15,15}) \propto P(y)\prod_{i,j} P(F_{i,j}|y)$

in general

$|Y|$ parameters

$P(Y|F_1, \dots F_n) = P(Y)\prod_i P(F_i|Y)$

$|Y| \times |F|^n$ values $n \times |F| \times |Y|$ parameters

we have to specify how each future depends on the class

total number of parameters is linear in n

Model is simplistic but often works anyway

Inference for naive Bayes

Goal: compute posterior distribution over label variable Y

1) get joint probabilits of label and evidence for each label

$$P(Y, f_1, \dots f_n) = \begin{bmatrix} P(y_1, f_1, \dots f_n) \\ P(y_2, f_1 \dots f_n) \\ \vdots \\ P(y_k, f_1 \dots f_n) \end{bmatrix} \xrightarrow{①} \begin{bmatrix} P(y_1)\prod_i P(f_i|y_1) \\ \vdots \\ P(y_k)\prod_i P(f_i|y_k) \end{bmatrix} + ②$$

2) sum to get probability of evidence

$P(f_1 \dots f_n)$ ③

3) normalize by dividing ① by ②

$$P(Y|f_1 \dots f_n)$$

what do we need to use naive Bayes?

- inference method
- Estimates of local conditional probabilits tables
  - $P(Y)$ prior over labels
  - $P(F_i|Y)$ for each feature (evidence variable)
  - these parameters are collectively called the parameters of the model (θ)

Training and testing

empirical risk minimization

we want the model that does best on the true test distribution

we don't know the true distribution, so pick the best model on our actual training set

↑ optimisation problem

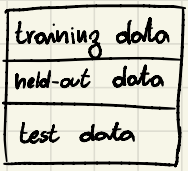
main worry: overfitting to the training set

better with more training data

better if we limit the complexity of our hypothesis

Data: labeled instances

features: attribute-value pairs which characterize each x.



experimentation cycle

- learn parameters on training set
- Tune hyperparameters on held-out set
- Compute accuracy of test set

Evaluation: over many possible metrics e.g. accuracy

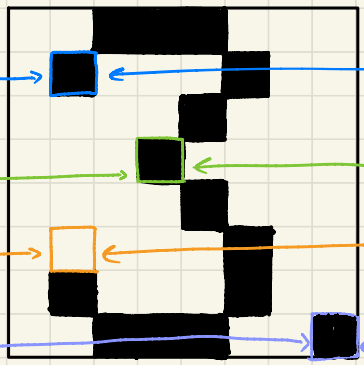
Accuracy -> fraction of instances predicted correctly

Overfitting and generalization

want a classifier which does well on test data

overfitting -> fitting training data very closely but not generalizing well.

example: overfitting



P(features, C=2)

P(C=2) = 0.1

P(on|C=2) = 0.8

P(on|C=2) = 0.1

P(on|C=2) = 0.1

P(on|C=2) = 0.01

P(features, C=3)

P(C=3) = 0.1

P(on|C=3) = 0.8

P(on|C=3) = 0.9

P(on|C=3) = 0.7

P(on|C=3) = 0.0

because of this random pixel with prob. 0 the prob of 3 is gonna get nullified, even tho up until that point it was the most likely one, so 2 wins

Dont give probabilits 0!

Parameter estimation

estimating the distribution of a random variable

elicitation: ask a human (hard)

empirically: use training data (learning)

for each outcome x look at the empirical rate of that value:

$$P_{ML}(x) = \frac{\text{count}(x)}{\text{total samples}}$$ ← estimate that maximizes the likelihood of the data

$$l(x, \theta) = \prod_i P_\theta(x_i)$$

Smoothing

- Relative frequencies are the maximum likelihood estimates

$$\theta_{ML} = \arg\max_\theta P(X|\theta) = \arg\max_\theta \prod_i P_\theta(x_i) \Longrightarrow P_{ML}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

- Another option is to consider the most likely parameter value given the data

$$\theta_{MAP} = \arg\max_\theta P(\theta|X) = \arg\max_\theta P(X|\theta)\, P(\theta)/P(X) = \Longrightarrow ??? = \arg\max_\theta P(X|\theta)P(\theta)$$

Unseen events, Laplace smoothing

Laplace estimate:

pretend you saw every outcome once more than you actually did

$$P_{LAP}(x) = \frac{c(x)+1}{\sum_x [c(x)+1]} = \frac{c(x)+1}{N+|x|}$$

example ⓥ ⓥ ⓑ

$P_{ML}(x) = \langle 2/3, 1/3 \rangle$

$P_{LAP}(x) = \langle 3/5, 2/5 \rangle$

we can extend it

+K instead of +1

$$P_{LAP} = \frac{c(x)+K}{N+K|x|}$$

$P_{LAP,0}(x) = \langle 2/3, 1/3 \rangle$

$P_{LAP,1}(x) = \langle 3/5, 2/5 \rangle$

$P_{LAP,100}(x) = \langle \frac{102}{203}, \frac{101}{203} \rangle$ ← gets closer to 50/50

K is the strenght of the prior

Tuning on held-out data

we have two kinds of unknowns

parameters : $P(y)$, $P(X|y)$

Hyperparameters : eg. amount/type of smoothing to do, k, α

Learn parameters from training data

tune Hyperp. from different data, and for each one train and test on the held-out data

choose the best value and do a final test on the test data

What to do about errors

we need more features

Baselines

Help determine how hard the task is and know what a "good" accuracy is

weak baseline: most frequent label classifier

gives all test instances whatever label was most common in the training set.

for real research usually use previous work as a (strong) baseline

Confidences from a classifier

- confidence: posterior probability of the top label

$$\text{confidence}(x) = \max_y P(y|x)$$

represents how sure the classifier is of the classification

no guarantee of correctness

- calibration

weak calibration : $\uparrow$ confidence $\Rightarrow \uparrow$ accuracy

strong calibration : confidence predicts accuracy rate