COMP 790-125: Goals for today

- Generative vs. discriminative learning
- Learning parameters in a Bayes Net
- Simple learning examples: Naive Bayes
- Inference

Generative vs. discriminative

In training a logistic regression classifier we were modeling $p(y|\mathbf{x})$ where y is a label and x is a vector of features.

This is not a joint model of all of our data, we simply aimed to learn how to predict y from features $\mathbf{x}$.

This is called discriminative training: SVMs,CRFs, decision trees all fall into this category.

Generative side

If we wanted to predict a feature $\mathbf{x}$ from y but we only trained discriminatively we would be in trouble.

Generative models do not make the distinction between labels/response variables and features/predictors.

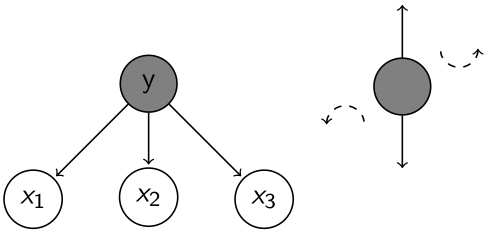
Training a generative model in a context of a classification problem amounts to learning a joint probability $p(y, \mathbf{x})$.

Naive Bayes: a generative model

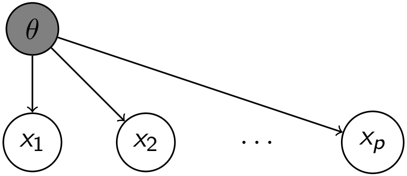
Generative model

$$\begin{aligned} p(y = c) &= \pi_c \\ p(x_j | y = c) &= f_{cj}(x_j | \theta_{cj}) \end{aligned}$$

Given label y features x_j are conditionally independent.



Conditional independencies - IID as a conditional independence



Given parameters θ the N samples drawn from $p(\mathbf{x}|\theta)$ (e.g. $x_1, x_2, \dots x_p$) are conditionally independent, and hence

$$p(x_1, \dots x_p|\theta) = \prod_i p(x_i|\theta)$$

Naive Bayes makes assumptions about independence of features, IID about independence of samples. Think of them as rows and columns of feature matrix.

Fitting Naive Bayes

Log-Likelihood

$$
\begin{aligned}
\mathrm{LL}(\theta) &= \log \prod_{i=1}^{n} p(y_i, x_{i1}, x_{i2}, \ldots, x_{ip}) \\
&= \log \prod_{i=1}^{n} p(x_{i1}, x_{i2}, \ldots, x_{ip} | y_i) p(y_i) \\
&= \log \prod_{i=1}^{n} p(x_{i1} | y_i) p(x_{i2} | y_i) \ldots p(x_{ip} | y_i) p(y_i) \\
&= \log \prod_{i=1}^{n} p(y_i) \prod_{j} p(x_{ij} | y_i) \\
&= \log \prod_{i=1}^{n} \prod_{c} \pi_c^{[y_i = c]} \prod_{j} f_{cj}(x_{ij} | \theta_{cj})^{[y_i = c]} \\
&= \sum_{i=1}^{n} \sum_{c} [y_i = c] \left(\log \pi_c + \sum_{j} \log f_{cj}(x_{ij} | \theta_{cj}) \right)
\end{aligned}
$$

Fitting Naive Bayes

The maximum likelihood fit is given by

$$\begin{aligned}\pi_c &= \frac{\sum_i [y_i = c]}{n} \\ \theta_{cj} &= \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} [y_i = c] \log f_{cj}(x_{ij}|\theta_{cj})\end{aligned}$$

The parameters θ_{cj} depend only on the data in class c and only on the values of j^{th} feature.

Prediction given a feature vector $\mathbf{x}$ is done simply by computing

$$\operatorname*{argmax}_{c} p(y = c|\mathbf{x}) = \operatorname*{argmax}_{c} \frac{p(y = c, \mathbf{x})}{p(\mathbf{x})} = \operatorname*{argmax}_{c} \pi_c \prod_j f_{cj}(x_j|\theta_{cj})$$

Getting logistic regression from a Naive Bayes

An example of Naive Bayes is a mixture of Gaussians with diagonal covariances.

Here we look at an example with a shared diagonal covariance. This covariance scales the units of different features and is not class specific (this is exactly what standardization does).

$$\begin{aligned} p(y=c) &= \pi_c \\ p(\mathbf{x}|y=c) &= \mathcal{N}(\mathbf{x}|\mu_c, \mathbf{diag}(\sigma_1^2, \ldots, \sigma_p^2)) \\ &= \prod_{j=1}^{p} \mathcal{N}(x_j|\mu_{cj}, \sigma_j^2) \end{aligned}$$

Once we know the class, each feature value can be seen as generated independently from a different univariate Gaussian.

Logistic is the shape of my $p(y|\mathbf{x})$ in that model

We will assume that we have only 2 classes

$$p(y=1|\mathbf{x}) = \frac{p(y=1)p(\mathbf{x}|y=1)}{p(y=1)p(\mathbf{x}|y=1)+p(y=2)p(\mathbf{x}|y=2)}$$

$$= \frac{\pi_1 \frac{1}{Z}\prod_j \exp\left\{-\frac{(x_j-\mu_{1j})^2}{2\sigma_j^2}\right\}}{\pi_1 \frac{1}{Z}\prod_j \exp\left\{-\frac{(x_j-\mu_{1j})^2}{2\sigma_j^2}\right\} + (1-\pi_1)\frac{1}{Z}\prod_j \exp\left\{-\frac{(x_j-\mu_{2j})^2}{2\sigma_j^2}\right\}}$$

$$= \frac{1}{1+\frac{1-\pi_1}{\pi_1}\prod_j \exp\left\{-\frac{(x_j-\mu_{2j})^2-(x_j-\mu_{1j})^2}{2\sigma_j^2}\right\}}$$

$$= \left(1+\exp\left\{-\log\left\{\frac{p}{1-p}\right\} - \sum_j \frac{x_j(\mu_{2j}-\mu_{1j})+\mu_{2j}^2-\mu_{1j}^2}{2\sigma_j^2}\right\}\right)^{-1}$$

Logistic is the shape of my $p(y|\mathbf{x})$ ... a lot of the time

If your mixture model has mixing components that come from the exponential family and take form

$$p(\mathbf{x}|\eta) = \exp\left\{\eta'\mathbf{x} - \alpha(\eta)\right\} h(\mathbf{x})$$

this includes Poisson, gamma[1], Dirichlet, multinomial then posterior probability is a logistic

$$p(y|\mathbf{x}, \eta_1, \eta_2) = \left(1 + \exp\left\{-(\eta_2 - \eta_1)'\mathbf{x} - \alpha(\eta_2) + \alpha(\eta_1)\right\}\right)^{-1}$$

[1] shape parameter is assumed fixed across classes

Naive Bayes vs logistic regression

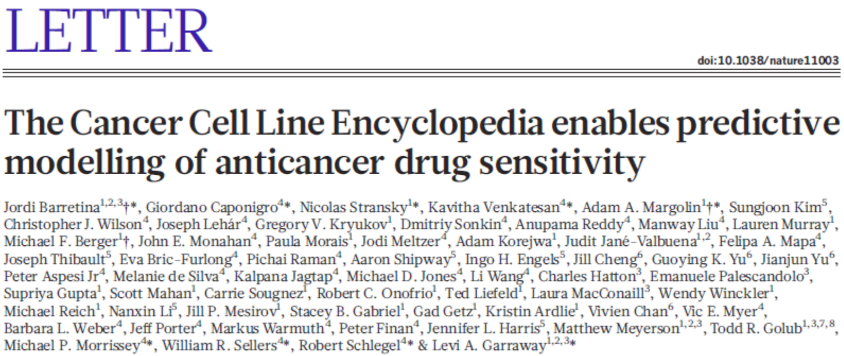
Naive Bayes: maximizes $p(y, \mathbf{x})$

Logistic Regression: maximizes $p(y|\mathbf{x})$

Folklore has it that discriminative methods do better on discriminative tasks in the presence of large amounts of data. This is not always borne out in practice.

If there is missing data or your task is not of a discriminative nature then fitting a generative model is preferred.

A recent example of NB application



LETTER

doi:10.1038/nature11003

The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity

Jordi Barretina[1,2,3]†*, Giordano Caponigro[4]*, Nicolas Stransky[1]*, Kavitha Venkatesan[4]*, Adam A. Margolin[1]†*, Sungjoon Kim[5], Christopher J. Wilson[4], Joseph Lehár[4], Gregory V. Kryukov[1], Dmitriy Sonkin[4], Anupama Reddy[4], Manway Liu[4], Lauren Murray[1], Michael F. Berger[1]†, John E. Monahan[4], Paula Morais[1], Jodi Meltzer[4], Adam Korejwa[1], Judit Jané-Valbuena[1,2], Felipa A. Mapa[4], Joseph Thibault[5], Eva Bric-Furlong[4], Pichai Raman[4], Aaron Shipway[5], Ingo H. Engels[5], Jill Cheng[6], Guoying K. Yu[6], Jianjun Yu[6], Peter Aspesi Jr[4], Melanie de Silva[4], Kalpana Jagtap[4], Michael D. Jones[4], Li Wang[4], Charles Hatton[3], Emanuele Palescandolo[3], Supriya Gupta[1], Scott Mahan[1], Carrie Sougnez[1], Robert C. Onofrio[1], Ted Liefeld[1], Laura MacConaill[3], Wendy Winckler[1], Michael Reich[1], Nanxin Li[5], Jill P. Mesirov[1], Stacey B. Gabriel[1], Gad Getz[1], Kristin Ardlie[1], Vivien Chan[6], Vic E. Myer[4], Barbara L. Weber[4], Jeff Porter[4], Markus Warmuth[4], Peter Finan[4], Jennifer L. Harris[5], Matthew Meyerson[1,2,3], Todd R. Golub[1,3,7,8], Michael P. Morrissey[4]*, William R. Sellers[4]*, Robert Schlegel[4]* & Levi A. Garraway[1,2,3]*

Inference

Given a probabilistic model $p(x_1, x_2, \dots x_p)$ we can inquire about

- (marginal) probability $p(X_A)$
- (conditional) probability $p(X_B|X_O)$

where $A, B, O \subset \{1, 2, \dots p\}$ and $B \cap O = \emptyset$

If X_O is the set of all observed variables (i.e. data) then the conditional probability $p(X_B|X_O)$ is also called posterior probability.

Inference

Further, we may inquire about most probable assignments

- $\mathrm{argmax}_v\, p(X_A = v)$
- $\mathrm{argmax}_v\, p(X_B = v | X_O)$

Again, if X_O is the set of all observed variables, the maximum of the conditional probability is called Maximum-a-Posteriori (MAP) estimate.

ML and MAP

The distinction between Maximum Likelihood and Maximum-a-Posteriori estimates

- $\operatorname{argmax}_{X_A} p(X_O|X_A)$ (ML)
- $\operatorname{argmax}_{X_A} p(X_O, X_A)$ (MAP)

Examples

- ML: fitting linear regression model

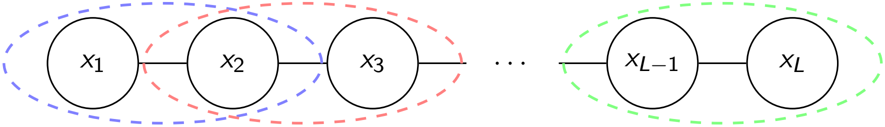
$$\operatorname*{argmax}_{\beta} p(\mathbf{y}|\mathbf{X}, \beta)$$

- MAP: fitting linear regression with a Gaussian prior (ridge penalty)

$$\operatorname*{argmax}_{\beta} p(\mathbf{y}|\mathbf{X}, \beta)p(\beta) = \operatorname*{argmax}_{\beta} p(\mathbf{y}, \beta|\mathbf{X})$$

Inference in a graphical model

Let's work with an undirected chain first



Recall that for an undirected model joint probability is given by product of potentials across cliques

$$\begin{aligned} p(\mathbf{x}) &= \frac{1}{Z}\phi_1(x_1, x_2)\phi_2(x_2, x_3)\dots\phi(x_{L-1}, x_L) \\ &= \frac{1}{Z}\prod_{l=1}^{L-1}\phi_l(x_l, x_{l+1}) \end{aligned}$$

Marginalization

Say we are interested in computing a marginal $p(x_k)$ of our undirected chain model

$$p(x_k) = \sum_{x_1}\sum_{x_2}\cdots\sum_{x_{k-1}}\sum_{x_k+1}\cdots\sum_{x_L}\frac{1}{Z}\prod_{l=1}^{L-1}\phi(x_l, x_{l+1})$$

doing this naively will be exponential in L.
Instead we distribute ϕs shrewdly

$$p(x_k) = \frac{1}{Z}\times$$
$$\sum_{x_{k-1}}\phi_{k-1}(x_{k-1}, x_k)\sum_{x_{k-2}}\phi_{k-2}(x_{k-2}, x_{k-1})\cdots\sum_{x_2}\phi_2(x_2, x_3)\sum_{x_1}\phi_1(x_1, x_2)$$
$$\times\sum_{x_{k+1}}\phi_k(x_k, x_{k+1})\cdots\sum_{x_{L-1}}\phi_{L-1}(x_{L-2}, x_{L-1})\sum_{x_L}\phi_{L-1}(x_{L-1}, x_L)$$

Marginalization – α, β messages

$$p(x_k) = \frac{1}{Z} \times$$

$$\overbrace{\sum_{x_{k-1}} \phi_{k-1}(x_{k-1}, x_k) \sum_{x_{k-2}} \phi_{k-2}(x_{k-2}, x_{k-1}) \cdots \sum_{x_2} \phi_2(x_2, x_3) \sum_{x_1} \phi_1(x_1, x_2)}^{\alpha_k(x_k)}$$

$$\times \underbrace{\sum_{x_{k+1}} \phi_k(x_k, x_{k+1}) \cdots \sum_{x_{L-1}} \phi_{L-1}(x_{L-2}, x_{L-1}) \sum_{x_L} \phi_{L-1}(x_{L-1}, x_L)}_{\beta_k(x_k)}$$

A complicated way of saying:

$$\begin{aligned}
\alpha_k(x_k) &= \sum_{x_1} \cdots \sum_{x_{k-1}} \prod_{l=1}^{k-1} \phi_l(x_l, x_{l+1}) \\
\beta_k(x_k) &= \sum_{x_{k+1}} \cdots \sum_{x_L} \prod_{l=k}^{L-1} \phi_l(x_l, x_{l+1})
\end{aligned}$$

Marginalization – α, β messages

But also

$$p(x_k) = \frac{1}{Z}\times$$

$$\overbrace{\sum_{x_{k-1}} \phi_{k-1}(x_{k-1}, x_k) \overbrace{\sum_{x_{k-2}} \phi_{k-2}(x_{k-2}, x_{k-1}) \cdots \sum_{x_2} \phi_2(x_2, x_3) \sum_{x_1} \phi_1(x_1, x_2)}^{\alpha_{k-1}(x_{k-1})}}^{\alpha_k(x_k)}$$

$$\times \underbrace{\sum_{x_{k+1}} \phi_k(x_k, x_{k+1}) \underbrace{\cdots \sum_{x_{L-1}} \phi_{L-1}(x_{L-2}, x_{L-1}) \sum_{x_L} \phi_{L-1}(x_{L-1}, x_L)}_{\beta_{k+1}(x_k+1)}}_{\beta_k(x_k)}$$

Lookie a recursion and in case you did not recognize a dynamic programming opportunity 2 slides ago it is winking at you now.

Marginalization – α, β messages

$$\begin{aligned}
\alpha_i(x_i) &= \sum_{x_{i-1}} \phi_{x_{i-1}}(x_{i-1}, x_i)\alpha_{i-1}(x_{i-1}) \\
\beta_i(x_i) &= \sum_{x_{i+1}} \phi_{x_i}(x_i, x_{i+1})\beta_{i+1}(x_{i+1})
\end{aligned}$$

In case of αs you work forward: using the table computed for α_{i-1} you compute table for α_i.
In case of β you work backward: using the table computed for β_{i+1} you compute table for β_i.

Once you have all α and all β tables, getting marginal probabilities of any stretch of variables is trivial, for example

$$p(x_i, x_{i+1}, x_{i+2}) = \frac{1}{Z}\alpha_i(x_i)\phi_i(x_i, x_{i+1})\phi_{i+1}(x_{i+1}, x_{i+2})\beta_{i+2}(x_{i+2})$$

What about Z?

$$\begin{aligned}
Z &= \sum_{x_1} \cdots \sum_{x_L} \prod_j \phi_j(x_j, x_j + 1) \\
&= \sum_{x_L} \alpha_L(x_L) \\
&= \sum_{x_1} \beta_1(x_1) \\
&= \sum_{x_i} \alpha_i(x_i)\beta_i(x_i)
\end{aligned}$$

Avoiding forward-backward bugs

Off by one is the favorite – as a result either a potential is not multiplied in or it is multiplied in a couple of times.

Luckily you know a useful fact about Z and you can assert that your forward and backward pass yield the same Z.

In your code do not multiply potentials and αs, they should be stored in the log domain and you should use the log-sum trick when you need a real domain sum, e.g.

$$\log \alpha_k(x_k) = \mathrm{logsum}(\log \alpha_{k-1}(:) + \log \phi_{k-1}(:, x_k))$$

Inference in an undirected tree

A node x_k splits a chain into two components and α_k β_k gave us their contribution to the marginal $p(x_k)$

With trees, a node splits the tree into multiple components.

$$p(x_k) = \frac{1}{Z} \prod_{x_j \in n(x_k)} m_{x_j, x_k}(x_k)$$

where $n(x_k)$ is the set of neighbors of node x_k in the tree.
In this notation

$$\begin{aligned} \alpha_k(x_k) &= m_{x_{k-1}, x_k}(x_k) \\ \beta_k(x_k) &= m_{x_{k+1}, x_k}(x_k) \end{aligned}$$

Computing messages

Similar to the recursion we derived for chain we can construct a recursion for a tree

$$m_{x_j,x_k}(x_k) \;=\; \sum_{x_j} \phi_{jk}(x_j, x_k) \prod_{x_l \in n(x_j), l \neq k} m_{x_l,x_j}(x_j)$$

where ϕ_{jk} is the potential on the clique consisting of variables x_j, x_k.

In the case of αs our forward pass always ensured that the α_{k-1} was computed prior to α_k. We need an order for message computations that provides a similar guarantee.

Root-leaf-root

In an undirected tree we can designate any node a root.

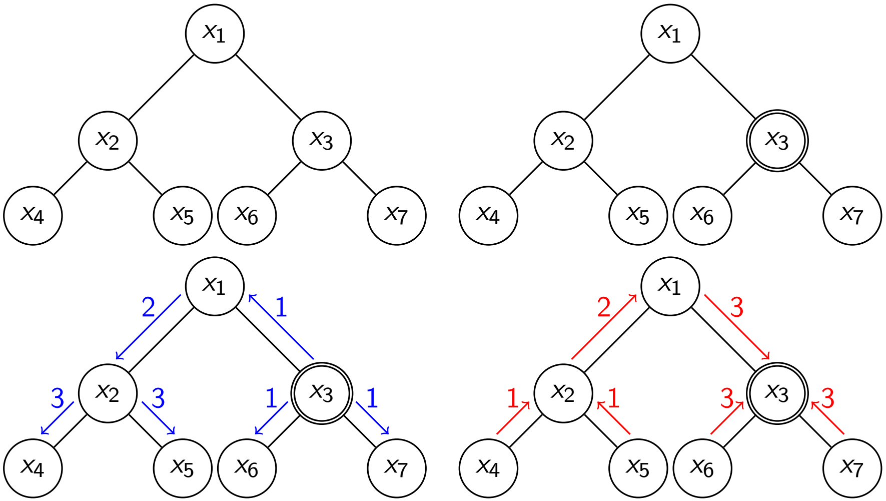
Given a root we can sort nodes by their distance from it.

We can now think of a downward pass and a upward pass.

In an upward pass we compute messages from nodes further away to nodes closer and in the order of the distance from the root.

In an downward pass we simply proceed in the opposite order.

Illustration of message computation



Marginal

As in the case of chain, marginal is product of all incoming messages

$$p(x_k) \propto \prod_{x_j \in n(x_k)} m_{x_j, x_k}(x_k)$$

And again

$$Z = \sum_{x_k} \prod_{x_j \in n(x_k)} m_{x_j, x_k}(x_k)$$

for any x_k so you can check your code for producing consistent Z across all variables.

Incorporating observations

Most of the time you will be interested in computing a marginal distribution *but* conditioned on some set of observed variables x_O.

$$p(x_k | x_O = v_O) = \sum_{x_H} p(x_k, x_H | x_O = v_O)$$

where $k \notin H$ and $H \cap O = \emptyset$ and $\{k\} \cup O \cup H = \{1, \ldots p\}$.

We can incorporate observations directly into our potentials by adjusting them

$$\phi^E(x_j = v_j, x_l = v_l) = \begin{cases} 0, & j \in O, x_j \neq v_j \text{ or } l \in O, x_l \neq v_l \\ \phi(x_j, x_l), & \text{otherwise} \end{cases}$$

thus using potentials ϕ^E ensures that configurations that violate the observations have probability 0.

We did ...

- Learning parameters in a Bayes Net
- Simple learning examples: Naive Bayes
- Generative vs. discriminative learning
- Inference