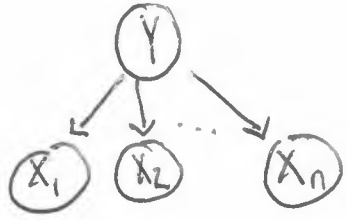
Naive Bayes Classifier

- Family of classification models often used in text classification
- generative
- not Bayesian (but can be made bayesian)
- major assumption: given the class, y, each feature is conditionally indep. of all other features given y

$Y \sim \text{Cat}(\phi)$

$X_j | Y = y \sim P(\ominus)$

j = the j^{th} feature of X



$y \in \{1, \dots, K\}$; $\phi \in [0,1]^K$

$x \in \mathbb{R}^n$

$$P(y | x_1, \dots x_n) = \frac{P(x_1, \dots, x_n | y) P(y)}{P(x_1, \dots, x_n)} = \frac{\overbrace{P(x_1 | y, x_2, \dots) P(x_2 | y, x_3 \dots) \dots P(x_{n-1} | y, x_n) P(x_n | y)}^{\text{chain rule of probability}} P(y)}{P(x_1, \dots, x_n)}$$

$$\overset{(CI)}{=} \frac{P(y) \prod_{j=1}^{n} P(x_j | y)}{P(x_1, \dots, x_n)}$$

Class prediction:

$$\hat{y} = \underset{y \in \mathcal{Y}}{\operatorname{arg\,max}} \left\{ P(y; \hat{\phi}) \prod_{j=1}^{n} P(x_j | y; \hat{\phi}_j) \right\}$$

MLE training (could also use MAP estimate if you want, or go full Bayesian to get a posterior)

Let all parameters be in a set, Θ ← all $x^{(i)} \in \mathbb{R}^n$

$$L(\Theta) = P(\mathcal{D}; \Theta) = P(x^{(1)}, y^{(1)}, x^{(2)}, y^{(2)}, \ldots, x^{(m)}, y^{(m)}) \overset{\text{iid}}{=} \prod_{i=1}^{m} P(x^{(i)}, y^{(i)}) = \prod_{i=1}^{m} P(x^{(i)} | y^{(i)}) P(y^{(i)})$$

$$\overset{\text{(NB)}}{=} \prod_{i=1}^{m} \prod_{j=1}^{n} P(x_j^{(i)} | y^{(i)}) P(y^{(i)}) = \prod_{i=1}^{m} P(y^{(i)}; \phi) \prod_{j=1}^{n} P(x_j^{(i)} | y^{(i)}; \phi_{j|y^{(i)}})$$

⇒ take log and find argmax of Θ, possibly using Lagrange multipliers to make sure $\sum_{k} \phi = 1$

- MLE estimates are very intuitive and do not require iterative solns. (eg. SGD)
  - eg: $\phi_1 = \dfrac{\#\text{ of class } 1 \text{ in data set}}{\#\text{ in data set}}$
- MLEs might benefit from Laplace smoothing, which corresponds to putting a prior on ϕ with most probability mass $\neq 0$

- depending on application, $P(x_j|y)$ could be:
  - Gaussian
  - categorical
  - bernoulli
  - etc...

eg: email classification

- Let $x_j = 0$ denote that the j^{th} word in our dictionary is not in the email and $x_j = 1$ denote that it is.
- $\therefore$ a feature vector for an email can be represented as $x = [0, 1, 1, .. 0, .. 1]^T$ (for example) $\in \{0,1\}^n$

$Y \sim Bern(\phi)$; $\phi \in [0,1]$

$X_j | Y: \not{y} \sim Bern(\phi_{j|y}) \Rightarrow 2 \times (n-1)$ parameters (probs. sum to one) to estimate w/ MLE

Pros

- MLE done in closed form, so fast
- scalable $\Rightarrow$ only requires # parameters $\approx$ # features (i.e., not many parameters to fit)
  - means its high bias, low variance
  - thus, only need small amount of training data (since not too many params to estimate!)

- rather than try to estimate a joint distribution

$P(x_1, x_2, \ldots, x_n, y)$

which is a PDF/PMF in a very high dim. space (which would take many parameters to actually fit and lots of data), the "Naive" assumption allows us to decompose the problem to estimating n 1-D (univariate) PDFs/PMFs.

$(= P(x_1|y) P(x_2|y) \ldots P(x_n|y))$

Cons

- many times cond. independence does not hold (but NB still seems to perform well in many applications)
- high bias model (sometimes too simple) so that other, more complicated models might perform better (eg. RF)