CS 446 Homework 2

William Lee, NetID wl72

Problem 1: Naive Bayes

(a)

i. If $X \perp Y \mid Z$, then

$$P(X, Y \mid Z) = P(X \mid Y, Z)P(Y \mid Z) = P(X \mid Z)P(Y \mid Z).$$

Thus, the factorization holds.

ii. No. Conditional independence does not imply marginal independence. $X \perp Y \mid Z$ does not mean $X \perp Y$, so in general

$$P(X, Y) \neq P(X)P(Y).$$

iii. For d Boolean features and C classes, the number of parameters is

$$d \times C.$$

iv. For Gaussian Naive Bayes, each feature given class c requires a mean μ_{jc} and variance σ_{jc}^2. Hence

$$2dC \quad \text{parameters.}$$

v. The denominator $\sum_v P(Y = v) \prod_k P(X_k \mid Y = v)$ is the same for all classes, so omitting it does not change the arg max.

vi. Yes. We can compute

$$P(X) = \sum_c P(Y = c) \prod_j P(X_j \mid Y = c).$$

(b)

Given conditional independence:

$$P(X_1, X_2, X_3, Y) = P(Y)P(X_1 \mid Y)P(X_2 \mid Y)P(X_3 \mid X_1).$$

MLE estimators:

$$P(Y = 1) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{y^{(i)} = 1\},$$
$$P(X_1 = 1 \mid Y = y) = \frac{\sum_{i=1}^{N} \mathbf{1}\{X_1^{(i)} = 1, y^{(i)} = y\}}{\sum_{i=1}^{N} \mathbf{1}\{y^{(i)} = y\}},$$
$$P(X_3 = 1 \mid Y = y) = \frac{\sum_{i=1}^{N} \mathbf{1}\{X_3^{(i)} = 1, y^{(i)} = y\}}{\sum_{i=1}^{N} \mathbf{1}\{y^{(i)} = y\}}.$$

(c)

We have:

$$X_1 \mid Y = y \sim \mathcal{N}(\mu_{1y}, 1), \quad X_2 \mid Y = y \sim \mathcal{N}(\mu_{2y}, 1), \quad X_3 \mid X_1 = x_1 \sim \mathcal{N}(2x_1, 1).$$

i. MAP rule:

$$y^* = \arg\max_y P(Y = y)P(X_1 \mid Y = y)P(X_2 \mid Y = y)P(X_3 \mid X_1).$$

Since $P(X_3 \mid X_1)$ is independent of Y, it cancels. Classification depends only on X_1, X_2.

ii. For $X^{(a)} = (0.2, 0.7, -10)$ and $X^{(b)} = (0.2, 0.7, 10)$, the likelihood ratio is identical since X_3 cancels. The predicted labels are the same.

Problem 2: Gaussian Naive Bayes

(a)

MLE estimates:

$$\mu_{y,j} = \frac{1}{N_y}\sum_{i:y^{(i)}=y} x_j^{(i)}, \quad \sigma_{y,j}^2 = \frac{1}{N_y}\sum_{i:y^{(i)}=y} \left(x_j^{(i)} - \mu_{y,j}\right)^2.$$

(b)

$$\hat{p} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\{y^{(i)} = 0\}.$$

(c)

Classification rule (log form):

$$\hat{y} = \arg\max_{y} \left[\log p(y) + \sum_{j=1}^{d} \left(-\tfrac{1}{2} \log(2\pi\sigma_{y,j}^2) - \frac{(x_j - \mu_{y,j})^2}{2\sigma_{y,j}^2} \right) \right].$$

(d)

With equal priors and equal variances σ_j^2:

$$\log \frac{P(Y=1 \mid x)}{P(Y=0 \mid x)} = \sum_{j=1}^{d} \frac{\mu_{1j} - \mu_{0j}}{\sigma_j^2} x_j + \text{const.}$$

Thus the decision boundary is linear:

$$y(x) = \begin{cases} 1, & \sum_j w_j x_j > \tau, \\ 0, & \sum_j w_j x_j < \tau, \end{cases}$$

for some w_j, τ.

Problem 3: Logistic Regression

(a)

$$\sigma(-s) = \frac{1}{1+e^{s}} = 1 - \sigma(s).$$

Thus $P(y=-1 \mid x) + P(y=1 \mid x) = 1$.

(b)

$$\sigma'(s) = \frac{e^{-s}}{(1+e^{-s})^2} = \sigma(s)(1-\sigma(s)).$$

(c)

Log-likelihood:

$$\ell(w) = \sum_{i=1}^{N} \log \sigma(y^{(i)} w^\top x^{(i)}).$$

Gradient:

$$\nabla_w \ell(w) = \sum_{i=1}^{N} y^{(i)} x^{(i)} \big(1 - \sigma(y^{(i)} w^\top x^{(i)})\big).$$

(d)

Hessian:

$$H = -\sum_{i=1}^{N} \sigma(z_i)(1 - \sigma(z_i)) x^{(i)} {x^{(i)}}^\top,$$

with $z_i = y^{(i)} w^\top x^{(i)}$.

(e)

For any z,

$$z^\top H z = -\sum_{i=1}^{N} \sigma(z_i)(1 - \sigma(z_i))(z^\top x^{(i)})^2 \leq 0.$$

Thus H is negative semidefinite, so $\ell(w)$ is concave.

(f)

Gradient Descent update:

$$w^{(t+1)} = w^{(t)} + \alpha \sum_{i=1}^{N} y^{(i)} x^{(i)} \big(1 - \sigma(y^{(i)} w^\top x^{(i)})\big).$$

(g)

Newton's Method update:

$$w^{(t+1)} = w^{(t)} - H^{-1} \nabla_w \ell(w).$$

Problem 4: Programming - Written

(a)iii

After applying non-linear feature transformations (such as x_1^2, x_2^2, and x_1x_2), the data becomes linearly separable in the transformed feature space. The logistic regression model can then learn a linear boundary in this higher dimensional space, which corresponds to a non linear boundary in the original space. This explains why the transformed feature model correctly separates the data while the original linear model cannot.

(b)ii

Comparing models with L1, L2, and no regularization:

- **No regularization**: weights can grow large and the model may overfit.
- **L2 regularization (Ridge)**: penalizes large weights smoothly, leading to smaller but nonzero coefficients. The decision boundary is more stable and less sensitive to noise.
- **L1 regularization (Lasso)**: penalizes the sum of absolute values, encouraging many weights to be exactly zero. This performs feature selection, producing sparse solutions.

L1 is preferred when we suspect only a few features are truly relevant. L2 is chosen when all features contribute and we want smooth shrinkage rather than sparsity.

(d)ii

The plot of cost versus updates highlights the trade-offs among the three gradient descent variants:

- **Batch Gradient Descent**: stable and converges smoothly, but requires full passes over the dataset and is computationally heavy per update.
- **Stochastic Gradient Descent (SGD)**: updates after each example, so it converges quickly in practice, but the path is noisy and oscillates around the minimum.

- **Mini-batch Gradient Descent**: balances the two extremes by using small batches (e.g., 32). It improves efficiency over batch gradient descent while reducing the instability of SGD.

Thus, mini batch is often the most practical choice, combining efficiency with stable convergence.