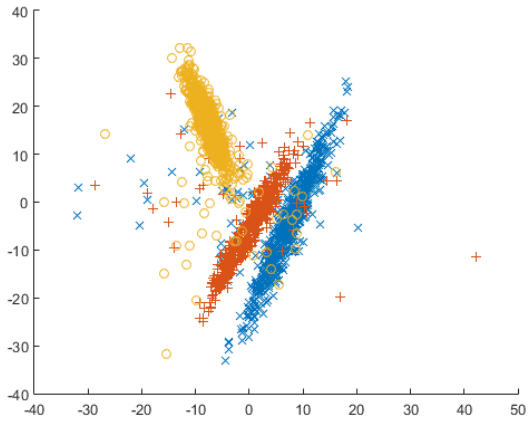
CPSC 440/540 Machine Learning (Jan-Apr 2023)
Assignment 4 – due Thursday April 13th at **11:59pm**

The assignment instructions are the same as for the previous assignments. If you do the assignment with a partner, please only hand in one copy for the group (using the appropriate Gradescope feature).

1 Generative Classifiers with Gaussian Assumption [20 points]

Consider the 3-class classification dataset in this image:



In this dataset, we have 2 features and each colour represents one of the classes. Note that the classes are highly structured: the colours each roughly follow a Gausian distribution plus some noisy samples.

Since we have an idea of what the features look like for each class, we might consider classifying inputs x using a *generative classifier*. In particular, we are going to use Bayes rule to write

$$p(y^i = c \mid x^i, \Theta) = \frac{p(x^i \mid y^i = c, \Theta) \cdot p(y^i = c \mid \Theta)}{p(x^i \mid \Theta)},$$

where Θ represents the parameters of our model. To classify a new example $\tilde{x}^i$, generative classifiers would use

$$\hat{y}^i = \underset{y \in \{1,2,\dots,k\}}{\arg\max}\ p(y^i = c \mid x^i, \Theta) = \underset{y \in \{1,2,\dots,k\}}{\arg\max}\ p(\tilde{x}^i \mid y^i = c, \Theta) p(y^i = c \mid \Theta)$$

where in our case the total number of classes k is 3. Modeling $p(y^i = c \mid \Theta)$ is easy: we can just use a k-state categorical distribution,

$$p(y^i = c \mid \Theta) = \theta_c,$$

where θ_c is a single parameter for class c. The maximum likelihood estimate of θ_c is given by n_c/n, the number of times we have $y^i = c$ (which we've called n_c) divided by the total number of data points n.

Modeling $p(x^i \mid y^i = c, \Theta)$ is the hard part: we need to know the *probability of seeing the feature vector x^i given that we are in class c*. This corresponds to solving a density estimation problem for each of the k possible classes. To make the density estimation problem tractable, we'll assume that the distribution of x^i given that $y^i = c$ is given by a $\mathcal{N}(\mu_c, \Sigma_c)$ Gaussian distribution for a class-specific μ_c and Σ_c,

$$p(x^i \mid y^i = c, \Theta) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma_c|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x^i - \mu_c)^\top \Sigma_c^{-1}(x^i - \mu_c)\right).$$

Since we are distinguishing between the probability under k different Gaussians to make our classification, this is called *Gaussian discriminant analysis* (GDA). In the special case where we have a constant $\Sigma_c = \Sigma$ across all classes it is known as *linear discriminant analysis* (LDA) since it leads to a linear classifier between any two classes (while the region of space assigned to each class forms a convex polyhedron, as in k-means clustering and softmax classification). Another common restriction on the Σ_c is that they are diagonal matrices, since this only requires $O(d)$ parameters instead of $O(d^2)$ (corresponding to assuming that the features are independent univariate Gaussians given the class label). Given a dataset $\mathcal{D} = \{(x^i, y^i)\}_{i=1}^n$, where $x^i \in \mathbb{R}^d$ and $y^i \in \{1, \dots, k\}$, the maximum likelihood estimate (MLE) for the μ_c and Σ_c in the GDA model is the solution to

$$\underset{\mu_1,\mu_2,\dots,\mu_k,\Sigma_1,\Sigma_2,\dots,\Sigma_k}{\arg\max} \prod_{i=1}^{n} p(x^i \mid y^i, \mu_{y^i}, \Sigma_{y^i}).$$

This means that the negative log-likelihood will be equal to

$$\begin{aligned} -\log p(X \mid y, \Theta) &= -\sum_{i=1}^{n} \log p(x^i \mid y^i, \mu_{y^i}, \Sigma_{y^i}) \\ &= \sum_{i=1}^{n} \frac{1}{2}(x^i - \mu_{y^i})^\top \Sigma_{y^i}^{-1}(x^i - \mu_{y^i}) + \frac{1}{2}\sum_{i=1}^{n} \log|\Sigma_{y^i}| + \text{const.} \end{aligned}$$

In class we stated the MLE for this model under the assumption that we use full covariance matrices and that each class has its own covariance.

(1.1) [5 points] Derive the MLE for the GDA model under the assumption of *common diagonal covariance* matrices, $\Sigma_c = D$. Note that this means there are d total parameters for the model's covariances (plus kd parameters in each class's separate mean, μ_c).

Hint: You might be able to significantly simply notation if you write something like $\sum_{i:y^i=c}$, or define some shortcut notation like $\sum_{i \in y_c}$ to mean the same thing; just make sure it's clear. You also might want to use $n_c = \sum_{i \in y_c} 1$ to be the number of cases where $y^i = c$.

Hint: The determinant of a diagonal matrix is the product of its diagonal entries. The inverse is the diagonal matrix where each diagonal entry is inverted.

Answer: TODO

(1.2) [5 points] Derive the MLE for the GDA model under the assumption of *individual scaled-identity* matrices, $\Sigma_c = \sigma_c^2 I$; here there are k total covariance parameters, one per class.

Answer: TODO

(1.3) [5 points] When you run `main.py gda` it loads a variant of the dataset in the figure that has 12 features and 10 classes. This data has been split up into a training and test set, and the code fits a k-nearest neighbour classifier to the training set then reports the accuracy on the test data (around $\sim 63\%$ test error). The k-nearest neighbour model does poorly here since it doesn't take into account the Gaussian-like structure in feature space for each class label.

Fill in the class `GDA` to fit a GDA model to this dataset, using the MLE with individual *full* covariance matrices. Hand in your code, and report the test set accuracy.

Hint: You can use the result from class about the MLE for this model.

Hint: You probably want to handle everything "in log space" for your computation.

Answer: TODO

(1.4) [5 points] The data here has some outliers. Let's try to replace the Gaussian distribution of the previous solution with a more robust multivariate t distribution. Unfortunately this distribution doesn't have

a closed-form MLE, but `student_t.MultivariateT` implements numerical optimization. Fill in the class `generative_classifiers.TDA`, called by `main.py tda`, to fit a generative model based on the multivariate t distribution. Report your test accuracy and hand in your code.

Hint: These take a little longer to fit; you might want to print some kind of status update as you go to make sure it's happening. The `tqdm` package would be a very nice, fancy version of that....

Answer: TODO

2 Discrete Markov Chain Inference [25 points]

The function *example_markov.jl* loads the initial state probabilities and transition probabilities for a Markov chain model,

$$p(x_1, x_2, \ldots, x_d) = p(x_1) \prod_{j=2}^{d} p(x_j \mid x_{j-1}),$$

corresponding to the "grad student Markov chain" from class.

Note that we're assuming the Markov chain starts at time 1; since Python indexing starts from time 0, be careful.

(2.1) [3 points] Write a function, `MarkovChain.sample`, that uses ancestral sampling to sample sequences x from this Markov chain of length d. Hand in this code, and report the univariate marginal probabilities for time 50 using a Monte Carlo estimate based on 10,000 samples (from `main.py mc-sample`).

Hint: `rng.choice` might be helpful.

Answer: TODO

(2.2) [3 points] Write a function, `MarkovChain.marginals`, that uses the CK equations to compute the exact univariate marginals up to a given time d. Use `main.py mc-marginals` to find the exact marginals at time 50. Hand in this code, report the exact univariate marginals at time 50 (to three decimal places), and report how this differs from the marginals in the previous question.

Answer: TODO

(2.3) [2 points] What is the state c with highest marginal probability, $p(x_j = c)$, for each time j?

Answer: TODO

(2.4) [5 points] Write a function, `MarkovChain.mode`, that uses the Viterbi decoding algorithm for Markov chains to find the optimal decoding up to a time d. Hand in this code and report the optimal decoding of the Markov chain up to time 50, and up to 100.

Answer: TODO

(2.5) [2 points] Report all the univariate conditional probabilities at time 50 if the student starts in grad school, $p(x_{50} = c \mid x_1 = 2)$, for all c.

Hint: you can do this by changing the input to the CK equations.

Answer: TODO

(2.6) [3 points] Report for all c the univariate conditional probabilities $p(x_3 = c \mid x_6 = 5)$ ("where you were likely to be 3 years after graduation, if you ended up in academia after 6 years") obtained using a Monte Carlo estimate based on 10,000 samples and rejection sampling. Also report the number of samples accepted from the 10,000 samples.

Answer: TODO

(2.7) [5 points] Give code implementing a dynamic programming approach to exactly compute $p(x_3 = c \mid x_6 = 5)$, and report the exact values for all c.

Answer: TODO

(2.8) [2 points] Why is $p(x_j = 6 \mid x_{20} = 5)$ equal to zero for all j less than 20?

Answer: TODO

3 Markov Chain Monte Carlo for Bayesian Logistic Regression [10 points]

`main.py mcmc-blogreg` loads a set of images of '2' and '3' digits. It then runs the Metropolis MCMC algorithm to try to generate samples from the posterior over w, in a logistic regression model with a Gaussian prior. Once the samples are generated, it makes a histogram of the samples for several of the variables.[1]

(3.1) [3 points] Why might the samples coming from the Metropolis algorithm as run here not give a good approximation to the posterior?

Answer: TODO

(3.2) [4 points] Modify the proposal used by the demo to $\hat{w} \sim \mathcal{N}(w, (1/100)I)$ instead of $\hat{w} \sim \mathcal{N}(w, I)$. Hand in your code and the updated histogram plot.

Answer: TODO

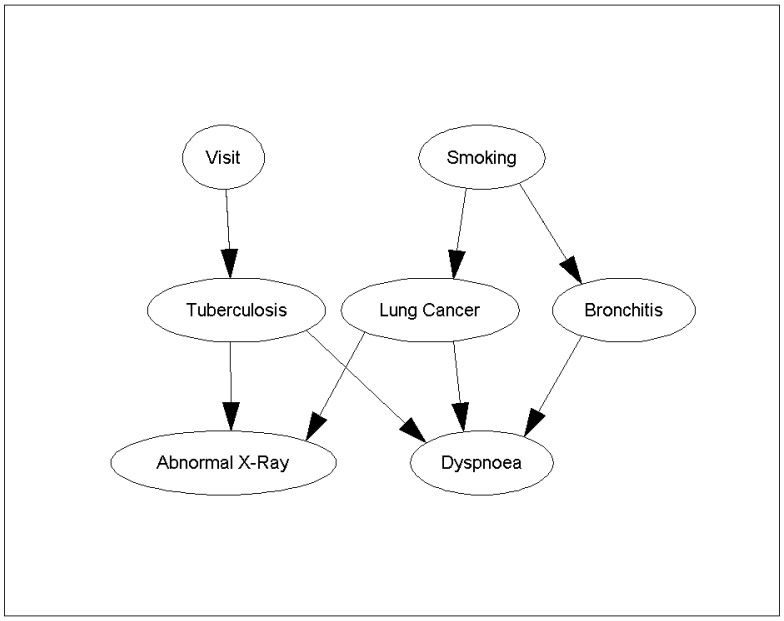
(3.3) [3 points] Modify the proposal to use $\hat{w} \sim \mathcal{N}(w, (1/10000)I)$. Do you think this performs better or worse than the previous choice? (Briefly explain.)

Answer: TODO

[1] The "positive" variables are some of the positive weights when you fit an L2-regularized logistic regression model to the this data. The "negative" variables are some of the negative regression weights in that model, and the "neutral" ones are set to 0 in that model.

4 D-Separation [15 points]

Consider the DAG model below, for distinguishing between different causes of shortness-of-breath (dyspnoea) and the causes of an abnormal lung x-ray, while modelling potential causes of these diseases too (whether the person is a smoker or had a 'visit' to a country with a high degree to tuberculosis).



We'll assume that the distribution over the variables is "faithful" to the graph, meaning that variables are conditionally independent if and only if the graph structure implies their independence. Under this assumption, say why each of the following conditional independence statements is true or false, along with a very brief justification:

Each of these is worth [1.5 points].

(4.1) (Smoking) $\perp\!\!\!\perp$ (Bronchitis).

Answer: TODO

(4.2) (Smoking) $\perp\!\!\!\perp$ (Dyspnoea).

Answer: TODO

(4.3) (Tuberculosis) $\perp\!\!\!\perp$ (Lung Cancer).

Answer: TODO

(4.4) (Abnormal X-Ray) $\perp\!\!\!\perp$ (Visit) | (Tuberculosis).

Answer: TODO

(4.5) (Bronchitis) $\perp\!\!\!\perp$ (Lung Cancer) | (Smoking).

Answer: TODO

(4.6) (Abnormal X-Ray) ⫫ (Dyspnoea).

Answer: TODO

(4.7) (Tuberculosis) ⫫ (Lung Cancer) | (Dyspnoea).

Answer: TODO

(4.8) (Visit,Smoking) ⫫ (Abnormal X-Ray, Dyspnoea) | (Tuberculosis, Lung Cancer, Bronchitis)

Answer: TODO

(4.9) (Tuberculosis) ⫫ (Bronchitis) | (Abnormal X-Ray).

Answer: TODO

(4.10) (Smoking) ⫫ (Visit) | (Dyspnoea).

Answer: TODO

5 Very-Short Answer Questions [30 points]

Each of these is worth [2 points].

(5.1) How does the affine property allow us to sample from multivariate Gaussians?

Answer: TODO

(5.2) What is the relationship between using a product of Gaussians and using a multivariate Gaussian?

Answer: TODO

(5.3) With a Gaussian likelihood, a Gaussian prior for its mean, and a fixed covariance, how do we know that the posterior predictive will also be a Gaussian?

Answer: TODO

(5.4) How do the sparsity patterns of the covariance and precision matrix in a multivariate Gaussian relate to independence and conditional independence in the models?

Answer: TODO

(5.5) In linear discriminant analysis, why might we not assign a point to its closest mean?

Answer: TODO

(5.6) The maximum of the posterior predictive in Bayesian linear regression gives the same prediction as if we used the MAP estimate. What is a reason we might use Bayesian linear regression anyways?

Answer: TODO

(5.7) What is the key difference between Monte Carlo approximations and variational approximations?

Answer: TODO

(5.8) What is a key advantage of "end-to-end" training of computer vision system?

Answer: TODO

(5.9) What is $\nabla_\theta \log Z(\theta)$ for an exponential family in canonical form? (Give an interpretable form in terms of the sufficient statistics; no need to derive it, just the answer.)

Answer: TODO

(5.10) What is the difference between computing marginals and computing the stationary distribution of a Markov chain.

Answer: TODO

(5.11) What is the cost of generating a sample from a Markov chain of length d with k possible states for each time? What is the cost of decoding?

Answer: TODO

(5.12) In what setting is it unnecessary to include the q function in the Metropolis-Hastings acceptance probability?

Answer: TODO

(5.13) What is "explaining away"?

Answer: TODO

(5.14) If two variables are not d-separated, are they necessarily dependent? If two variables are d-separated, are they necessarily independent?

Answer: TODO

(5.15) Describe how we could use ancestral sampling to sample from the joint density over x and y defined by a Gaussian discriminant analysis model.

Answer: TODO