

# CPSC 540 Assignment 3 (due February 27)

## Density Estimation and Project Proposal

### 1 Discrete and Gaussian Variables

#### 1.1 MLE for General Discrete Distribution

Consider a density estimation task, where we have two variables ( $d = 2$ ) that can each take one of  $k$  discrete values. For example, we could have

$$X = \begin{bmatrix} 1 & 3 \\ 4 & 2 \\ k & 3 \\ 1 & k-1 \end{bmatrix}.$$

The likelihood for example  $x^i$  under a general discrete distribution would be

$$p(x_1^i, x_2^i | \Theta) = \theta_{x_1^i, x_2^i},$$

where  $\theta_{c_1, c_2}$  gives the probability of  $x_1$  being in state  $c_1$  and  $x_2$  being in state  $c_2$ , for all the  $k^2$  combinations of the two variables. In order for this to define a valid probability, we need all elements  $\theta_{c_1, c_2}$  to be non-negative and they must sum to one,  $\sum_{c_1=1}^k \sum_{c_2=1}^k \theta_{c_1, c_2} = 1$ .

1. Given  $n$  training examples, [derive the MLE for the  \$k^2\$  elements of  \$\Theta\$](#) .
2. Because of the sum-to-1 constraint, there are only  $(k^2 - 1)$  degrees of freedom in the discrete distribution, and not  $k^2$ . [Derive the MLE for this distribution assuming that](#)

$$\theta_{k, k} = 1 - \sum_{c_1=1}^k \sum_{c_2=1}^k \mathcal{I}[c_1 \neq k, c_2 \neq k] \theta_{c_1, c_2},$$

so that the distribution only has  $(k^2 - 1)$  parameters.

3. If we had separate parameter  $\theta_{c_1}$  and  $\theta_{c_2}$  for each variables, a reasonable choice of a prior would be a product of Dirichlet distributions,

$$p(\theta_{c_1}, \theta_{c_2}) \propto \theta_{c_1}^{\alpha_{c_1} - 1} \theta_{c_2}^{\alpha_{c_2} - 1}.$$

For the general discrete distribution, a prior encoding the same assumptions would be

$$p(\theta_{c_1, c_2}) \propto \theta_{c_1, c_2}^{\alpha_{c_1} + \alpha_{c_2} - 2}.$$

[Derive the MAP estimate under this prior \(assuming we use  \$k^2\$  variables to parameterize  \$\Theta\$ \)](#)

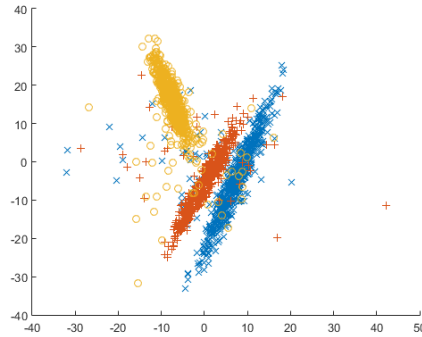
Hint: it is convenient to write the likelihood for an example  $i$  in the form

$$p(x^i | \Theta) = \prod_{c \in [k]^2} \theta_c^{\mathcal{I}[x^i=c]},$$

where  $c$  is a vector containing  $(c_1, c_2)$ ,  $[x^i = c]$  evaluates to 1 if all elements are equal, and  $[k]^2$  is all ordered pairs  $(c_1, c_2)$ . You can use the Lagrangian to enforce the sum-to-1 constraint on the log-likelihood, and you may find it convenient to define  $N_c = \sum_{i=1}^n \mathcal{I}[x^i = c]$ .

## 1.2 Generative Classifiers with Gaussian Assumption

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 Gaussian 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

$$p(y = c|x, \Theta) = \frac{p(x|y = c, \Theta) \cdot p(y = c|\Theta)}{p(x|\Theta)},$$

where  $\Theta$  represents the parameters of our model. To classify a new example  $\hat{x}$ , generative classifiers would use

$$\hat{y} = \arg \max_{y \in \{1, 2, \dots, k\}} p(\hat{x}|y = c, \Theta)p(y = c|\Theta),$$

where in our case the total number of classes  $k$  is 3 (The denominator  $p(\hat{x}|\Theta)$  is irrelevant to the classification since it is the same for all  $y$ .) Modeling  $p(y = c|\Theta)$  is easy: we can just use a  $k$ -state categorical distribution,

$$p(y = c|\Theta) = \theta_c,$$

where  $\theta_c$  is a single parameter for class  $c$ . The maximum likelihood estimate of  $\theta_c$  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  $p(x|y = c, \Theta)$  is the hard part: we need to know the *probability of seeing the feature vector  $x$  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  $x$  given that  $y = c$  is given by a  $\mathcal{N}(\mu_c, \Sigma_c)$  Gaussian distribution for a class-specific  $\mu_c$  and  $\Sigma_c$ ,

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

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). Another common restriction on the  $\Sigma_c$  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  $\mu_c$  and  $\Sigma_c$  in the GDA model is the solution to

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

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

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

1. Derive the MLE for the GDA model under the assumption of *common diagonal covariance matrices*,  $\Sigma_c = D$  ( $d$  parameters). (Each class will have its own mean  $\mu_c$ .)
2. Derive the MLE for the GDA model under the assumption of *individual scale-identity matrices*,  $\Sigma_c = \sigma_c^2 I$  ( $k$  parameters).
3. It's painful to derive these from scratch, but you should be able to see a pattern that would allow other common restrictions. Without deriving the result from scratch (hopefully), [give the MLE for the case of individual full covariance matrices](#),  $\Sigma_c$  ( $O(kd^2)$  parameters).
4. When you run `example_generative` 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 ( $\sim 36\%$ ). 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. Write a function `generativeGaussian` that fits a GDA model to this dataset (using individual full covariance matrices). [Hand in the function and report the test set accuracy](#).
5. In this question we would like to replace the Gaussian distribution of the previous problem with the more robust multivariate-t distribution so that it isn't influenced as much by the noisy data. Unlike the previous case, we don't have a closed-form solution for the parameters. However, if you run `example_tdist` it generates random noisy data and fits a multivariate-t model (you will need to add the `minFunc` directory to the Matlab path for the demo to work). By using the `multivariateT` model, write a new function `generativeStudent` that implements a generative model that is based on the multivariate-t distribution instead of the Gaussian distribution. [Report the test accuracy with this model](#).

Hints: you will be able to substantially simplify the notation in parts 1-3 if you use the notation  $\sum_{i \in y_c}$  to mean the sum over all values  $i$  where  $y^i = c$ . Similarly, you can use  $n_c$  to denote the number of cases where  $y_i = c$ , so that we have  $\sum_{i \in y_c} 1 = n_c$ . Note that the determinant of a diagonal matrix is the product of the diagonal entries, and the inverse of a diagonal matrix is a diagonal matrix with the reciprocals of the original matrix along the diagonal. For part three you can use the result from class regarding the MLE of a general multivariate Gaussian. You may find it helpful to use the included `logdet.m` function to compute the log-determinant in more numerically-stable way.

## 1.3 Self-Conjugacy for the Mean Parameter

If  $x$  is distributed according to a Gaussian with mean  $\mu$ ,

$$x \sim \mathcal{N}(\mu, \sigma^2),$$

and we assume that  $\mu$  itself is distributed according to a Gaussian

$$\mu \sim \mathcal{N}(\alpha, \gamma^2),$$

then the posterior  $\mu|x$  also follows a Gaussian distribution.<sup>1</sup> [Derive the form of the \(Gaussian\) distribution for  \$p\(\mu|x, \alpha, \sigma^2, \gamma^2\)\$ .](#)

Hints: Use Bayes rule and use the  $\propto$  sign to get rid of factors that don't depend on  $\mu$ . You can “complete the square” to make the product look like a Gaussian distribution, e.g. when you have  $\exp(ax^2 - bx + \text{const})$  you can factor out an  $a$  and add/subtract  $(b/2a)^2$  to re-write it as

$$\begin{aligned} \exp(ax^2 - bx + \text{const}) &\propto \exp(ax^2 - bx) = \exp(a(x^2 - (b/a)x)) \\ &\propto \exp(a(x^2 - (b/a)x + (b/2a)^2)) = \exp(a(x - (b/2a))^2). \end{aligned}$$

Note that multiplying by factors that do not depend on  $\mu$  within the exponent does not change the distribution. In this question you will want to complete the square to get the distribution on  $\mu$ , rather than  $x$ . You may find it easier to solve this problem if you parameterize the Gaussians in terms of their ‘precision’ parameters (e.g.,  $\lambda = 1/\sigma^2$ ,  $\lambda_0 = 1/\gamma^2$ ) rather than their variances  $\sigma^2$  and  $\gamma^2$ .

## 2 Mixture Models and Expectation Maximization

### 2.1 Semi-Supervised Gaussian Discriminant Analysis

Consider fitting a GDA model where some of the  $y^i$  values are missing at random. In particular, let's assume we have  $n$  labeled examples  $(x^i, y^i)$  and then another  $t$  unlabeled examples  $(x^i)$ . This is a special case of *semi-supervised learning*, and fitting generative models with EM is one of the oldest semi-supervised learning techniques. When the classes exhibit clear structure in the feature space, it can be very effective even if the number of labeled examples is very small.

1. [Derive the EM update for fitting the parameters of a GDA model \(with individual full covariance matrices\) in the semi-supervised setting where we have  \$n\$  labeled examples and  \$t\$  unlabeled examples.](#)
2. If you run the demo `example_SSL`, it will load a variant of the dataset from the previous question, but where the number of labeled examples is small and a large number of unlabeled examples are available. The demo first fits a KNN model and then a generative Gaussian model (once you are finished Question 1). Because the number of labeled examples is quite small, the performance is worse than in Question 2. Write a function `generativeGaussianSSL` that fits the generative Gaussian model of the previous question using EM to incorporate the unlabeled data. [Hand in the function and report the test error when training on the full dataset.](#)
3. Repeat the previous part, but using the “hard”-EM algorithm where we explicitly classify all the unlabeled examples. [How does this change the performance and the number of iterations?](#)

---

<sup>1</sup>We say that the Gaussian distribution is the ‘conjugate prior’ for the Gaussian mean parameter (we’ll formally discuss conjugate priors later in the course). Another reason the Gaussian distribution is important is that it is the only (non-trivial) continuous distribution that has this ‘self-conjugacy’ property.

Hint: for the first question most of the work has been done for you in the EM notes on the course webpage. You can use the result (\*\*) and the update of  $\theta_c$  from those notes, but you will need to work out the update of the parameters of the Gaussian distribution  $p(x^i|y^i, \Theta)$ .

Hint: for the second question, although EM often leads to simple updates, implementing them correctly can often be a pain. One way to help debug your code is to compute the observed-data log-likelihood after every iteration. If this number goes down, then you know your implementation has a problem. You can also test your updates of sets of variables in this way too. For example, if you hold the  $\mu_c$  and  $\Sigma_c$  fixed and only update the  $\theta_c$ , then the log-likelihood should not go down. In this way, you can test each of combinations of updates on their to make sure they are correct.

## 2.2 Mixture of Bernoullis

The function `example_Bernoulli` loads a binarized version of the MNIST dataset and fits a density model that uses an independent Bernoulli to model each feature. It reports the average NLL on the test data and shows 4 samples generated from the model. Unfortunately, the test NLL is infinity and the samples look terrible.

1. To address the problem that the average NLL is infinity, modify the `densityBernoulli` function to implement Laplace smoothing based on an extra argument  $\alpha$ . [Hand in the code and report the average NLL with  \$\alpha = 1\$ .](#)
2. Write a new function implementing the mixture of Bernoullis model with Laplace smoothing of the  $\theta$  values (note that Laplace smoothing only change the M-step). [Hand in the code and report the average NLL with  \$\alpha = 1\$  and  \$k = 10\$  for a particular run of the algorithm, as well as 4 samples from the model and 4 of the cluster images.](#)

## 3 Project Proposal

For the final of this assignment, you must [submit a project proposal](#) for your course project. The proposal should be a maximum of 2 pages (and 1 page or half of a page is ok if you can describe your plan concisely). The proposal should be written for me and the TAs, so you don't need to introduce any ML background but you will need to introduce non-ML topics. The projects must be done in groups of 2-3. If you are doing your assignment in a group that is different from your project group, only 1 group member should include the proposal as part of their submission (we'll do the merge across assignments, and this means that assignments could have multiple proposals). Please state clearly who is involved with each project proposal.

There is quite a bit of flexibility in terms of the type of project you do, as I believe there are many ways that people can make valuable contributions to research. However, note that ultimately the project will have three parts:

1. A very short paper review summarizing the pros and cons of a particular paper on the topic (due with Assignment 4).
2. A short literature review summarizing at least 10 papers on a particular topic (due with Assignment 5).
3. A final report containing at most 6 pages of text (the actual document can be longer due to figures, tables, references, and proofs) that emphasizes a particular "contribution" (i.e., what the project has added to the world).

The reason for this, even though it's strange for some possible projects, is that this is the standard way that results are communicated to the research community.

The three main ingredients of the project proposal are:

1. What problem you are focusing on.
2. What you plan to do.
3. What will be the “contribution”.

Also, note that for the course project that negative results (i.e., we tried something that we thought we would work in a particular setting but it didn’t work) are acceptable.

Here are some standard project “templates” that you might want to follow:

- **Application bake-off:** you pick a specific application (from your research, personal interests, or maybe from Kaggle) or a small number of related applications, and try out a bunch of techniques (e.g., random forests vs. logistic regression vs. generative models). In this case, the contribution would be showing that some methods work better than others for this specific application (or your contribution could be that everything works equally well/badly).
- **New application:** you pick an application where ML methods have previously applied, and you test out whether ML methods are effective for the task. In this case, the contribution would be knowing whether ML is suitable for the task.
- **Scaling up:** you pick a specific machine learning technique, and you try to figure out how to make it run faster or on larger datasets (for example, how do we apply kernel methods when  $n$  is very large). In this case, the contribution would be the new technique and an evaluation of its performance.
- **Improving performance:** you pick a specific machine learning technique, and try to extend it in some way to improve its performance (for example, how can we efficiently use non-linearity within graphical models). In this case, the contribution would be the new technique and an evaluation of its performance.
- **Generalization to new setting:** you pick a specific machine learning technique, and try to extend it to a new setting (for example, making a graphical-model version of random forests). In this case, the contribution would be the new technique and an evaluation of its performance.
- **Perspective paper:** you pick a specific topic in ML, read a larger number of papers on the topic, then write a report summarizing what has been done on the topic and what are the most promising directions of future work. In this case, the contribution would be your summary of the relationships between the existing works, and your insights about where the field is going.
- **Coding project:** you pick a specific method or set of methods (like independent component analysis), and build an implementation of them. In this case, the contribution would be the implementation itself.
- **Theory:** you pick a theoretical topic (like the variance of cross-validation), read what has been done about it, and try to prove a new result (usually by relaxing existing assumptions or adding new assumptions). The contribution could be a new analysis of an existing method, or why some approaches to analyzing the method will not work.

The above are just suggestions, and many projects will mix several of these templates together, but if you are having trouble getting going then it’s best to stick with one of the above templates. Also note that the above includes topics not covered in the course (like random forests), so there is flexibility in the topic, but the topic should be closely-related to ML.

This question is mandatory but will not be formally marked: it’s just a sanity check that you have at least one project idea that fits within the scope of 540 course project, and it’s an excuse for you to allocate some time to thinking about the project. Also, there is flexibility in the choice of project topics even after the proposal: if you want to explore different topics you can ultimately choose to do a project that is unrelated

to the one in your proposal/paper-review/literature-review, although it will likely be easier to do all 4 parts on the same topic.