appendix B
Answers to exercises

2.1 Derive full conditionals $p(x_A|x_B)$ for multivariate Gaussian distribution, where A and B are subsets of $x_1, x_2, ..., x_n$ of jointly Gaussian random variables.

Let's partition our vector x into two subsets, x_A and x_B, and then we can write down the mean and covariance matrices in block form:

$$\mu = \begin{pmatrix} \mu_A \\ \mu_B \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{pmatrix}$$

To compute the conditional distributions, we get the following result:

$$\begin{aligned} p(x_A|x_B) &= N\left(x_A|\mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(x_B - \mu_B), \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}\right) \\ p(x_B|x_A) &= N\left(x_B|\mu_B + \Sigma_{BA}\Sigma_{AA}^{-1}(x_A - \mu_A), \Sigma_{BB} - \Sigma_{BA}\Sigma_{AA}^{-1}\Sigma_{AB}\right) \end{aligned}$$

2.2 Derive marginals $p(x_A)$ and $p(x_B)$ for multivariate Gaussian distribution, where A and B are subsets of $x_1, x_2, ..., x_n$ of jointly Gaussian random variables.

Let's partition our vector x into two subsets, x_A and x_B, and then we can write down the mean and covariance matrices in block form:

$$\mu = \begin{pmatrix} \mu_A \\ \mu_B \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{pmatrix}$$

To compute the marginals, we simply read off corresponding rows and columns from mean and covariance:

$$\begin{aligned} p(x_A) &= \int N(x|\mu, \Sigma)dx_B = N(x_A|\mu_A, \Sigma_{AA}) \\ p(x_B) &= \int N(x|\mu, \Sigma)dx_B = N(x_B|\mu_B, \Sigma_{BB}) \end{aligned}$$

2.3 Let $y \sim N(\mu, \Sigma)$, where $\Sigma = \mathrm{L}\mathrm{L}^{\mathrm{T}}$. Show that you can get samples y as follows: $x \sim N(0, I)$; $y = \mathrm{L}x + \mu$.

Let $y = \mathrm{L}x + \mu$. Then, $\mathrm{E}[y] = \mathrm{L}\mathrm{E}[x] + \mu = 0 + \mu = \mu$. Similarly, $\mathrm{cov}(y) = \mathrm{L}\ \mathrm{cov}(x)\mathrm{L}^{\mathrm{T}} + 0 = LIL^{\mathrm{T}} = LL^{\mathrm{T}} = \Sigma$. Since y is an affine transformation of a Gaussian RV, y is also Gaussian distributed as $y \sim N(\mu, \Sigma)$.

3.1 Compute KL divergence between two univariate Guassians: $p(x) \sim N(\mu_1, \sigma_1^2)$ and $q(x) \sim N(\mu_2, \sigma_2^2)$.

$$\begin{aligned} KL(p||q) &= -\int p(x) \log q(x)dx + \int p(x) \log p(x)dx \\ &= \frac{1}{2} \log \left(2\pi\sigma_2^2\right) + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} \\ &\quad - \frac{1}{2}\left(1 + \log 2\pi\sigma_1^2\right) \\ &= \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \end{aligned}$$

3.2 Compute $E[X]$, $\mathrm{Var}(X)$, and $\mathrm{H}(X)$ for a Bernoulli distribution.

A Bernoulli distribution is defined as $p(x) = p^x(1-p)^{1-x}$, where $x \in \{0, 1\}$. By the definitions of mean, variance, and entropy, we can compute the following quantities:

$$E[X] = \sum_{x \in \{0,1\}} p_x(x)x = p_x(0) \times 0 + p_x(1) \times 1 = p$$

$$\begin{aligned} var(X) &= E[X^2] - E[X]^2 = \sum_{x \in \{0,1\}} p_x(x)x^2 - p^2 \\ &= p_x(0) \times 0^2 + p_x(1) \times 1^2 - p^2 = p(1-p) \end{aligned}$$

$$\begin{aligned} H(X) &= -\sum_{x \in \{0,1\}} p_x(x) \log p_x(x) \\ &= -p_x(0) \log p_x(0) - p_x(1) \log p_x(1) \\ &= -(1-p) \log(1-p) - p \log p \end{aligned}$$

3.3 Derive the mean, mode, and variance of a Beta(a,b) distribution.

The beta distribution has support over the interval [0, 1] and is defined as follows:

$$\text{Beta}(x|a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1}$$

We have the following expressions for the mean, mode, and variance of the Beta distribution:

$$E[X] = \frac{a}{a+b} \qquad \text{mode}[X] = \frac{a-1}{a+b-2} \qquad \text{VAR}(X) = \frac{ab}{(a+b)^2(a+b+1)}$$

4.1 Prove the following binomial identity: $C(n,k) = C(n-1,k-1) + C(n-1,k)$.

We'll be using an algebraic proof. By definition, expanding the left-hand side, we have the following:

$$\begin{aligned}
\binom{n-1}{k-1} + \binom{n-1}{k} &= \frac{(n-1)!}{(n-k)!\cdot(k-1)!} + \frac{(n-1)!}{(n-k-1)!\cdot(k)!} \\
&= \frac{(n-1)!}{(n-k-1)!\cdot(k-1)!} \cdot \left(\frac{1}{n-k} + \frac{1}{k}\right) \\
&= \frac{n!}{(n-k)!\cdot k!} \\
&= \binom{n}{k}
\end{aligned}$$

4.2 Derive the Gibbs inequality $H(p,q) \geq H(q)$, where $H(p,q) = -\sum p(x) \log q(x)$ is the cross-entropy and $H(q) = -\sum q(x) \log q(x)$ is the entropy.

First, we show that KL divergence is nonnegative:

$$\begin{aligned}
-KL(p||q) &= -\sum_x p(x) \log \frac{p(x)}{q(x)} dx = \sum_x p(x) \log \frac{q(x)}{p(x)} dx \\
&\leq \log \sum_x p(x) \frac{q(x)}{p(x)} dx = \log \sum_x q(x) \\
&\leq \log 1 = 0
\end{aligned}$$

Using the previously derived property, we can write down the Gibbs inequality as follows:

$$\begin{aligned}
KL(p||q) &\geq 0 \\
\sum_x p(x) \log p(x) - \sum_x p(x) \log q(x) &\geq 0 \\
-H(p) + H(p,q) &\geq 0 \\
H(p,q) &\geq H(p)
\end{aligned}$$

4.3 Use Jensen's inequality with $f(x) = \log(x)$ to prove the AM $\geq$ GM inequality.

Since $f(x) = \log(x)$ is a concave function, we can write the Jensen's inequality as follows:

$$\begin{aligned} f(E[x]) &\geq E[f(x)] \\ \log(E[x]) &\geq E[\log(x)] \\ \log(\frac{x_1 + \cdots + x_n}{n}) &\geq \frac{\log x_1 + \cdots + \log x_n}{n} \\ \log(\frac{x_1 + \cdots + x_n}{n}) &\geq \log(x_1 \times \cdots \times x_n)^{\frac{1}{n}} \\ AM &\geq GM \end{aligned}$$

4.4 Prove that $I(x;y) = H(x) - H(x|y) = H(y) - H(y|x)$.

By the definition of mutual information, we have the following:

$$\begin{aligned} I(X;Y) &= \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \\ &= E_{p(x,y)}\left[\log \frac{p(x|y)}{p(x)}\right] = E_{p(x,y)}\left[\log \frac{p(y|x)}{p(y)}\right] \\ &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \end{aligned}$$

5.1 Given a data point $y \in \mathrm{R}^{\mathrm{d}}$ and a hyperplane $\theta \cdot x + \theta_0 = 0$, compute the Euclidean distance from the point to the hyperplane.

Let x be a point on the hyperplane—in other words, ensure it satisfies the equation $\theta \cdot x + \theta_0 = 0$. To find the Euclidean distance from the point to the hyperplane, we construct a vector $y - x$ and project it onto the unique vector perpendicular to the plane. Thus, we have the following:

$$\begin{aligned} d &= \|\mathrm{proj}_\theta(y - x)\| \\ &= \left\| \frac{(y-x)\cdot\theta}{\theta\cdot\theta}\theta \right\| = |y\cdot\theta - x\cdot\theta| \frac{\|\theta\|}{\|\theta\|^2} \\ &= \frac{|y\cdot\theta - x\cdot\theta|}{\|\theta\|} = \frac{|y\cdot\theta + \theta_0|}{\|\theta\|} \end{aligned}$$

5.2 Given a primal linear program (LP) min $c^{\mathrm{T}} x$ subject to $Ax <= b$, x >= 0, write down the dual version of the LP.

Each variable in the primal LP corresponds to a constraint in the dual LP, and vice versa. The primal is a minimization problem, so the dual is a maximization problem. The primal constraints are less than or equal, so the dual variables are greater than or

equal. The primal variables are nonnegative, so the dual constraints are nonnegative. In other words, the dual is the following:

$$\begin{aligned} &\max b^T w \\ &\text{s.t. } A^T w \geq c \\ &w \geq 0 \end{aligned}$$

5.3 Show that the radial basis function kernel is equivalent to the computing similarity between two infinite dimensional feature vectors.

The radial basis function kernel can be expanded using the multinomial theorem, as follows. Assuming the sigma is equal to 1, we have the following:

$$\begin{aligned} \kappa(x, x') &= \exp\left(-\frac{1}{2}||x - x'||^2\right) \\ &= \exp\left(x^T x' - \frac{1}{2}||x||^2 - \frac{1}{2}||x'||^2\right) \\ &= \exp\left(x^T x'\right)\exp\left(-\frac{1}{2}||x||^2\right)\exp\left(-\frac{1}{2}||x'||^2\right) \\ &= \sum_{k=0}^{\infty}\frac{\left(x^T x'\right)^k}{k!}\exp\left(-\frac{1}{2}||x||^2\right)\exp\left(-\frac{1}{2}||x'||^2\right) \end{aligned}$$

This implies that RBF expansion has an infinite number of dimensions.

5.4 Verify that the learning rate schedule $\eta_k = (\tau_0 + k)^{-\kappa}$ satisfies Robbins-Monro conditions.

To show that the learning rate satisfies Robbins-Monro conditions, we need to show the following for nonnegative τ_0 and κ in $(0.5,1]$:

$$\sum_{k=1}^{\infty}\eta_k = \sum_{k=1}^{\infty}\frac{1}{(\tau_0 + k)^{\kappa}} = \propto$$

$$\sum_{k=1}^{\infty}\eta_k^2 = \sum_{k=1}^{\infty}\frac{1}{(\tau_0 + k)^{2\kappa}} < \infty$$

This is true according to power series convergence.

5.5 Compute the derivative of the sigmoid function $\sigma(a) = [1+\exp(-a)]^{-1}$.

We can compute the derivative as follows:

$$\begin{aligned}
\frac{d}{dx}\sigma(x) &= \frac{d}{dx}(1+e^{-x})^{-1} \\
&= -(1+e^{-x})^{-2}(-e^{-x}) \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}}\frac{e^{-x}}{1+e^{-x}} \\
&= \sigma(x)(1-\sigma(x))
\end{aligned}$$

5.6 Compute the runtime and memory complexity of the Bernoulli naive Bayes algorithm. The following pseudo-code captures the Bernoulli naive Bayes algorithm:

```
Training:
N_c = 0, N_jc = 0
for i = 1, 2, ..., n do
    c = y_i // class label for ith example
    N_c = N_c + 1
    for j = 1, ..., D do
        if x_ij = 1 then
            N_jc = N_jc + 1
    end for
end for
π̂_c = N_c / N, θ̂_jc = N_jc / N
return π̂_c, θ̂_jc
Testing (for a single test document):
for c = 1, 2, ..., C do
    log p[c] = log π_c
    for j = 1, 2, ..., D do
        if x_j = 1 then
            log p[c] += log θ̂_jc
        else
            log p[c] += log(1 − θ̂_jc)
    end for
end for
c = arg max_c log p[c]
return c
```

Given that we have two nested `for` loops, the runtime complexity is $O(ND)$, where N is the number of training documents and D is the dictionary size. The runtime complexity during test time is $O(TCD)$, where T is the number of test documents, T is the number of classes, and D is the dictionary size. Similarly, space complexity is the size of arrays required to store model parameters that grow as $O(DC)$.

6.1 Compute the runtime and memory complexity of a KNN regressor.

The following pseudo-code captures the KNN regression algorithm:

```
1: class KNN:
2: function knn_search(K, X, y, Q)
3: for query in Q:
4:     idx = argsort(euclidean_dist(query, X))[:K]
5:     knn_labels = [y[i] for i in idx]
6:     y_pred = mean(knn_labels)
7: end for
8: return y_pred
```

As we can see, we have a sorting operation that takes $O(N \log N)$ time, where N is the number of training data points. Thus, the runtime complexity is $O(N \log N + K) = O(N \log N)$ for each query q in Q. The KNN regressor is a nonparametric model; therefore, it has O(1) parameter space complexity.

6.2 Derive the Gaussian process update equations based on the rules for conditioning multivariate Gaussian random variables.

Consider the following joint GP distribution:

$$\begin{pmatrix} f \\ f_* \end{pmatrix} \sim \left(\begin{pmatrix} \mu \\ \mu_* \end{pmatrix}, \begin{pmatrix} K & K_* \\ K_*^T & K_{**} \end{pmatrix} \right)$$

Here, we want to predict the function outputs $y^*=f(x^*)$ (i.e., starred variables represent a prediction on test data X^* and $K=\kappa(X,X)$, $K^*=\kappa(X,X^*)$ and $K^{**}=\kappa(X^*,X^*)$). Using the following rules for conditional distributions, we have the following:

$$\begin{aligned} p(x_A|x_B) &= N\left(x_A|\mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(x_B - \mu_B), \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}\right) \\ p(x_B|x_A) &= N\left(x_B|\mu_B + \Sigma_{BA}\Sigma_{AA}^{-1}(x_A - \mu_A), \Sigma_{BB} - \Sigma_{BA}\Sigma_{AA}^{-1}\Sigma_{AB}\right) \end{aligned}$$

Here, set A corresponds to X and set B corresponds to X^*. We get the Gaussian process update equations:

$$\begin{aligned} p(f_*|X_*, X, f) &\sim N(f_*|\mu_*, \Sigma_*) \\ \mu_* &= \mu(X_*) + K_*^T K^{-1}(f - \mu(X)) \\ \Sigma_* &= K_{**} - K_*^T K^{-1} K_* \end{aligned}$$

7.1 Explain how temperature softmax works for different values of the temperature parameter T.

Temperature softmax is defined as follows:

$$p_i = \text{softmax}(x, T) = \frac{\exp\left(\frac{x_i}{T}\right)}{\sum_{j=1}^{N} \exp\left(\frac{x_j}{T}\right)}$$

When T=1, we get the regular soft-max function. As T approaches infinity, the output distribution becomes more uniform (all outcomes are more equally likely). Thus, heating the distribution increases its entropy. As T approaches 0, we are decreasing the entropy of the output distribution, thereby accentuating high probability outputs.

7.2 In the forward–backward HMM algorithm, store the latent state variable z (as part of the HMM class) and compare the inferred z against the ground truth z.

To store the latent state variable z, initialize it as part of HMM class constructor: `self.z=np.zeros(self.n)` and modify the subsequent code to use `self.z[idx]` instead of `z[idx]`.

8.1 Show that the Dirichlet distribution $\text{Dir}(\theta \mid \alpha)$ is a conjugate prior to the multinomial likelihood by computing the posterior. How does the shape of the posterior vary as a function of the posterior counts?

Suppose we observe N dice rolls D=$\{x_1,\ldots,x_n\}$, where x_i in $\{1,\ldots,K\}$. If we assume the data is independent and identically distributed, the likelihood has the following form:

$$p(\mathcal{D}|\theta) = \prod_{k=1}^{K} \theta_k^{N_k}$$

Here, N_k is the number of times event k occurred. Since the parameter vector lives on a K-dimensional probability simplex, we need a prior that has support over this simplex. The Dirichlet distribution satisfies this criterion:

$$\text{Dir}(\theta|\alpha) = \frac{1}{B(\alpha)} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1}$$

Multiplying the likelihood and the prior, we see that the posterior is also Dirichlet:

$$\begin{aligned} p(\theta|\mathcal{D}) &\propto p(\mathcal{D}|\theta)p(\theta) \\ &\propto \prod_{k=1}^{K} \theta_k^{N} \theta_k^{\alpha_k - 1} = \prod_{k=1}^{K} \theta_k^{\alpha_k + N_k - 1} \\ &= \text{Dir}(\theta|\alpha_1 + N_1, \ldots, \alpha_K + N_K) \end{aligned}$$

We can see that the posterior counts are obtained by adding the prior counts and the observed counts. Since the posterior distribution has the same form as the prior, Dirichlet is said to be conjugate prior to the multinomial likelihood. As the observed counts increase, the Dirichlet posterior becomes more concentrated.

8.2 Explain the principle behind K-means++ initialization.

K-means++ initialization addresses the problem of poor centroid initialization that leads to poor clustering results. The basic idea is that the initial cluster centroids should be far away from each other. The principle behind K-means++ initialization is to first sample a cluster centroid at random and then sample the subsequent centroids with probability proportional to the distance away from the existing centroids. This ensures the centroids are spread out and provides a better initialization strategy.

8.3 Prove the cyclic permutation property of the trace: tr(ABC) = tr(BCA) = tr(CAB).

Using the definition of trace, we can expand the product as follows:

$$tr(ABC) = \sum_i (ABC)_{ii} = \sum_i \sum_j A_{ij}(BC)_{ji} = \sum_i \sum_j \sum_k A_{ij}B_{jk}C_{ki};$$

Now, we can exchange the product of matrices and prove the permutation property:

$$tr(ABC) = \sum_i \sum_j \sum_k A_{ij}B_{jk}C_{ki} = \sum_i \sum_j \sum_k B_{jk}C_{ki}A_{ij} = tr(BCA)$$

8.4 Compute the runtime of the principal component analysis algorithm.

The pseudo-code for principal component analysis is as follows:

```
1: class PCA
2: function transform(X, K):
3: Σ = covariance_matrix(X)
4: V, λ = eig(Σ)
5: idx = argsort(λ)
6: V_pca = V[idx][ :K]
7: λ_pca = λ [idx][: K]
8: X_pca = XV_pca
9: return X_pca
```

The PCA algorithm has three computationally expensive steps: covariance matrix computation, eigenvalue decomposition, and sorting. Let N be the number of data points, each represented with D features, and the covariance matrix computation is $O(ND^2)$, where we assume $D < N$. The eigenvalue decomposition complexity is $O(D^3)$, while the sorting operation takes $O(D\log D)$ time. Therefore, the runtime of the PCA algorithm is $O(ND^2 + D^3 + D\log D) = O(ND^2 + D^3)$.

9.1 Explain how latent Dirichlet allocation can be interpreted as nonnegative matrix factorization.

Recall the term–document matrix A_{vd}, where V is the vocabulary and D is the number of documents. Since we are interested in finding the number of topics K, LDA can be interpreted as a matrix factorization problem that takes the term–document matrix A_{vd} and factorizes it into a product of topics W_{vk} and topic proportions H_{kd}: $A=WH$. Since topics and topic proportions are probability distributions, the resulting matrices are constrained to be nonnegative:

$$\min_{W \geq 0, H \geq 0} ||A - WH||_F$$

Therefore, LDA can be interpreted as nonnegative matrix factorization that can be solved using the alternating least squares (ALS) algorithm.

9.2 Explain why sparsity is desirable in inferring general graph structure.

Sparsity is a desirable property in graph structure inference because we want to be able to interpret our results. A fully connected graph does not provide much insight; therefore, we would like to adjust the threshold in the inverse covariance matrix estimator until we achieve a sparse graph structure we can draw meaningful conclusions from.

9.3 List several NP-hard problems that can be approximated with the simulated annealing algorithm.

Several applications of simulated annealing include the traveling salesman problem (TSP), maximum cut, independent set, and many others.

9.4 Brainstorm problems that can be efficiently solved by applying a genetic algorithm.

Some examples of problems that can be solved using genetic algorithms are the 8 queens problem, the traveling salesman problem, the feature selection problem, neural network topology, and many others.

10.1 Explain the purpose of nonlinearities in neural networks.

The purpose of nonlinearities in neural networks is to model nonlinear relationships between the input and output of a network layer. Without nonlinearities, the neural network will behave as a single-layer perceptron.

10.2 Explain the vanishing/exploding gradient problem.

Exploding gradients occur when the gradients get larger and larger with every layer during backpropagation. The vanishing gradient problem occurs when the gradients approach zero during backpropagation.

10.3 Describe some of the ways to increase neural model capacity and avoid overfitting.

To increase neural model capacity, we can increase the size and the number of layers. This can potentially lead to overfitting. To combat overfitting, we can use weight decay, dropout, and early stopping.

10.4 Why does the number of filters increase in LeNet architecture as we go from input to output?

The number of filters increases to capture more complex patterns in the image data. The earlier layers are used to extract geometric features, like edges, corners, and so on, while subsequent layers combine these patterns into more complex shapes.

10.5 Explain how an Adam optimizer works.

Adam can be seen as a variant on the combination of RMSProp and momentum optimizers. The update looks like RMSProp, except a smooth version of the gradient is used instead of the raw stochastic gradient. The full Adam update also includes a bias correction mechanism:

$$\begin{aligned} g &= \frac{1}{m}\nabla_\theta \sum_i L\left(\left(x^{(i)};\theta\right), y^{(i)}\right) \\ m &= \beta_1 m + (1-\beta_1) g \\ s &= \beta_2 v + (1-\beta_2) g^T g \\ \theta &= \theta - \epsilon_k \times \frac{m}{\sqrt{s+eps}} \end{aligned}$$

11.1 What is a receptive field in a CNN?

A receptive field is the number of input pixels that produce a feature map. It is a measure of association of an output feature (of any layer) to the input region.

11.2 Explain the benefit of residual connections by deriving the backward pass.

The benefit of residual connections is that they allow gradients to flow through the network directly, without passing through nonlinear activation functions that cause the gradients to explode or vanish (depending on the weights).

Let $H = F(x) + x$. Then, we get the following.

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial H}\frac{\partial H}{\partial x} = \frac{\partial L}{\partial H}\left(\frac{\partial F}{\partial x} + 1\right) = \frac{\partial L}{\partial H}\frac{\partial F}{\partial x} + \frac{\partial L}{\partial H}$$

11.3 Compare the pros and cons of using CNNs, RNNs, and transformer neural networks.

CNNs are highly parallelizable and, therefore, computationally fast. They are great at learning and detecting patterns in signals; however, they don't account for signal history. RNNs maintain an internal model of the past; however, they are restricted in

parallel computation. Transformers have a high model capacity for learning; however, they may result in overfitting.

11.4 Give an example of a neural network that uses amortized variational inference.

In the example we looked at in this book, we used a multilayer perceptron to learn the mixture model parameters. Thus, MLP is one example of a neural network that can amortize variational inference computations over multiple data points.

11.5 Show via an example the intuition behind the GCN forward equation: $D^{(-1/2)}(A+I)D^{(-1/2)}XW+b$.

The product AX in the equation represents a summation of neighboring node features in matrix X. By adding identity to A, we now have a summation that includes the node itself: $(A+I)X$. To normalize this product, we form a diagonal degree matrix D and pre and post multiply our product by the square root of D. Finally, we multiply our expression by the weight matrix W and add a bias term b.