1 Dimensionality Reduction

1.1 Linear Dimensionality Reduction

Goal: Find a linear embedding to minimize the *reconstruction error* of the original data.

1.1.1 Compression Perspective

For $k = 1$, we approximate each data point $x_i \in \mathbb{R}^d$ by a scalar $z_i \in \mathbb{R}$, and a direction $w \in \mathbb{R}^d$, so that $x_i \approx z_i w$. We want:

$$\min_{w,\{z_i\}} \sum_{i=1}^{n} \|x_i - z_i w\|^2 \quad \text{subject to} \quad \|w\| = 1.$$

If the data are centered (mean zero), the optimal z_i is given by $z_i = w^T x_i$. Hence, it suffices to optimize w.

1.1.2 General Case: k-Dimensional Subspace

For $k > 1$, we choose a matrix $W \in \mathbb{R}^{d\times k}$ with orthonormal columns and represent each x_i by coordinates $z_i \in \mathbb{R}^k$. The best reconstruction is

$$x_i \approx W z_i, \quad z_i = W^T x_i.$$

We minimize

$$\sum_{i=1}^{n} \|x_i - W(W^T x_i)\|^2.$$

This is the *Principal Component Analysis (PCA)* problem.

1.2 Principal Component Analysis (PCA)

- **Covariance Matrix:** Assume data are centered. The sample covariance is $C = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T$.
- **Eigen-Decomposition:** PCA chooses W whose columns are the top k eigenvectors of C. These eigenvectors (principal components) maximize variance along each axis while being orthonormal.
- **Minimize Reconstruction Error:** The chosen subspace is optimal for squared-error reconstruction.

1.2.1 SVD Connection

Let X be the $n \times d$ data matrix (rows are data points). A singular value decomposition is $X = U\Sigma V^T$, with $V \in \mathbb{R}^{d\times d}$ orthogonal. The columns of V are eigenvectors of $X^T X$. The first k columns of V correspond to the top k principal components.

1.2.2 Comparison to Other Methods

- **K-means vs. PCA:** Both can be viewed as "compression" but impose different structural constraints (PCA requires orthogonality, k-means uses cluster centers).
- **Autoencoders vs. PCA:** When autoencoders use linear activations, they are equivalent to PCA. Nonlinear activations can capture more complex low-dimensional structures.

1.3 Autoencoders

Autoencoders learn a function f such that $f(x) \approx x$, thereby forcing an intermediate bottleneck layer of dimension k to capture essential structure. Concretely:

$$\hat{x} = g(h(x)),$$

where $h : \mathbb{R}^d \to \mathbb{R}^k$ (the encoder) and $g : \mathbb{R}^k \to \mathbb{R}^d$ (the decoder). We train weights to minimize

$$\sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2.$$

- **Neural Network Autoencoder:** Typically a multilayer neural network used as h and g.
- **Linear Autoencoder:** Reduces to PCA.
- **Nonlinear Autoencoder:** Can capture manifolds or other complexities unreachable by linear PCA.

1.3.1 Training

- Use gradient-based methods (e.g., stochastic gradient descent) to learn network parameters.
- Initialization can be challenging; random starting points can lead to local minima or poor reconstructions.

2 Probabilistic Modeling & Parameter Estimation

2.1 Motivation for Probabilistic Modeling

- Traditional approaches: Fit functions $h(x)$ in supervised/unsupervised settings to minimize a loss (e.g., squared loss, hinge loss).
- **Limitation:** Purely deterministic viewpoint provides no direct measure of *uncertainty*.
- **Goal of Probabilistic Modeling:**
  - Quantify uncertainty about predictions.
  - Incorporate *prior assumptions* or domain knowledge about the data-generating process.

2.2 Connecting Risk Minimization and Probability Distributions

- **Supervised Learning Recap:** We typically assume i.i.d. data $D = \{(x_i, y_i)\}$ from an unknown distribution $P(X, Y)$.
- **Risk (Expected Error):**

$$R(h) \;=\; \mathbb{E}_{x,y}\big[\ell(y,\, h(x))\big],$$

  where ℓ is a chosen loss (e.g., squared loss).
- **Bayes' Optimal Predictor:** For squared error, the risk is minimized by $h^*(x) = \mathbb{E}[Y \mid X = x]$. In practice, $P(X, Y)$ is unknown; we must estimate from finite data.

2.3 Estimating Conditional Distributions

- We can try to learn an estimate $\hat{P}(Y|X)$ and use $\hat{y} = \mathbb{E}_{\hat{P}}[Y \mid X = x]$.
- **Parametric Estimation:** Assume a form $P(Y|X, \theta)$, then estimate parameters θ from data by maximizing the (conditional) likelihood:

$$\theta^* \;=\; \arg\max_{\theta}\; P(y_1, \ldots, y_n \mid x_1, \ldots, x_n, \theta).$$

- This *maximum likelihood estimation (MLE)* often coincides with standard methods. For example, under Gaussian noise, MLE aligns with least squares.

2.4 Example: Linear Regression with Gaussian Noise

- Suppose $y_i = w^T x_i + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ i.i.d.
- Then:

$$P(y_i \mid x_i, w) \;=\; \mathcal{N}(w^T x_i,\; \sigma^2).$$

- **Log-Likelihood:**

$$-\log P(y_1, \ldots, y_n \mid x_1, \ldots, x_n, w) \;\propto\; \sum_{i=1}^{n} \big(y_i - w^T x_i\big)^2.$$

- **Result:** MLE under this assumption reduces to solving a least squares problem. More generally, for any hypothesis class H, MLE with i.i.d. Gaussian noise is equivalent to minimizing $\sum (y_i - h(x_i))^2$.

2.5 Bias–Variance–Noise Decomposition

- Even if $h^* \in H$, finite data cause estimation error (variance). We also have irreducible noise.
- For squared error, expected risk can be decomposed as:

$$\mathbb{E}_{D,x,y}\big[(y - \hat{h}_D(x))^2\big] \;=$$

$$\underbrace{\big(\hat{h}_D(x) - h^*(x)\big)^2}_{\text{Bias}^2} \;+\; \underbrace{\text{Var}[\hat{h}_D(x)]}_{\text{Variance}} \;+\; \underbrace{\mathbb{E}[(y - h^*(x))^2]}_{\text{Noise}}.$$

- **Tradeoff:** High-capacity models have low bias but high variance; simpler models can reduce variance at the cost of higher bias. Regularization methods (e.g., Ridge, Lasso) strategically *add bias* to reduce variance.

2.6 Bayesian Interpretations

- **Prior Knowledge:** Instead of purely maximizing likelihood, we can incorporate a *prior* $P(w)$ over parameters.
- **MAP Estimation:**

$$w_{\text{MAP}} = \arg\max_{w}\; P(w \mid D) = \arg\max_{w}\; P(w) \prod_{i=1}^{n} P(y_i \mid x_i, w).$$

- For Gaussian priors on w, MAP estimation yields *Ridge Regression*. Different priors correspond to different regularizers.

3 Logistic Regression

3.1 Motivation for Statistical Classification

- In regression, we modeled continuous values (e.g., via least squares).
- In **classification**, the target y is a discrete label (e.g., $\{+1, -1\}$ or multiple classes).
- **Bayes' Optimal Classifier**:

$$h^*(x) \;=\; \arg\max_{y} P(Y = y \mid X = x).$$

Finding $P(Y \mid X)$ from data is thus a natural approach.

3.2 Logistic Regression for Binary Classification

- **Model Assumption:**

$$P(Y = +1 \mid X = x, w) \;=\; \sigma(w^\top x) \quad \text{where } \sigma(z) = \frac{1}{1 + e^{-z}}.$$

- Here, $w \in \mathbb{R}^d$ is the parameter vector. The predicted probability for the positive class $(+1)$ is $\sigma(w^\top x)$; for the negative class (-1), it is $1 - \sigma(w^\top x)$.
- **Interpretation:** Rather than modeling y directly with a linear function, we model the *log odds* of class membership linearly:

$$\log \frac{P(Y = +1 \mid x)}{P(Y = -1 \mid x)} \;=\; w^\top x.$$

3.3 Bernoulli Likelihood and Logistic Loss

- The binary label y_i is assumed to be drawn from a Bernoulli distribution:

$$P(y_i \mid x_i, w) \;=\; \sigma\big(y_i\, w^\top x_i\big),$$

where we encode $y_i \in \{+1, -1\}$, so $\sigma\big(y_i\, w^\top x_i\big) = \frac{1}{1+\exp(-\,y_i\, w^\top x_i)}$.

- **Log-likelihood:** Given n i.i.d. samples, the log-likelihood is

$$\ell(w) \;=\; \sum_{i=1}^{n} \log \sigma\big(y_i\, w^\top x_i\big).$$

- **Negative Log-likelihood (Logistic Loss):**

$$-\ell(w) \;=\; \sum_{i=1}^{n} \log\Big(1 + \exp\big(-\,y_i\, w^\top x_i\big)\Big).$$

Minimizing this loss corresponds to *maximum likelihood estimation (MLE)* for logistic regression.

3.4 Gradient Computation

- For a single training example (x, y), the logistic loss is:

$$\ell\big(w; x, y\big) \;=\; \log\big(1 + \exp(-\,y\, w^\top x)\big).$$

- **Gradient:**

$$\nabla_w\, \ell\big(w; x, y\big) \;=\; \frac{-\,y\, x}{1 + \exp\big(y\, w^\top x\big)}.$$

- For the full dataset, we sum the gradients or apply mini-batch / stochastic gradient descent.

3.5 Regularized Logistic Regression

- **Motivation:** Prevent overfitting and control model complexity.
- **Ridge (L2) Regularization:** Add $\lambda\|w\|_2^2$ to the logistic loss:

$$\min_w \;\sum_{i=1}^{n} \log\big(1 + e^{-\,y_i\, w^\top x_i}\big) \;+\; \lambda\|w\|_2^2.$$

- **Lasso (L1) Regularization:** Add $\lambda\|w\|_1$ to the logistic loss (enforces sparsity):

$$\min_w \;\sum_{i=1}^{n} \log\big(1 + e^{-\,y_i\, w^\top x_i}\big) \;+\; \lambda\|w\|_1.$$

- **Interpretation:** These can be seen as *Maximum A Posteriori (MAP)* estimates with Gaussian (L2) or Laplace (L1) priors on w.

3.6 Algorithmic Outline

1. **Initialize** parameters $w^{(0)}$ (e.g., to small random values or zeros).
2. **Repeat** (for $t = 1, 2, \ldots$):
   (a) Draw a mini-batch (or single sample) (x_i, y_i) from the training set.
   (b) Compute gradient of the (regularized) logistic loss.
   (c) Update $w^{(t+1)} = w^{(t)} - \eta\, \nabla_w\big[\text{loss}\big]$, where η is the learning rate.
3. **Convergence** can be checked by monitoring changes in w or by evaluating a separate validation set.

3.7 Prediction

- **Probability of Positive Class:**

$$P(Y = +1 \mid x, w) = \sigma\big(w^\top x\big).$$

- **Binary Decision:**

$$\hat{y} \;=\; \text{sign}\big(w^\top x\big).$$

- For multi-class problems, the model generalizes to *softmax regression* (each class has its own parameter vector).

4 Generative Modeling

4.1 Discriminative vs. Generative Approaches

- **Discriminative Models:** Estimate $P(y \mid x)$ directly (e.g., logistic regression). Focus on decision boundary or conditional probability of labels given inputs.
- **Generative Models:** Estimate the joint distribution $P(x, y)$, often factorized as $P(y)\, P(x \mid y)$. Once we have $P(x, y)$, we can use Bayes' rule to get $P(y \mid x)$.
- **Key Differences:**
  - Generative models can generate new data and detect outliers (since $P(x)$ is modeled).
  - Discriminative models often yield better classification accuracy when $P(x)$ is complex or high-dimensional.

4.2 Naïve Bayes Classifier

4.2.1 Model Setup

- **Goal:** Classify data into one of c classes $y \in \{1, \ldots, c\}$.
- **Assumption:** Features $x = (x_1, \ldots, x_d)$ are conditionally independent given y. Thus,

$$P(x \mid y) \;=\; \prod_{i=1}^{d} P\big(x_i \mid y\big).$$

- Combined with $P(y)$, we obtain the joint:

$$P(x, y) \;=\; P(y) \prod_{i=1}^{d} P\big(x_i \mid y\big).$$

4.2.2 Parameter Estimation (MLE)

- **Label Prior:**

$$P(y) \;\approx\; \frac{\#(\text{points in class } y)}{n}.$$

- **Conditional Feature Distribution:** Each $x_i \mid y$ can be modeled according to a parametric family. Examples:
  - Bernoulli or Multinomial if x_i is categorical.
  - Gaussian if x_i is continuous (leading to *Gaussian Naïve Bayes*).
- Use the training data $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ to fit these distributions by maximum likelihood.

4.2.3 Prediction

- For a new input x, predict the class $\hat{y}$ by:

$$\hat{y} \;=\; \arg\max_y \; P(y \mid x) \;=\; \arg\max_y \; P(y) \prod_{i=1}^{d} P\big(x_i \mid y\big).$$

- In practice, for numerical stability, use log probabilities:

$$\hat{y} \;=\; \arg\max_{y} \Big(\ln P(y) \;+\; \sum_{i=1}^{d} \ln P\big(x_i \mid y\big)\Big).$$

4.3 Gaussian Naïve Bayes (GNB)

4.3.1 Model Setup

- Suppose features x are real-valued and for each class y, the conditional distribution $x_i \mid (y)$ is Gaussian with mean $\mu_{y,i}$ and variance $\sigma^2_{y,i}$:

$$P\big(x_i \mid y\big) \;=\; \mathcal{N}\big(x_i \;\big|\; \mu_{y,i},\; \sigma^2_{y,i}\big).$$

- The parameters $\{\mu_{y,i}, \sigma^2_{y,i}\}$ are estimated from the training data by MLE (i.e., sample mean and variance per class).

4.3.2 Prediction and Decision Boundaries

- Once we have $\mu_{y,i}$, $\sigma^2_{y,i}$, and $P(y)$, the posterior $P(y \mid x)$ is computed via Bayes' rule:

$$P(y \mid x) \;=\; \frac{P(y)\,\prod_{i=1}^{d} \mathcal{N}\big(x_i \;\big|\; \mu_{y,i},\; \sigma^2_{y,i}\big)}{\sum_{y'}\; P(y')\,\prod_{i=1}^{d} \mathcal{N}\big(x_i \mid \mu_{y',i},\; \sigma^2_{y',i}\big)}.$$

- For binary classification ($y \in \{-1, +1\}$) and a simplified assumption of class-invariant variances, GNB yields a linear decision boundary resembling logistic regression.

4.4 Comparison: Discriminative vs. Generative

- **Discriminative (e.g. logistic regression):**
  - Directly model $P(y \mid x)$ (or a decision boundary).
  - Often more accurate classification if x is high-dimensional and complex.
  - Cannot generate new samples x or detect outliers as easily.
- **Generative (e.g. Naïve Bayes):**
  - Model $P(x, y)$ or $P(x \mid y)$ and $P(y)$.
  - Can generate samples from the learned model and identify outliers by evaluating $P(x)$.
  - May be more sensitive to model mismatch if $P(x)$ is not well-approximated.

5 Generative Classification

5.1 Recap: Gaussian Naïve Bayes (GNB)

- **Basic Assumption:** Condition on the class label $y \in \{1, \ldots, c\}$, features $x = (x_1, \ldots, x_d)$ are independent Gaussians.
- **Parameters:** For each class y, we estimate a mean $\mu_{y,i}$ and variance $\sigma^2_{y,i}$ for each feature i.
- **Decision Rule:** Predict

$$\hat{y} = \arg\max_{y}\; P(y) \prod_{i=1}^{d} \mathcal{N}\big(x_i \mid \mu_{y,i}, \sigma^2_{y,i}\big).$$

- **Limitation:** The conditional independence assumption can lead to *overconfidence*, as correlations among features are ignored.

5.2 Gaussian Bayes Classifiers (GBC)

5.2.1 Generalizing GNB

- Instead of assuming each feature is independent given y, we allow $x \mid y$ to follow a *multivariate* Gaussian with mean $\mu_y \in \mathbb{R}^d$ and covariance $\Sigma_y \in \mathbb{R}^{d \times d}$.
- Hence, for class y,

$$P(x \mid y) \;=\; \frac{1}{(2\pi)^{\frac{d}{2}}\sqrt{\det(\Sigma_y)}} \exp\Big(-\tfrac{1}{2}(x-\mu_y)^\top \Sigma_y^{-1}(x-\mu_y)\Big).$$

- This approach captures correlations among features, unlike naive Bayes.

5.2.2 Parameter Estimation (MLE)

- **Label Prior:** $P(y) \approx \frac{\#(\text{training points in class } y)}{n}$.
- **Mean & Covariance:** For each class y,

$$\mu_y \;=\; \frac{1}{n_y} \sum_{(x_i, y_i) = D_y} x_i,$$

$$\Sigma_y \;=\; \frac{1}{n_y} \sum_{(x_i, y_i) = D_y} (x_i - \mu_y)(x_i - \mu_y)^\top,$$

where n_y is the number of samples in class y, and D_y is the subset of training data in class y.

5.2.3 Decision Rule

- By Bayes' rule,

$$P(y \mid x) \;=\; \frac{P(y)\,\mathcal{N}(x \mid \mu_y, \Sigma_y)}{\sum_{y'} P(y')\,\mathcal{N}(x \mid \mu_{y'}, \Sigma_{y'})}.$$

- Hence, classification uses a **discriminant function**, e.g. for binary classification:

$$f(x) = \ln P(y = +1 \mid x) - \ln P(y = -1 \mid x),$$

which simplifies under Gaussian assumptions.

5.3 Special Cases

5.3.1 Fisher's Linear Discriminant Analysis (LDA)

- **Assumption:** Two classes ($c = 2$), each with mean μ_+, μ_- but *common* covariance matrix Σ.

- **Linear Discriminant:** The difference in log-likelihoods becomes a linear function in x:

$$f(x) \;=\; (\mu_+ - \mu_-)^\top \Sigma^{-1} x \;+\; \text{constant}.$$

- **Equivalence with Logistic Regression:** Under these assumptions, LDA yields a decision boundary identical to logistic regression, though LDA is generative (models $P(x, y)$) and logistic regression is discriminative (models $P(y \mid x)$).

5.3.2 Naïve Bayes as GBC with Diagonal Covariance

- **Naïve Bayes** can be viewed as a GBC where Σ_y is restricted to be diagonal (ignoring off-diagonal covariances).
- Reduces parameters from $O(d^2)$ to $O(d)$ for each class.
- Potentially less accurate if features are strongly correlated.

5.4 Regularization and Priors

- **Overfitting Risk:** Estimating a full covariance Σ_y can require many parameters ($d(d+1)/2$ per class), leading to overfitting.
- **Remedies:**
  - Restrict to simpler structures: shared covariance (LDA) or diagonal covariance (Naïve Bayes).
  - Incorporate *priors* on μ_y or Σ_y (e.g., conjugate priors), effectively shrinking estimates.
  - Perform dimensionality reduction or add regularization terms.

5.5 Comparisons

- **Generative** (e.g., GBC) vs. **Discriminative** (e.g., logistic regression):
  - Generative: Can handle outlier detection via $P(x)$ and data generation, but may be less robust to misspecified $P(x)$.
  - Discriminative: Often better classification accuracy in high-dimensional settings and fewer assumptions on x.
- **GNB vs. GBC**:
  - GNB: Diagonal covariance $\implies$ fewer parameters, more risk of "overconfidence" if features are correlated.
  - GBC: Full covariance $\implies$ more flexible but can overfit without regularization or sufficient data.

6 Gaussian Mixture Models (GMMs)

6.1 Motivation

- In generative models like Gaussian Bayes Classifiers, we assume labeled data and fit one Gaussian per class.
- **Key Question:** What if labels are *missing*? Could we still fit a model to data that (we suspect) come from a mixture of Gaussians?
- **Example:** Data are drawn from multiple "clusters," each described by its own Gaussian. The cluster assignments (latent labels) are unknown.

6.2 Definition and Parametrization

- A **Gaussian Mixture Model (GMM)** is a convex combination of Gaussian densities:

$$p(x) \;=\; \sum_{z=1}^{k} \pi_z \, \mathcal{N}\big(x \mid \mu_z, \Sigma_z\big),$$

  where:
  - k is the number of mixture components (clusters).
  - $\pi_z \geq 0$, $\sum_{z=1}^{k} \pi_z = 1$ are the mixture weights.
  - Each component has mean μ_z and covariance Σ_z.
- We can think of a latent (unobserved) variable z indicating which component generated a given data point x.

6.3 MLE for Gaussian Mixture Models

- Given data $D = \{x_1, \ldots, x_n\}$ (unlabeled), we want to fit the parameters $\{\pi_z, \mu_z, \Sigma_z\}_{z=1}^{k}$ by maximizing the likelihood:

$$\mathcal{L}(\{\pi_z, \mu_z, \Sigma_z\}) \;=\; \prod_{i=1}^{n} \sum_{z=1}^{k} \pi_z \, \mathcal{N}\big(x_i \mid \mu_z, \Sigma_z\big).$$

- The log-likelihood is non-convex and direct gradient methods can be unstable due to constraints ($\sum_z \pi_z = 1$, $\Sigma_z \succ 0$).

6.4 Expectation-Maximization (EM) Algorithm

- **Idea:** If the latent "cluster labels" z_i were known, we could estimate each Gaussian in closed form (as in a Gaussian Bayes classifier). But z_i are unknown.
- **EM Approach:** Iteratively refine a guess of $\{z_i\}$ (soft assignments) and update parameters:
  1. **E-step:** For each data point x_i, compute the posterior distribution over clusters:

$$\gamma_{i,z} \;=\; P(z \mid x_i) \;=\; \frac{\pi_z \, \mathcal{N}(x_i \mid \mu_z, \Sigma_z)}{\sum_{z'} \pi_{z'} \, \mathcal{N}(x_i \mid \mu_{z'}, \Sigma_{z'})}.$$

2. **M-step:** Update mixture weights, means, and covariances using the soft assignments $\gamma_{i,z}$:

$$\pi_z = \frac{1}{n}\sum_{i=1}^{n}\gamma_{i,z}, \quad \mu_z = \frac{\sum_{i=1}^{n}\gamma_{i,z}\,x_i}{\sum_{i=1}^{n}\gamma_{i,z}},$$

$$\Sigma_z = \frac{\sum_{i=1}^{n}\gamma_{i,z}\,(x_i-\mu_z)(x_i-\mu_z)^\top}{\sum_{i=1}^{n}\gamma_{i,z}}.$$

- **Guaranteed Monotonic Increase in Log-likelihood:** EM iterates ensure each step does not decrease the data log-likelihood.
- **Initialization Sensitivity:** Different initial parameters can lead to different local maxima; multiple restarts are common.

6.5 Model Degeneracies and Regularization

- **Covariance Collapse:** A Gaussian might "collapse" onto a single data point with near-zero variance, leading to infinite log-likelihood.
- **Remedy:** Use a small regularization term added to each covariance matrix (or a Bayesian prior like the Inverse Wishart) to prevent singularities.

6.6 Interpretation

- **Clustering:** GMM gives soft cluster assignments $\gamma_{i,z}$ instead of the hard assignments from e.g. k-means.
- **Density Estimation:** Once fitted, $p(x)$ can be computed for any point x, enabling outlier/anomaly detection by thresholding $p(x)$.
- **Relation to Gaussian Bayes Classifiers (GBC):**
  - GBC: The "cluster label" is observed (*the class label*). MLE is straightforward and closed-form.
  - GMM: The cluster label z is *latent*; we solve it with EM.
- **Classification via Mixtures:** If labeled data are available for each class, each class can be modeled by a mixture of Gaussians, and classification proceeds via Bayes' rule.

7 Deep Generative Models

7.1 Motivation

- Traditional generative models (e.g., Gaussian mixture models) make strong assumptions on data distribution; can struggle with complex, high-dimensional domains (images, audio, text).
- **Goal:** Use flexible neural networks to learn a generative model for complex distributions, often without explicit likelihood formulas.
- Examples of **deep generative modeling** techniques:
  - Generative Adversarial Networks (GANs)
  - Denoising Diffusion Models (DDMs)
  - Variational Autoencoders (VAEs) (not fully covered here, but widely used)

7.2 Generative Adversarial Networks (GANs)

7.2.1 Key Idea

- **Generator (G):** Takes a noise vector z (e.g., from a Gaussian) and outputs a "synthetic" sample x_{fake}.
- **Discriminator (D):** Attempts to distinguish "real" data from "fake" data produced by G.
- **Adversarial Objective:**

$$\min_G \max_D \; \mathbb{E}_{x\sim p_{\text{data}}}\big[\log D(x)\big] + \mathbb{E}_{z\sim p_z}\big[\log\big(1-D(G(z))\big)\big].$$

- This is a **two-player game** in parameter space; G tries to fool D, and D tries to detect fakes.

7.2.2 Training

- **Simultaneous Gradient Descent:** Update D to better classify real vs. fake, then update G to produce samples more likely to be classified as real.
- **Challenges:**
  - *Convergence Issues:* The objective is a saddle point, not a simple minimum; training can oscillate.
  - *Mode Collapse:* G can learn to produce limited varieties of samples that fool D but fail to cover the whole data distribution.

7.3 Denoising Diffusion Models (DDMs)

7.3.1 Forward and Reverse Processes

- **Forward Diffusion:** Gradually add noise to real data $x_0 \to x_1 \to \cdots \to x_T$, eventually reaching a noisy x_T close to pure noise.
- **Reverse (Denoising) Process:** Learn to iteratively remove noise step-by-step to reconstruct a sample from x_T back to x_0.
- **Trainable Neural Network:** Models each reverse transition $p_\theta(x_{t-1} \mid x_t)$, typically approximated by a conditional Gaussian for small steps.

7.3.2 Training and Generation

- **Objective:** Minimize denoising error, e.g. predicting noise or x_{t-1} from x_t.
- **Sampling:** Start with random noise x_T, iteratively apply the learned reverse diffusion to produce x_0 (a synthetic sample).
- **Benefits:**

- *Stable Training:* Each small diffusion step is simpler to learn.
- *High-Quality Samples:* DDMs can match or surpass GAN performance on images.

7.4 Comparison and Challenges

- **GANs:**
  - Can produce sharp, high-resolution samples.
  - Prone to mode collapse and training instability.
  - No explicit probability density function for $p(x)$; hard to evaluate likelihood or coverage.
- **Diffusion Models:**
  - More stable training, less prone to mode collapse.
  - Slower sampling times due to iterative denoising steps.
  - Achieve state-of-the-art results in image synthesis and text-to-image tasks.
- **General Pitfalls:**
  - Evaluating generative models remains non-trivial (lack of straightforward log-likelihood for implicit models).
  - Hyperparameter tuning (architectures, noise schedules, adversarial loss weighting) is often domain-specific.

8 1. [13 points + 4 bonus pts] Linear neural networks

For input $x \in \mathbb{R}^{d_0}$, a linear neural network $F : \mathbb{R}^{d_0} \to \mathbb{R}$ of depth L will output

$$F(x) = W_L W_{L-1} \cdots W_1 x,$$

where each $W_l \in \mathbb{R}^{d_l \times d_{l-1}}$; d_l denotes the number of input units at layer $l \in \{1, \ldots, L-1\}$ and $d_L = 1$. We aim to train F to minimize the mean squared error loss on predicting real-valued scalar labels y. The loss is specified by

$$\ell(F) = \frac{1}{2n} \sum_{i=1}^{n} \big(F(x_i) - y_i\big)^2,$$

where i ranges over the dataset of size n.

(a)

Determine whether each of the following statements is True or False, and briefly justify your answers.

(1) [2 points, TF]

$F(x)$ is a linear function in x. That is, for all $a, b \in \mathbb{R}$ and all input vectors $x, x' \in \mathbb{R}^{d_0}$,

$$F(ax + bx') = aF(x) + bF(x').$$

(A) True (B) False

Solution:

True. F is a linear function with $W_L W_{L-1} \cdots W_1$ being effectively a single matrix (or row vector when $d_L = 1$) multiplying x.

(2) [2 points, TF]

Networks with increasing depth L allow one to model more complex relationships between x and y.

(A) True (B) False

Solution:

False. Even with increasing depth, F remains a linear function of x. Extra linear layers do not increase representational power in this purely linear setting.

(3) [2 points, TF]

$\ell(F)$ is convex with respect to W_l for all $l \in \{1, \ldots, L\}$.

(A) True (B) False

Solution:

True. F is linear in each W_l. The squared-error loss is convex when viewed as a function of any one W_l (holding the others fixed).

(b)

Now consider the case where $L = 2$; that is, $F(x) = W_2 W_1 x$. When running gradient descent to minimize $\ell(F)$, we need to compute the gradients of each layer. The next two questions ask you to compute them.

Hint: Make sure that the gradients have the correct shape, such that the gradient descent updates can be written as

$$W_i^{(t+1)} \leftarrow W_i^{(t)} - \eta \nabla_{W_i} \ell\big(F^{(t)}\big).$$

(1) [3 points, MC]

What is the correct expression for $\frac{\partial \ell(F)}{\partial W_2}$?

(A) $\frac{1}{n} \sum_{i=1}^{n} (F(x_i) - y_i)\, W_1 x_i$
(B) $\frac{1}{n} \sum_{i=1}^{n} F(x_i)\, W_1 x_i$
(C) $\frac{1}{n} \sum_{i=1}^{n} (F(x_i) - y_i)\, x_i^\top\, W_1^\top$
(D) $\frac{1}{n} \sum_{i=1}^{n} F(x_i)\, x_i^\top\, W_1^\top$

Solution:

(C). The shape and chain rule considerations yield $\frac{\partial \ell}{\partial W_2} \propto (F(x_i) - y_i)\, x_i^\top W_1^\top$. Accounting for the correct dimensions and factor $1/n$ leads to option (C).

(2) [4 points, MC]

We now use backpropagation to compute $\frac{\partial \ell(F)}{\partial W_1} \in \mathbb{R}^{d_1 \times d_0}$. What is the correct expression for $\frac{\partial \ell(F)}{\partial W_1}$?

(A) $\frac{1}{n} \sum_{i=1}^{n} (F(x_i) - y_i)\, x_i^\top\, W_2^\top$
(B) $\frac{1}{n} \sum_{i=1}^{n} F(x_i)\, x_i^\top\, W_2^\top$
(C) $\frac{1}{n} \sum_{i=1}^{n} (F(x_i) - y_i)\, W_2^\top\, x_i^\top$
(D) $\frac{1}{n} \sum_{i=1}^{n} F(x_i)\, W_2^\top\, x_i^\top$

Solution:

(C). By the chain rule and matching dimensions, the correct form is $\frac{1}{n} \sum (F(x_i) - y_i)\, W_2^\top x_i^\top$.

(c) [Bonus: 4 points, MC]

We have a single-layer fully connected neural network with input nodes $v_i, i = 1, \ldots, d$, and a single output node v_{out}. The activation function of the output node v_{out} is the identity function. We initialize every weight independently with a standard Gaussian distribution

$$w_i \sim \mathcal{N}(0, \sigma^2), \quad i \in \{1, \ldots, d\}.$$

To avoid overfitting, we use dropout and thus independently set each node v_j to zero with probability $1 - p$.

Assume we give as input to the network d independent random variables $X_i, i = 1, \ldots, d$, with

$$\mathbb{E}[X_i] = 0, \quad \mathbb{E}[X_i^2] = 1.$$

How should we choose σ^2 so that we have $\mathbb{E}[v_{\text{out}}] = 0$ and $\mathbb{E}[v_{\text{out}}^2] = 1$? Here the randomness is over the random variables X_i, weights w_i, and node dropout events.

(A) $\sigma^2 = \frac{2}{d\,p\,(1-p)}$
(B) $\sigma^2 = \frac{2}{d\,p}$
(C) $\sigma^2 = \frac{1}{d\,p\,(1-p)}$
(D) $\sigma^2 = \frac{1}{d\,p}$
Solution:
(D). We have

$$v_{\text{out}} = \sum_{i=1}^{d} w_i\, X_i\, D_i,$$

where D_i is a Bernoulli variable with probability p. Then

$$\mathbb{E}[v_{\text{out}}^2] = \sum_{i=1}^{d} \mathbb{E}[w_i^2]\,\mathbb{E}[X_i^2]\,\mathbb{E}[D_i^2] = d\,\sigma^2\,p.$$

Setting that equal to 1 yields $\sigma^2 = \frac{1}{d\,p}$.

9 2. [6 points + 3 bonus pts] Neural networks

(a) [2 points, MC]

Which of the following statements about training neural networks is True?

(A) Nonlinear activation functions are often only applied to the output units.

(B) The cross entropy loss is designed for regression tasks, where the goal is to predict arbitrary real-valued labels.

(C) Increasing the minibatch size in stochastic gradient descent (SGD) lowers the variance of gradient estimates (assuming data points in the mini-batch are selected independently).

(D) For a minibatch size of 1, SGD is guaranteed to decrease the loss in every iteration.

(E) None of the above.

Solution:

(C). Nonlinear activations are crucial in hidden layers (not just output). The cross entropy loss is primarily for classification, not general real-valued regression. SGD with minibatch size 1 is not guaranteed to reduce the loss at every step. Increasing batch size does lower variance in the gradient estimate.

(b) [2 points, MC]

Which of the following statements about convolutional neural networks (CNNs) for image analysis is True?

(A) They do not require non-linear activations to learn non-linear decision boundaries.

(B) They can only be used in shallow neural networks.

(C) Pooling layers reduce the spatial resolution of the image.

(D) They cannot be used for unsupervised learning.

(E) None of the above.

Solution:

(C). Pooling layers reduce the spatial resolution. CNNs do require nonlinear activations to learn complex patterns, can be used in deep architectures, and can be applied to unsupervised learning tasks.

(c) [2 points, MC]

Which of the following statements about Generative Adversarial Networks (GANs) is False?

(A) GANs represent a neural network architecture that is comprised of a generator and a discriminator.

(B) Training a GAN requires finding a saddle point of the objective function rather than a local optimum.

(C) In practice, training a GAN is difficult due to mode collapse and training instability.

(D) GANs can be evaluated by computing their log-likelihood for held-out samples.

(E) None of the above.

Solution:

(D) is false. Although you can approximate such evaluations in special cases, GANs do not directly provide a tractable log-likelihood measure, making it difficult to evaluate by standard likelihood-based approaches.

(d) [Bonus: 3 points, MC]

You train a generative adversarial network with neural network discriminator D and neural network generator G. Let $p_{\text{data}}(x)$ be the probability density function of training examples x, and $p_G(x)$ be the probability density of x under the generator G. The optimal discriminator for G is defined as

$$D^* = \arg\max_D \Big(\mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim \mathcal{N}(0,I)}[\log(1 - D(G(z)))] \Big).$$

For a training sample x with $p_{\text{data}}(x) = \frac{1}{100}$ and $p_G(x) = \frac{1}{50}$, what is the probability of D^* classifying x as being from the generator?

(A) $\frac{1}{4}$
(B) $\frac{1}{3}$
(C) $\frac{1}{2}$
(D) $\frac{2}{3}$
(E) $\frac{3}{4}$
Solution:
(D). The optimal discriminator is $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}$.

Thus the probability of classifying x as from the generator is

$$1 - D^*(x) = 1 - \frac{1/100}{(1/100) + (1/50)} = 1 - \frac{1}{1+2} = \frac{2}{3}.$$

10 3. [6 points] Clustering

(a) [2 points, MC]

Which of the following statements is False about k-means clustering?

(A) It seeks cluster centers and assignments to minimize the within-cluster sum of squares.

(B) For fixed assignments of sample points to cluster centers, computing the optimal cluster centers is a non-convex optimization problem.

(C) It is an appropriate algorithm if the underlying clusters are separable, spherical and approximately of the same size.

(D) None of the above.

Solution:

(B) is false. For fixed assignments, finding optimal cluster centers is a convex problem in the centers. Minimizing the sum of squared distances to a set of points has a unique global minimum (their mean) for each cluster.

(b) [2 points, MC]

Which of the following statements is False about Lloyd's algorithm for k-means clustering?

(A) It is guaranteed to monotonically decrease the k-means cost function (the average squared distance).

(B) It is not guaranteed to terminate with the globally optimal solution.

(C) Using specialized initialization schemes (e.g. k-means++) can improve the quality of solutions found by the algorithm.

(D) The number of iterations until convergence is guaranteed to be polynomial in the number of cluster centers and data points.

Solution:

(D) is false. Lloyd's algorithm can, in worst-case scenarios, take super-polynomial time to converge. Statements (A), (B), and (C) are true.

(c) [2 points, MC]

In k-means clustering, which of the following approaches is not suitable for selecting the cluster number k?

(A) By using a heuristic like the elbow method that identifies the diminishing returns in the loss function.

(B) By using a regularizor to favor simpler models with lower k.

(C) By using a validation set to select the best k on the held-out data.

(D) None of the above.

Solution:

(C) is not suitable, because with more clusters one often can keep decreasing the in-sample or out-of-sample distortion. A simple validation set approach does not typically fix an optimal k by standard risk. Heuristics like the elbow method or adding a complexity penalty are common.

11 4. [13 points] Principal component analysis

(a)

Determine whether each of the following statements is True or False, and briefly justify your answers.

(1) [2 points, TF]

PCA helps us find a linear mapping to a lower dimensional space.

(A) True (B) False

Solution:

True. By definition, PCA finds a linear projection to the principal components.

(2) [2 points, TF]

Imagine two features are identical in the whole dataset, that is, they are identical among all data samples $x_1, \ldots, x_n$. Then, utilizing PCA, we can strictly reduce the dimension of the dataset by at least one with zero reconstruction error.

(A) True (B) False

Solution:

True. If two features are identical, one direction of variation is completely redundant, giving a zero eigenvalue in the covariance matrix. PCA can compress by at least one dimension without losing information.

(b)

We apply PCA to the points plotted in the figure below. Vectors $(v_1, \ldots, v_5)$ in the figure indicate possible directions of the principal components. Note that these vectors are scaled in the figure for easier readability, and only their direction is relevant. Answer the following two questions:

(Imagine a figure named pca_example.pdf illustrating points and poss

(1) [2 points, MC]

Which vector corresponds to the first principal component?

(A) v_1 (B) v_2 (C) v_3 (D) v_4 (E) v_5

Solution:

(A). The first principal component is in the direction of maximal variance, marked by v_1.

(2) [2 points, MC]

Which vector corresponds to the second principal component?

(A) v_1 (B) v_2 (C) v_3 (D) v_4 (E) v_5

Solution:

(C). The second principal component is orthogonal to the first and corresponds to v_3 here.

(c)

In PCA, we map data points $x_i \in \mathbb{R}^d$, $i = 1, \ldots, n$, to $z \in \mathbb{R}^k$ with $k \ll d$ by solving the following optimization problem:

$$C^* = \frac{1}{n} \min_{W \in \mathbb{R}^{d \times k},\, W^\top W = I,\, z_1, \ldots, z_n \in \mathbb{R}^k} \sum_{i=1}^{n} \| W z_i - x_i \|_2^2.$$

We denote by $W^*, z_1^*, \ldots, z_n^*$ the optimal solution of the above, and assume the data are centered so that $\sum_{i=1}^n x_i = 0$.

(1) [2 points, MC]

What holds for z_i^*?
(A) $z_i^* = W^{*\top}(W^* W^{*\top})^{-1} x_i$
(B) $z_i^* = (W^{*\top} W^*)^{-1} W^{*\top} x_i$
(C) $z_i^* = (W^* W^{*\top})^{-1} W^{*\top} x_i$
(D) $z_i^* = W^{*\top}(W^* W^{*\top})^{-1} W^* x_i$
Solution:
(B). Because $W^\top W = I$ in PCA, the projection of x_i is simply $W^{*\top} x_i$. That is the same as $(W^{*\top} W^*)^{-1} W^{*\top} x_i$. In fact, $W^{*\top} W^* = I_k$.

(2) [3 points, MC]

What is the value of $\mathrm{Tr}(W^* W^{*\top})$?
(A) n
(B) k
(C) d
(D) $\max(n, d)$
Solution:
(B). Since the columns of W^* are orthonormal in $\mathbb{R}^d$, the trace of $W^* W^{*\top}$ counts the number of orthonormal vectors, i.e.k.

12 5. [16 points] Probabilistic classification

(a) [2 points, MC]

Consider a dataset $D = \{x_i\}_{i=1}^n$ and assume that the likelihood function $P(D \mid \theta)$ depends on some parameters θ to be estimated. Furthermore, we assume we have access to the true prior $P(\theta)$ over the parameters. Choose the correct statement:

(A) The maximum a posteriori (MAP) estimate and the maximum likelihood estimate (MLE) coincide, since they both involve a maximization of the likelihood $P(D|\theta)$.

(B) The posterior distribution over θ is given by $P(\theta|D) = P(D|\theta)\, P(\theta)$.

(C) The MLE estimate maximizes $P(D|\theta)$ and the MAP estimate maximizes $P(\theta|D)$.

(D) The MLE estimate is a point estimate, whereas the MAP estimate outputs a distribution.

Solution:
(C). The posterior is proportional to the product of likelihood and prior, but that is $\propto$. So (B) as an equality is missing the normalizing constant. The MLE maximizes $P(D|\theta)$. The MAP maximizes $P(\theta|D)$.

(b) [2 points, MC]

Match the following statistical learning models (left) with their corresponding likelihood functions (right) for MLE estimate. Fill in the blank.
(1) Logistic regression
(2) Least squares for linear regression
(A) Multinomial likelihood
(B) Gaussian likelihood
(C) Bernoulli likelihood
(D) Laplace likelihood
Solution:
(1) Logistic regression → (C) Bernoulli likelihood
(2) Least squares linear regression → (B) Gaussian likelihood

(c) [3 points, MC]

You have trained a generative model on binary labels and you want to predict the label for a new data point x. According to your model:

$$P(Y = 0) = 0.5, \quad P(Y = 1) = 0.5,$$

$$P(x \mid Y = 0) = 0.1, \quad P(x \mid Y = 1) = 0.3.$$

What is the probability of x belonging to class 0?
(A) 0.05
(B) 0.1
(C) 0.25
(D) cannot be calculated without knowing $p(x)$
Solution:
(C) 0.25. By Bayes' rule:

$$P(Y = 0 \mid x)$$

$$= \frac{P(x \mid Y = 0)\, P(Y = 0)}{P(x \mid Y = 0)\, P(Y = 0) + P(x \mid Y = 1)\, P(Y = 1)}$$

$$= \frac{0.1 \times 0.5}{0.1 \times 0.5 + 0.3 \times 0.5} = 0.25.$$

(d)

Consider a classification task using linear discriminant analysis (LDA). Given a data set $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, let n_j represent the number of samples such that $y = j$. We assume the following simplified model,

$$P(X \mid Y = j) = \mathcal{N}(\mu_j, I),$$

where $I \in \mathbb{R}^{d \times d}$ is the identity matrix and $j \in \{1, 2\}$. Furthermore, let $P(Y = 1) = p$, and $P(Y = 2) = 1 - p$.

(1) [2 points, MC]

Which of the following expresses the MLE $\hat{\mu}_1$ of μ_1?
(A) $\hat{\mu}_1 = \frac{1}{n_1} \sum_{i:y_i=1} x_i$
(B) $\hat{\mu}_1 = \frac{n_1}{n_1+n_2} \sum_{i:y_i=1} x_i$
(C) $\hat{\mu}_1 = \frac{n_1}{n_2} \sum_{i:y_i=1} x_i$
(D) cannot be derived from the information given
Solution:

(A). The MLE for the mean of a Gaussian given the samples in that class is the average of those samples.

(2) [4 points, MC]

Suppose that we have the estimates $\hat{p}_j$ and $\hat{\mu}_j$ for all $j \in \{1, 2\}$. Given a data point x, which of the following is a valid procedure to predict the class label of x using LDA? (Note: $\hat{p}_1 = \hat{p}$, $\hat{p}_2 = 1 - \hat{p}$.)

(A) $\hat{y} = \arg\max_{j\in\{1,2\}} (x - \hat{\mu}_j)^\top \hat{\mu}_j + \log(\hat{p}_j)$

(B) $\hat{y} = \arg\max_{j\in\{1,2\}} (2x - \hat{\mu}_j)^\top \hat{\mu}_j + 2\log(\hat{p}_j)$

(C) $\hat{y} = \arg\max_{j\in\{1,2\}} (x - \hat{\mu}_j)^\top \hat{\mu}_j + \hat{p}_j$

(D) $\hat{y} = \arg\max_{j\in\{1,2\}} \exp\Big(-\frac{\|x-\hat{\mu}_j\|^2}{2\hat{p}_j}\Big)$

Solution:

(B). The discriminant function in LDA under identical covariance I has the form proportional to $(\hat{\mu}_j)^\top x - \frac{1}{2}\|\hat{\mu}_j\|^2 + \log(\hat{p}_j)$. After rearranging, that can be written in an equivalent form. The constant 2 factor emerges from expanding the negative norm term. (B) is the correct functional form for the argmax classification.

(3) [3 points, MC]

If

$$x^\top(\hat{\mu}_1 - \hat{\mu}_2) > \frac{1}{2}\Big(\hat{\mu}_1^\top\hat{\mu}_1 - \hat{\mu}_2^\top\hat{\mu}_2\Big),$$

x is classified as

(A) 1 for any $\hat{p}$

(B) 2 for any $\hat{p}$

(C) 1 if $\hat{p} = 0.5$

(D) 2 if $\hat{p} = 0.5$

Solution:

(C). The discriminant includes a term $\log\frac{\hat{p}}{1-\hat{p}}$. If $\hat{p} = 0.5$, then $\log(\frac{0.5}{0.5}) = 0$, so the condition reduces to $x^\top(\hat{\mu}_1 - \hat{\mu}_2) > \frac{1}{2}(\ldots)$. That corresponds to class 1 when this inequality is satisfied.

13 6. [15 points] EM algorithm

We consider a mixture of two exponential distributions, where a random variable $X \in \mathbb{R}$ is constructed as follows: First the class $Z \in \{0, 1\}$ is chosen according to

$$P(Z = 0) = \pi_0, \quad P(Z = 1) = \pi_1 = 1 - \pi_0.$$

Then the random variable X is sampled from an exponential distribution with parameter λ_0 or λ_1, depending on the class Z. Formally:

$$P(x \mid Z = 0) = \begin{cases} \lambda_0 e^{-\lambda_0 x}, & \text{if } x \geq 0, \\ 0, & \text{otherwise,} \end{cases}$$

$$P(x \mid Z = 1) = \begin{cases} \lambda_1 e^{-\lambda_1 x}, & \text{if } x \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

We are given n i.i.d. observations $x_{1:n} = (x_1, \ldots, x_n)$ from the model above. Let $z_{1:n} = (z_1, \ldots, z_n)$ be the latent variables, where $z_i = 0$ if the ith observation belongs to class 0, and $z_i = 1$ if it belongs to class 1. Denote $\theta = (\lambda_0, \lambda_1, \pi_0, \pi_1)$.

(a) [3 points, MC]

What is the correct expression for the complete-data likelihood $P(x_{1:n}, z_{1:n} \mid \theta)$?

(A) $\prod_{i=1}^n \big(\pi_0\lambda_0 e^{-\lambda_0 x_i}\big)^{1-z_i}\big(\pi_1\lambda_1 e^{-\lambda_1 x_i}\big)^{z_i}$

(B) $\prod_{i=1}^n \Big(\big(\pi_0\lambda_0 e^{-\lambda_0 x_i}\big)^{1-z_i} + \big(\pi_1\lambda_1 e^{-\lambda_1 x_i}\big)^{z_i}\Big)$

(C) $\prod_{i=1}^n \big(\lambda_0 e^{-\lambda_0 x_i}\big)^{1-z_i}\big(\lambda_1 e^{-\lambda_1 x_i}\big)^{z_i}$

(D) $\prod_{i=1}^n \Big(\big(\lambda_0 e^{-\lambda_0 x_i}\big)^{1-z_i} + \big(\lambda_1 e^{-\lambda_1 x_i}\big)^{z_i}\Big)$

Solution:

(A). The joint distribution for each datapoint includes both π_0 or π_1 (depending on z_i) and the exponential density. Then, across all n datapoints we multiply.

We run EM on this data. The vector $\theta^{(t)}$ denotes the parameters at iteration t. At iteration $t+1$, the E-step computes:

$$Q(\theta; \theta^{(t)}) := \mathbb{E}_{z_{1:n}}\big[\log P(x_{1:n}, z_{1:n} \mid \theta) \mid x_{1:n}, \theta^{(t)}\big].$$

Recall that

$$\gamma_1(x_i) = P(z_i = 1 \mid x_i, \theta^{(t)}), \quad \gamma_0(x_i) = 1 - \gamma_1(x_i).$$

(b) [3 points, MC]

What is the value of $Q(\theta; \theta^{(t)})$?

(A) $\sum_{i=1}^n \gamma_0(x_i)\big(\log(\lambda_0) - \lambda_0 x_i\big) + \gamma_1(x_i)\big(\log(\lambda_1) - \lambda_1 x_i\big)$

(B) $\sum_{i=1}^n \gamma_0(x_i)\big(\log(\pi_0\lambda_0) - \lambda_0 x_i\big) + \gamma_1(x_i)\big(\log(\pi_1\lambda_1) - \lambda_1 x_i\big)$

(C) $\sum_{i=1}^n \Big(\pi_0\lambda_0 e^{-\lambda_0 x_i}\Big)^{\gamma_0(x_i)} + \Big(\pi_1\lambda_1 e^{-\lambda_1 x_i}\Big)^{\gamma_1(x_i)}$

(D) $\sum_{i=1}^n \Big(\lambda_0 e^{-\lambda_0 x_i}\Big)^{\gamma_0(x_i)} + \Big(\lambda_1 e^{-\lambda_1 x_i}\Big)^{\gamma_1(x_i)}$

Solution:

(B). The complete-data log-likelihood is

$$\sum_{i=1}^n \big[(1 - z_i)\log(\pi_0\lambda_0) - \lambda_0 x_i\big] + \big[z_i \log(\pi_1\lambda_1) - \lambda_1 x_i\big].$$

Taking the expectation over z_i with weights $\gamma_0(x_i), \gamma_1(x_i)$ yields option (B).

(c) [3 points, MC]

What is the value of $\gamma_1(x_i)$?

(A) $\frac{\pi_1^{(t)}\lambda_1^{(t)} e^{-\lambda_1^{(t)} x_i}}{\pi_0^{(t)}\lambda_0^{(t)} e^{-\lambda_0^{(t)} x_i} + \pi_1^{(t)}\lambda_1^{(t)} e^{-\lambda_1^{(t)} x_i}}$

(B) $\pi_0^{(t)}\lambda_0^{(t)} e^{-\lambda_0^{(t)} x_i} + \pi_1^{(t)}\lambda_1^{(t)} e^{-\lambda_1^{(t)} x_i}$

(C) $\pi_1^{(t)}\lambda_1^{(t)} e^{-\lambda_1^{(t)} x_i}$

(D) $\frac{\lambda_1^{(t)} e^{-\lambda_1^{(t)} x_i}}{\lambda_0^{(t)} e^{-\lambda_0^{(t)} x_i} + \lambda_1^{(t)} e^{-\lambda_1^{(t)} x_i}}$

Solution:

(A). By Bayes' rule with mixture components. The denominator is the sum of numerator terms for each class.

(d) [3 points, MC]

What is the value of $\lambda_0^{(t+1)}$?

(A) $\frac{\sum_{i=1}^n \gamma_0(x_i)}{\sum_{i=1}^n x_i\,\gamma_0(x_i)}$

(B) $\frac{\sum_{i=1}^n x_i\,\gamma_0(x_i)}{\sum_{i=1}^n \gamma_0(x_i)}$

(C) $\frac{\sum_{i=1}^n x_i}{\sum_{i=1}^n \gamma_0(x_i)}$
(D) $\frac{\sum_{i=1}^n \gamma_0(x_i)}{\sum_{i=1}^n x_i}$
Solution:
(A). From setting the derivative of $Q(\theta;\theta^{(t)})$ to zero, solving for λ_0.

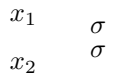
(e) [3 points, MC]

What is the value of $\pi_0^{(t+1)}$?
(A) $\sum_{i=1}^n \gamma_1(x_i)/n$
(B) $\sum_{i=1}^n \gamma_1(x_i)$
(C) $\sum_{i=1}^n \gamma_0(x_i)/n$
(D) $\sum_{i=1}^n \gamma_0(x_i)$
Solution:
(C). The M-step for π_0 is $\pi_0^{(t+1)} = \frac{1}{n}\sum_{i=1}^n \gamma_0(x_i)$. Because $\gamma_1(x_i) = 1 - \gamma_0(x_i)$, we get $\pi_1^{(t+1)} = 1 - \pi_0^{(t+1)}$.

14 1. (Neural network)

Consider the neural network given in the figure below. The numbers above the lines correspond to the weights of the connections. In the hidden layer, the activation function σ is applied.

Problem 2 (2018 Exam Question: ANN):
Consider the neural network given in the figure below. The numbers above the lines correspond to the weights of the connections. In the hidden layer, the activation function σ is applied and the network does not have any biases.



The diagram also shows weights labeled 1, 1, 0, 1, $\frac{1}{4}$, $\frac{3}{4}$ on each edge.

1. Read off the initial weights W_1 and w_2 from the network given above, so that the weights correspond to the following matrix notation of the neural network (with input $x = (x_1, x_2)$):
$$f(x; W_1, w_2) = w_2^\top \sigma(W_1 x).$$
2. You are given a data set $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ with n data points, where $x_i \in \mathbb{R}^2$ and $y_i \in \mathbb{R}$ for $i = 1, \ldots, n$. Write down the empirical risk $\hat{R}(D, W_1, w_2)$ for training the neural network using the squared loss with added L_2 regularization *only* on the output layer.
3. For training the neural network, the empirical risk function is often minimized using stochastic gradient descent (SGD). What is the difference between SGD and standard gradient descent in terms of computational complexity per gradient step?
4. Given W_1 and w_2 as in the diagram, find different weights $\widetilde{W}_1$ and $\widetilde{w}_2$ such that the resulting network has the same performance as the original network (for any activation function σ). In other words, find $\widetilde{W}_1$ and $\widetilde{w}_2$ so that
$$f(x; W_1, w_2) = f(x; \widetilde{W}_1, \widetilde{w}_2),$$
but $W_1 \neq \widetilde{W}_1$ and $w_2 \neq \widetilde{w}_2$.
5. Show that if σ is the ReLU activation function $\sigma(z) = \max(0, z)$, then the resulting network $f(x; W_1, w_2)$ is a piecewise linear function in x. Specifically, define 4 intervals on the real plane (where x_1 and x_2 are the axes) where the resulting function is linear, for the weights given in the diagram above.

Problem 3 (Expressiveness of Neural Networks):
In this question we will consider neural networks with sigmoid activation functions of the form
$$\phi(z) = \frac{1}{1 + \exp(-z)}.$$
If we denote by v_j^l the value of neuron j at layer l, its value is computed as
$$v_j^l = \phi\Big(w_0 + \sum_{i \in \text{Layer } l-1} w_{j,i}\, v_i^{l-1}\Big).$$
We will treat the final output $Y \in (0, 1)$ as 1 if it is greater than 0.5 and 0 otherwise.

1. Give three weights w_0, w_1, w_2 for a single unit with two inputs X_1 and X_2 that implements the logical OR function $Y = X_1 \vee X_2$.

Figure 1: (A reference figure for the above neural networks.)

Solution for the Neural Network Problems

a) We read off the weights:
$$W_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad w_2 = \begin{pmatrix} \frac{1}{4} \\ \frac{3}{4} \end{pmatrix}.$$
b) The empirical risk with squared loss and L_2 regularization on w_2 is:
$$\hat{R}(D, W_1, w_2) = \frac{1}{n}\sum_{i=1}^n \big(w_2^\top \sigma(W_1 x_i) - y_i\big)^2 \; + \; \| w_2 \|^2.$$
c) Stochastic gradient descent (SGD) uses a single data point (or a small batch) at each iteration, making the per-iteration cost $O(1)$ (or proportional to batch size). Standard (batch) gradient descent uses the entire dataset each step, costing $O(n)$ per iteration.
d) If σ is ReLU, then $a_1 = \max(x_1, 0)$ and $a_2 = \max(x_1 + x_2, 0)$. Hence $f = w_2^\top (a_1, a_2)^\top = \frac{1}{4}a_1 + \frac{3}{4}a_2$. This is piecewise linear in different regions of (x_1, x_2) space, specifically determined by $x_1 \leq 0$ or > 0 and $x_1 + x_2 \leq 0$ or > 0.
(Expressiveness) The single-unit OR function: choose w_0, w_1, w_2 so that $\phi(w_0 + w_1 X_1 + w_2 X_2) > 0.5$ exactly when X_1 or X_2 is 1.

15 2. (Dropout and Backpropagation)

Consider the following neural network with one input layer of 3 units, two hidden layers of 2 units, and one output layer. All units in the hidden layers use the sigmoid activation function $\sigma(z) = 1/(1+e^{-z})$, and no activation function is used in the output layer. For a training example (x, y) where $x \in \mathbb{R}^3$ and $y \in \mathbb{R}$, we use the loss $L = (y-f)^2$ at the output layer.

$$x_1 \quad x_2 \quad x_3 \qquad a_1^{(1)} \quad a_2^{(1)} \qquad a_1^{(2)} \quad a_2^{(2)} \qquad f$$

Weights between layers are denoted by $w_{ij}^{(k)}$, with the superscript (k) indicating the layer.

1. Given the training example (x_1, x_2, x_3) and label y, write down the sequence of calculations for computing the loss L. In particular, write down how to compute the intermediate values $a_i^{(1)}, a_i^{(2)}$, f, and L.

2. Consider using dropout to reduce overfitting. In particular, given training example (x_1, x_2, x_3) and label y, apply dropout for the 2nd hidden layer $(a_1^{(2)}, a_2^{(2)})$ with probability 0.4. Write down the expected value of the loss as a function of $a_1^{(2)}, a_2^{(2)}, w_1^{(3)}, w_2^{(3)}, y$.

3. At a certain iteration of SGD training with input (x_1, x_2, x_3) and label y, assume that $a_1^{(2)}$ gets dropped out and $a_2^{(2)}$ is kept after the forward pass. Write down the update rule for the weight $w_{21}^{(1)}$ during backpropagation. You may also use intermediate values $z_i^{(k)}$ where $\sigma\big(z_i^{(k)}\big) = a_i^{(k)}$.

Figure 2: (Dropout on the second hidden layer.)

Solution for Dropout and Backpropagation

(a) Forward pass for (x_1, x_2, x_3):

$$a_i^{(1)} = \frac{1}{1+\exp\big(-\sum_{j=1}^{3} w_{ji}^{(1)} x_j\big)},$$

$$a_i^{(2)} = \frac{1}{1+\exp\big(-\sum_{j=1}^{2} w_{ji}^{(2)}\, a_j^{(1)}\big)},$$

$$f = w_1^{(3)} a_1^{(2)} + w_2^{(3)} a_2^{(2)}, \quad L = (f-y)^2.$$

(b) With dropout on the 2nd hidden layer at probability 0.4, each unit $a_i^{(2)}$ is kept (multiplied by 1) with probability 0.4 or dropped (multiplied by 0) with probability 0.6. Let r_i be the random variable that is 1 if $a_i^{(2)}$ is kept and 0 if dropped. Then

$$f = w_1^{(3)}\big(a_1^{(2)} r_1\big) + w_2^{(3)}\big(a_2^{(2)} r_2\big).$$

The expected value of the loss is

$$\mathbb{E}[L] = \mathbb{E}\big[(f-y)^2\big] = \mathbb{E}\big[f^2\big] - 2\,y\,\mathbb{E}[f] + y^2.$$

Each r_i is Bernoulli(0.4). One can compute $\mathbb{E}[f]$ and $\mathrm{var}(f)$ accordingly and thus find $\mathbb{E}[L]$.

(c) If $a_1^{(2)}$ is dropped out and $a_2^{(2)}$ is kept, then $f = w_2^{(3)} a_2^{(2)}$. The update for $w_{21}^{(1)}$ uses backprop:

$$\frac{\partial L}{\partial w_{21}^{(1)}} = \frac{\partial L}{\partial f}\,\frac{\partial f}{\partial a_2^{(2)}}\,\frac{\partial a_2^{(2)}}{\partial a_1^{(1)}}\,\frac{\partial a_1^{(1)}}{\partial w_{21}^{(1)}}.$$

We then perform the usual gradient step

$$w_{21}^{(1)} \leftarrow w_{21}^{(1)} - \eta\,\frac{\partial L}{\partial w_{21}^{(1)}}.$$

16 3. (k-medians clustering)

Recall that for a given set of n points $x_i \in \mathbb{R}^d$, $i \in \{1, \ldots, n\}$, the k-means clustering algorithm aims to find the centers of k clusters $\mu = (\mu_1, \ldots, \mu_k)$ by minimizing

$$\hat{R}(\mu) = \sum_{i=1}^{n} \min_{j \in \{1,\ldots,k\}} \|\, x_i - \mu_j \|_2^2.$$

The algorithm iterates between assigning points to their closest cluster center, and then updating each center.

Now consider a different loss function

$$\hat{R}(\mu) = \sum_{i=1}^{n} \min_{j \in \{1,\ldots,k\}} \|\, x_i - \mu_j \|_1.$$

We have a similar iterative process:

1. Assignment step: assign each x_i to the cluster μ_j that minimizes $\|\, x_i - \mu_j \|_1$.

2. Update step: given the assigned points, choose the new center by minimizing the sum of $\|\cdot\|_1$ distances. This turns out to be the median in each coordinate. Hence this is called *k-medians.*

3. Show that this algorithm decreases the objective at each iteration and converges (similar argument to k-means).

4. In which situations would k-medians be preferred to k-means?

Solution for k-medians Clustering

(a) Assignment:

$$z_i \leftarrow \arg\min_{j \in \{1,\ldots,k\}} \|\, x_i - \mu_j \|_1.$$

(b) Update: For cluster j, let $C_j = \{\, i : z_i = j \,\}$. Minimizing $\sum_{i \in C_j} \|\, x_i - \mu_j \|_1$ is achieved by choosing each coordinate of μ_j to be the median of those coordinates among all points in C_j. Hence *k-medians.*

(c) Monotonic decrease in each step follows the same argument as in k-means: reassigning points cannot increase the total distance, and then updating centers optimally again reduces or keeps the same cost.

(d) k-medians is more robust to outliers than k-means, since the median is less sensitive than the mean to extreme values.

17 4. (Autoencoder)

Consider the neural autoencoder below where we compress a 3-dimensional input into a 2-dimensional latent space, with some activation function σ on the latent nodes and no activation function on the output layer. The input is (x_1, x_2, x_3), the latent representation is (a_1, a_2), and the reconstruction is (y_1, y_2, y_3).

$$x_1,\ x_2,\ x_3 \quad \to \quad (a_1, a_2) = \sigma(E\,x) \quad \to \quad (y_1, y_2, y_3) = D\,(a_1, a_2).$$

We define the reconstruction loss by summing the squared errors over each input dimension.

1. For a single data point $x = (x_1, x_2, x_3)$, write down the reconstruction loss $L(x)$ involving only x, z, D, and σ.

2. Given a dataset $X \in \mathbb{R}^{n\times 3}$ of $n \gg 3$ data points in $\mathbb{R}^3$, if $\sigma(z) = c\,z$ for some constant $c > 0$, find one solution of the optimization problem

$$E^*, D^* = \arg\min_{E,D} \frac{1}{n} \sum_{i=1}^{n} L\big(x_i\big).$$

   You are given the SVD $X = U\,\Sigma\,V^\top$, with $U \in \mathbb{R}^{n\times n}$, $\Sigma \in \mathbb{R}^{n\times 3}$, $V^\top \in \mathbb{R}^{3\times 3}$.

3. Let $\sigma(z) = \frac{1}{1+e^{-z}}$ (the sigmoid). Let $a = \sigma(z)$. Write down $\sigma'(z)$ in terms of a, not z.

4. Write down the update rule for weight e_{11} that does not involve z, during backpropagation of the reconstruction loss on a single sample $x = (x_1, x_2, x_3)$.

Solution for the Autoencoder

(a) Reconstruction loss for $x = (x_1, x_2, x_3)$:

$$L(x) = (y_1 - x_1)^2 + (y_2 - x_2)^2 + (y_3 - x_3)^2$$

$$= \sum_{i=1}^{3} \big(d_{1i}\,\sigma(z_1) + d_{2i}\,\sigma(z_2)\ -\ x_i\big)^2.$$

(b) If σ is linear (up to a constant c), the autoencoder essentially performs PCA when trying to minimize reconstruction error. One solution corresponds to picking E and D so that the 2-dimensional latent space is spanned by the principal components of X (the first 2 columns of V in the SVD).
(c) For the sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$, we have $\sigma'(z) = \sigma(z)\big(1 - \sigma(z)\big) = a(1-a)$.
(d) The update rule for e_{11} (the weight from x_1 to the first latent dimension) can be computed by chain rule. If $z_1 = e_{11}x_1 + e_{21}x_2 + e_{31}x_3$ (plus any relevant bias), then

$$\frac{\partial L}{\partial e_{11}} = \frac{\partial L}{\partial a_1}\,\frac{\partial a_1}{\partial z_1}\,\frac{\partial z_1}{\partial e_{11}} = \frac{\partial L}{\partial a_1}\,\big(a_1(1-a_1)\big)\,x_1,$$

and we then perform $e_{11} \leftarrow e_{11} - \eta\,\frac{\partial L}{\partial e_{11}}$ for the gradient step.

18 1. (Multiclass logistic regression)

The posterior probabilities for multiclass logistic regression can be given as a softmax transformation of hyperplanes, such that:

$$P(Y = k \mid X = x) = \frac{\exp(a_k^\top x)}{\sum_j \exp(a_j^\top x)}.$$

If we consider the use of maximum likelihood to determine the parameters a_k, we can take the negative logarithm of the likelihood function to obtain the cross-entropy error function for multiclass logistic regression:

$$E(a_1, \ldots, a_K) = -\ln\Big(\prod_{n=1}^{N}\prod_{k=1}^{K} P(Y = k \mid X = x_n)^{t_{nk}}\Big)$$

$$= -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk},$$

where $t_{nk} = 1\{\text{labelOf}(x_n) = k\}$, and $y_{nk} = P(Y = k \mid X = x_n)$.

18.0.1 (a)

For $j \in \{1, \ldots, K\}$, prove that

$$\frac{\partial\big(t_{nk} \ln y_{nk}\big)}{\partial a_j} = t_{nk}\big(\mathbf{1}\{k = j\} - y_{nj}\big)x_n.$$

18.0.2 (b)

Based on the result in (a), show that the gradient of the error function can be stated as

$$\nabla_{a_k} E(a_1, \ldots, a_K) = \sum_{n=1}^{N} \big[y_{nk} - t_{nk}\big]\,x_n.$$

Solution (sketch). Define $d_k = a_k^\top x$. Then

$$P(y = k \mid X = x) = \frac{\exp(a_k^\top x)}{\sum_j \exp(a_j^\top x)} = y_k(x).$$

You can first compute the derivatives

$$\frac{\partial y_k}{\partial d_j} = y_k\big(\mathbf{1}\{k = j\} - y_j\big).$$

Next, rewrite

$$\frac{\partial\big(t_{nk} \ln y_k(x_n)\big)}{\partial a_j} = t_{nk}\big(\mathbf{1}\{k = j\} - y_{nj}\big)x_n.$$

Summing across k and rearranging leads to

$$\nabla_{a_j} E(a_1, \ldots, a_K) = \sum_{n=1}^{N} \big[P(y = j \mid x_n) - t_{nj}\big]x_n,$$

yielding the stated result for $\nabla_{a_k} E$.

19 2. (Poisson Naive Bayes)

In this task we will use the Naive Bayes model for binary classification. Let $Y = \{0, 1\}$ be the set of labels and $X = \mathbb{N}^d$ a d-dimensional features space ($\mathbb{N} = \{0, 1, 2, \dots\}$). You are given a training set

$$D = \{(x_1, y_1), \dots, (x_n, y_n)\} \subset X \times Y.$$

19.0.1 (a)

Is the Naive Bayes model a generative or a discriminative model? Justify your answer.

Solution (sketch). Naive Bayes is a *generative* model because it models the joint data-generating distribution $P(X, Y)$ directly.

19.0.2 (b)

Let λ be a positive scalar, and assume that $z_1, \dots, z_m \in \mathbb{N}$ are m i.i.d. observations of a λ-Poisson distributed random variable. Find the maximum likelihood estimator for λ in this model. (Hint: A λ-Poisson distributed random variable Z takes values $k \in \mathbb{N}$ with probability

$$P(Z = k) = \frac{e^{-\lambda}\lambda^k}{k!}.$$

)

Solution (sketch). The MLE for λ in a Poisson(λ) distribution is its empirical mean:

$$\hat{\lambda} = \frac{1}{m}\sum_{j=1}^{m} z_j.$$

19.0.3 (c)

Let $p_0, p_1 \in [0, 1]$ be parameters with $p_0 + p_1 = 1$, and let $\lambda_0, \lambda_1 \in \mathbb{R}^d$ be vectors with non-negative components. Write down the joint distribution $P(X, Y)$ of the resulting model when training a Poisson Naive Bayes classifier by maximum likelihood estimation.

Solution (sketch). We estimate $p(y)$ as the empirical frequency of each class and each $\lambda_{y,j}$ as the empirical mean of the feature x_j within class y. The model is:

$$p(x, y) \;=\; p(y)\,\prod_{j=1}^{d} \frac{e^{-\lambda_{y,j}}(\lambda_{y,j})^{x_j}}{x_j!}.$$

19.0.4 (d)

Show that the predicted label for a new observation $x \in X$ is determined by a hyperplane. In other words,

$$y_{\text{pred}} \;=\; \big[a^\top x \;\geq\; b\big]$$

for some $a \in \mathbb{R}^d$, $b \in \mathbb{R}$.

Solution (sketch). We compare $P(y = 0 \mid x)$ and $P(y = 1 \mid x)$. Equivalently, we set $p(x, 0) = p(x, 1)$, substitute the Poisson forms, and take the log. The condition simplifies to a linear function in x,

$$\log\Big(\frac{p_0}{p_1}\Big) + \sum_{j=1}^{d}\Big(\lambda_{1,j} - \lambda_{0,j} + x_j \log(\lambda_{0,j}/\lambda_{1,j})\Big) \;=\; 0,$$

which corresponds to a hyperplane in x.

19.0.5 (e)

Define a cost function $c : Y \times Y \to \mathbb{R}$ so that $c(y_{\text{pred}}, y_{\text{true}})$ is the cost of predicting y_{pred} given the true label y_{true}. The Bayes optimal predictor for a cost function $c(\cdot, \cdot)$ is

$$y_{\text{Bayes}} = \arg\min_{y \in Y} \mathbb{E}_Y[c(Y, y) \mid X = x].$$

Write down a cost function such that this Bayes optimal predictor coincides with the predictor that minimizes misclassification probability.

Solution (sketch). Use the 0/1 loss:

$$c(y_{\text{pred}}, y_{\text{true}}) = \mathbf{1}\{y_{\text{pred}} \neq y_{\text{true}}\}.$$

20 3. (Poisson Latent Dirichlet Allocation)

We extend the Poisson Naive Bayes model to a Poisson LDA framework. Each data observation is represented by a d-dimensional count vector $x \in \mathbb{N}^d$. Assume there are K latent topics. The generative process for an observation is:

$$\theta \sim \text{Dirichlet}(\alpha) \quad \text{and} \quad x_j \sim \text{Poisson}\Big(\sum_{k=1}^{K} \theta_k\,\lambda_{k,j}\Big),$$

where $\alpha \in \mathbb{R}_+^K$, $\theta = (\theta_1, \dots, \theta_K)$, and $\lambda_{k,j} \geq 0$ for all k, j.

20.0.1 (a) Joint Likelihood

Write down

$$p(x, \theta \mid \alpha, \{\lambda_{k,j}\}).$$

Solution (sketch).

$$p(x, \theta \mid \alpha, \{\lambda_{k,j}\}) = p(\theta \mid \alpha)\,\prod_{j=1}^{d} p\big(x_j \mid \theta, \{\lambda_{k,j}\}\big).$$

Using the Dirichlet density and the Poisson distribution:

$$p(\theta \mid \alpha) = \frac{1}{B(\alpha)}\prod_{k=1}^{K}\theta_k^{\alpha_k - 1},$$

$$p\big(x_j \mid \theta, \{\lambda_{k,j}\}\big) = \frac{\exp\big(-\sum_{k=1}^{K}\theta_k\lambda_{k,j}\big)\big(\sum_{k=1}^{K}\theta_k\lambda_{k,j}\big)^{x_j}}{x_j!}.$$

20.0.2 (b) Marginal Likelihood

Derive

$$p(x \mid \alpha, \{\lambda_{k,j}\}) = \int_{\Delta_K} p(x, \theta \mid \alpha, \{\lambda_{k,j}\})\, d\theta,$$

and briefly explain why this integral is generally intractable.

Solution (sketch).

$$p(x \mid \alpha, \{\lambda_{k,j}\}) = \int_{\Delta_K} \frac{1}{B(\alpha)} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1} \prod_{j=1}^{d} \frac{\exp\!\big(-\sum_{k=1}^{K} \theta_k \lambda_{k,j}\big)\big(\sum_{k=1}^{K} \theta_k \lambda_{k,j}\big)^{x_j}}{x_j!}\, d\theta.$$

The terms $\sum_{k=1}^{K} \theta_k \lambda_{k,j}$ appear inside exponentials and powers, coupling $\theta_1, \ldots, \theta_K$ in a way that prevents a closed-form solution.

20.0.3 (c) Parameter Estimation (EM Update)

Suppose we have N data observations $\{x^{(i)}\}_{i=1}^{N}$. In an EM algorithm for estimating $\{\lambda_{k,j}\}$, assume that in the E-step for observation i, you compute an approximate posterior over $\theta^{(i)}$ and define

$$\gamma_k^{(i)} = \mathbb{E}_q[\theta_k].$$

Derive an update equation for $\lambda_{k,j}$ based on maximizing the expected complete-data log-likelihood.

Solution (sketch). The expected complete-data log-likelihood is

$$Q(\{\lambda_{k,j}\}) = \sum_{i=1}^{N} \sum_{j=1}^{d} \Big[-\sum_{k=1}^{K} \gamma_k^{(i)} \lambda_{k,j} + x_j^{(i)} \ln\Big(\sum_{k=1}^{K} \gamma_k^{(i)} \lambda_{k,j}\Big)\Big] + \text{const.}$$

Taking derivatives and setting to zero yields a multiplicative update such as

$$\lambda_{k,j}^{\text{new}} = \lambda_{k,j}^{\text{old}} \frac{\sum_{i=1}^{N} x_j^{(i)}\, \gamma_k^{(i)} \Big/ \big(\sum_{k'} \gamma_{k'}^{(i)} \lambda_{k',j}^{\text{old}}\big)}{\sum_{i=1}^{N} \gamma_k^{(i)}}.$$

21 4. (Mixture models and EM algorithm)

Consider a 1-dimensional Gaussian Mixture Model (GMM) with 2 clusters and parameters $(\mu_1, \sigma_1^2, \mu_2, \sigma_2^2, w_1, w_2)$. We have mixing weights w_1, w_2 and cluster centers (means) μ_1, μ_2, variances σ_1^2, σ_2^2. Given a dataset $D = \{x_1, x_2, x_3\} \subset \mathbb{R}$, each iteration of EM introduces a latent variable $z_i \in \{1, 2\}$ with indicators $z_{ic} = 1$ if $z_i = c$ and 0 otherwise.

21.0.1 (a) Complete-data log-likelihood

$$\ln f(D, Z \mid (\mu_1, \sigma_1^2, \mu_2, \sigma_2^2, w_1, w_2)) = \sum_{i=1}^{3} \sum_{c=1}^{2} z_{ic} \Big[\ln w_c + \ln \mathcal{N}(x_i; \mu_c, \sigma_c^2)\Big].$$

Assume our dataset is $x_1 = 1$, $x_2 = 10$, $x_3 = 20$, and at some step, the E-step yields

$$R = \begin{pmatrix} 1 & 0 \\ 0.4 & 0.6 \\ 0 & 1 \end{pmatrix},$$

where r_{ic} is the probability of x_i belonging to cluster c.

21.0.2 (b) Updating w_1, w_2

$$w_1' = \frac{1}{3}\big(1 + 0.4 + 0\big) = \frac{1.4}{3}, \quad w_2' = \frac{1}{3}\big(0 + 0.6 + 1\big) = \frac{1.6}{3}.$$

21.0.3 (c) Updating μ_1, μ_2

$$\mu_1' = \frac{1 \cdot 1 + 0.4 \cdot 10 + 0 \cdot 20}{1.4} = \frac{5}{1.4},$$
$$\mu_2' = \frac{0 \cdot 1 + 0.6 \cdot 10 + 1 \cdot 20}{1.6} = \frac{26}{1.6}.$$

21.0.4 (d) Updating σ_1^2, σ_2^2

$$(\sigma_1^2)' = \frac{1}{1.4}\Big[1\big(1 - \tfrac{5}{1.4}\big)^2 + 0.4\big(10 - \tfrac{5}{1.4}\big)^2 + 0\big(20 - \tfrac{5}{1.4}\big)^2\Big] = \frac{810}{49},$$
$$(\sigma_2^2)' = \frac{1}{1.6}\Big[0\big(1 - \tfrac{26}{1.6}\big)^2 + 0.6\big(10 - \tfrac{26}{1.6}\big)^2 + 1\big(20 - \tfrac{26}{1.6}\big)^2\Big] = \frac{375}{16}.$$

21.0.5 (e) Responsibilities sum to 1

For each x_i, $r_{i1} + r_{i2} = 1$. This follows from

$$r_{ic} = \frac{w_c \mathcal{N}(x_i;\, \mu_c, \sigma_c^2)}{w_1 \mathcal{N}(x_i;\, \mu_1, \sigma_1^2) + w_2 \mathcal{N}(x_i;\, \mu_2, \sigma_2^2)},$$

and is essential for interpreting r_{ic} as a valid probability.

21.0.6 (f) Pitfall and mitigation

A common pitfall is that poor initialization (e.g., means, variances) can make EM converge to a suboptimal local maximum. A straightforward mitigation is to run EM multiple times from different (random) starting points or to use a better initialization method (e.g., k-means).

22 5. (A different perspective on EM algorithm)

Let $P(X, Z)$ be any model with observed variables X and latent variables Z. Assume Z is discrete, taking values in $\{1, 2, \ldots, m\}$. For observed X, we want to maximize

$$\ell(\theta) = \log P(x; \theta) = \log \sum_{z=1}^{m} P(x, z; \theta).$$

Define $Q(Z)$ to be any distribution over the latent variables Z.

22.0.1 (a)

Show that if $Q(z) > 0$ whenever $P(x, z) > 0$, then

$$\ell(\theta) \;\geq\; \mathbb{E}_Q[\log P(X, Z)] - \sum_{z=1}^{m} Q(z) \log Q(z).$$

(Hint: use Jensen's inequality.)

Solution (sketch).

$$\ell(\theta) = \log \sum_{z=1}^{m} P(x, z; \theta) = \log \sum_{z=1}^{m} \frac{P(x, z; \theta)}{Q(z)} Q(z)$$

$$= \log \mathbb{E}_{Z \sim Q}\Big[\frac{P(x, Z; \theta)}{Q(Z)}\Big] \;\geq\; \mathbb{E}_Q\Big[\log \frac{P(x, Z; \theta)}{Q(Z)}\Big]$$

$$= \mathbb{E}_Q[\log P(x, Z; \theta)] - \sum_{z=1}^{m} Q(z) \log Q(z).$$

22.0.2 (b)

Show that for fixed θ, the above lower bound is maximized by $Q^*(Z) = P(Z \mid X; \theta)$, and that the bound then holds with equality.

Solution (sketch). Maximize

$$F(Q) = \sum_{z=1}^{m} Q(z) \log P(x, z; \theta) \;-\; \sum_{z=1}^{m} Q(z) \log Q(z),$$

subject to $\sum_z Q(z) = 1$. Using Lagrange multipliers, one finds

$$Q^*(z) = \frac{P(x, z; \theta)}{P(x; \theta)} = P(z \mid x; \theta).$$

Substituting back shows the bound is tight.

22.0.3 (c)

Derive

$$\mathrm{KL}\big(Q \,\|\, P(\cdot \mid x; \theta)\big) = \sum_{z=1}^{m} Q(z) \,\log\Big(\frac{Q(z)}{P(z \mid x; \theta)}\Big).$$

Explain how this relates to the lower bound on $\ell(\theta)$.

Solution (sketch). We have

$$\ell(\theta) = L(Q, \theta) + \mathrm{KL}\big(Q \,\|\, P(\cdot \mid x; \theta)\big),$$

where

$$L(Q, \theta) = \mathbb{E}_Q[\log P(x, Z; \theta)] + H(Q).$$

Since $\mathrm{KL}(\cdot\|\cdot) \geq 0$, $L(Q, \theta) \leq \ell(\theta)$, and equality holds if $Q = P(\cdot \mid x; \theta)$.

22.0.4 (d)

Discuss the impact if we choose $Q(Z)$ differently from $P(Z \mid x; \theta)$ on tightness of the bound and convergence.

Solution (sketch). Any other choice of $Q(Z)$ yields a positive KL divergence, so the bound becomes looser. While one can still view each EM step as maximizing a lower bound, a suboptimal $Q(Z)$ can slow or degrade convergence.

22.0.5 (e)

Show that alternating optimization over Q and θ corresponds to the EM procedure. What does this imply for monotonicity and convergence?

Solution (sketch). Optimizing w.r.t. Q (for fixed θ) recovers $Q^*(Z) = P(Z \mid x; \theta)$ (the E-step), while optimizing w.r.t. θ (for fixed Q) is the M-step. Each step increases or preserves the lower bound, guaranteeing that the log-likelihood is non-decreasing until convergence to a local optimum.