STA 314: Statistical Methods for Machine Learning I

Lecture 10 - Probabilistic Models

Chris J. Maddison

University of Toronto

Today

- Wrapping up inference and decision-making.
- Gaussian generative models.

MLE Recap

- Last time we discussed the maximum likelihood estimation view of machine learning:
- Specify a family of distributions $p(\mathbf{x}|\theta)$ parameterized by $\theta \in \Theta$.
- Observe a data set $\mathcal{D} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$.
- Under an IID assumption, MLE corresponds to

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta \in \Theta} \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)}|\theta)$$

MLE issue: Data Sparsity

- Maximum likelihood has a pitfall: if you have too little data, it can overfit.
- E.g., what if you flip the coin twice and get H both times?

$$\theta_{\mathrm{ML}} = \frac{N_H}{N_H + N_T} = \frac{2}{2+0} = 1$$

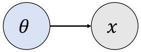
- Because it never observed T, it assigns this outcome probability 0. This problem is known as data sparsity.

Bayesiam Parameter Estimation

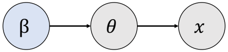
- Somehow we want to reflect our uncertainty in the true value of θ.
- Maybe the problem was that we summarized $\mathcal{D}$ in a single setting of the parameters $\hat{\boldsymbol{\theta}}_{\text{MLE}}$
- What if we summarized using a distribution? This will allow us to reflect that fact that we want to consider a variety of possible parameters weighted by some probability. This is the spirit behind Bayesian inference.

Bayesian Parameter Estimation

- In maximum likelihood, the observations are treated as random variables, but the parameters are not.



- The Bayesian approach treats the parameters as random variables as well. β is the set of parameters in the prior distribution of θ.



- To define a Bayesian model, we need to specify two distributions:
  - The prior distribution $p(\boldsymbol{\theta})$, which encodes our beliefs about the parameters *before* we observe the data
  - The likelihood $p(\mathcal{D} \,|\, \boldsymbol{\theta})$, same as in maximum likelihood

Bayesian Parameter Estimation

- The posterior distribution is the distribution that we will use to summarize $\mathcal{D}$.
- Using Bayes' Rule:

$$p(\boldsymbol{\theta} \mid \mathcal{D}) = \frac{p(\boldsymbol{\theta})p(\mathcal{D} \mid \boldsymbol{\theta})}{\int p(\boldsymbol{\theta}')p(\mathcal{D} \mid \boldsymbol{\theta}') \, \mathrm{d}\boldsymbol{\theta}'}.$$

- We rarely ever compute the denominator explicitly. In general, it is computationally intractable.

Bayesian Parameter Estimation

- Let's revisit the coin example. We already know the likelihood:

$$L(\theta) = p(\mathcal{D}|\theta) = \theta^{N_H}(1-\theta)^{N_T}$$

- It remains to specify the prior $p(\theta)$.
  - We can choose an uninformative prior, which assumes as little as possible. A reasonable choice is the uniform prior.
  - But our experience tells us 0.5 is more likely than 0.99. One particularly useful prior that lets us specify this is the beta distribution:

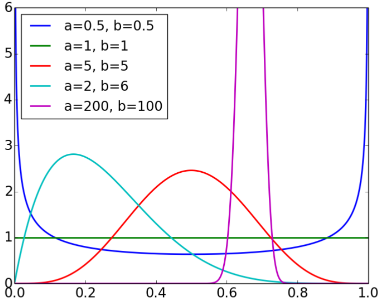
$$p(\theta; a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\,\theta^{a-1}(1-\theta)^{b-1}.$$

  - This notation for proportionality lets us ignore the normalization constant:

$$p(\theta; a, b) \propto \theta^{a-1}(1-\theta)^{b-1}.$$

Bayesian Parameter Estimation

- Beta distribution for various values of a, b:



- Some observations:
  - The expectation $\mathbb{E}[\theta] = a/(a+b)$ (easy to derive).
  - The distribution gets more peaked when a and b are large.
  - The uniform distribution is the special case where $a = b = 1$.
- The beta distribution is used for is as a prior for the Bernoulli distribution.

Bayesian Parameter Estimation

- Computing the posterior distribution:

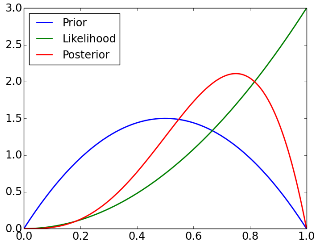
$$\begin{aligned} p(\boldsymbol{\theta} \,|\, \mathcal{D}) &\propto p(\boldsymbol{\theta}) p(\mathcal{D} \,|\, \boldsymbol{\theta}) \\ &\propto \left[\theta^{a-1}(1-\theta)^{b-1}\right]\left[\theta^{N_H}(1-\theta)^{N_T}\right] \\ &= \theta^{a-1+N_H}(1-\theta)^{b-1+N_T}. \end{aligned}$$

- This is just a beta distribution with parameters $N_H + a$ and $N_T + b$.
- The parameters a and b of the prior can be thought of as pseudo-counts.
  - The reason this works is that the prior and likelihood have the same functional form. This phenomenon is known as conjugacy (conjugate priors), and it's very useful.

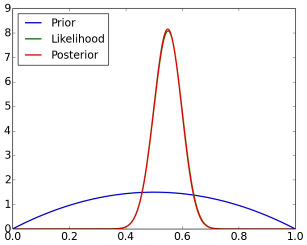
Bayesian Parameter Estimation

Bayesian inference for the coin flip example:

Small data setting
$N_H = 2,\ N_T = 0$



Large data setting
$N_H = 55,\ N_T = 45$



When you have enough observations, the data overwhelm the prior.

Bayesian Parameter Estimation

- What do we actually do with the posterior?
- The posterior predictive distribution is the distribution over future observables given the past observations. We compute this by marginalizing out the parameter(s):

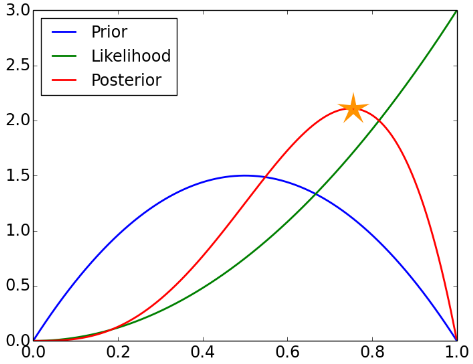
$$p(\mathcal{D}' \mid \mathcal{D}) = \int p(\boldsymbol{\theta} \mid \mathcal{D}) p(\mathcal{D}' \mid \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta}. \tag{1}$$

- For the coin flip example:

$$\begin{aligned} \theta_{\text{pred}} &= \Pr(\mathbf{x}' = H \mid \mathcal{D}) \\ &= \int p(\theta \mid \mathcal{D}) \Pr(\mathbf{x}' = H \mid \theta) \, \mathrm{d}\theta \\ &= \int \mathrm{Beta}(\theta; N_H + a, N_T + b) \cdot \theta \, \mathrm{d}\theta \\ &= \mathbb{E}_{\mathrm{Beta}(\theta; N_H + a, N_T + b)}[\theta] \\ &= \frac{N_H + a}{N_H + N_T + a + b}, \end{aligned} \tag{2}$$

Maximum A-Posteriori Estimation

- Maybe we can summarize the posterior using a single value?
- We can do this with maximum a-posteriori (MAP) estimation: find the most likely parameter settings under the posterior to summarize the posterior.



Maximum A-Posteriori Estimation

- This converts the Bayesian parameter estimation problem into a maximization problem

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{\mathrm{MAP}} &= \arg\max_{\boldsymbol{\theta}} \; p(\boldsymbol{\theta} \mid \mathcal{D}) \\ &= \arg\max_{\boldsymbol{\theta}} \; p(\boldsymbol{\theta}, \mathcal{D}) \\ &= \arg\max_{\boldsymbol{\theta}} \; p(\boldsymbol{\theta})\, p(\mathcal{D} \mid \boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \; \log p(\boldsymbol{\theta}) + \log p(\mathcal{D} \mid \boldsymbol{\theta})\end{aligned}$$

- We already saw an example of this in the homework.

Maximum A-Posteriori Estimation

- Joint probability in the coin flip example:

$$\begin{aligned}\log p(\theta, \mathcal{D}) &= \log p(\theta) + \log p(\mathcal{D} \mid \theta) \\ &= \text{Const} + (a-1)\log\theta + (b-1)\log(1-\theta) + N_H \log\theta + N_T \log(1-\theta) \\ &= \text{Const} + (N_H + a - 1)\log\theta + (N_T + b - 1)\log(1-\theta)\end{aligned}$$

- Maximize by finding a critical point

$$0 = \frac{\mathrm{d}}{\mathrm{d}\theta} \log p(\theta, \mathcal{D}) = \frac{N_H + a - 1}{\theta} - \frac{N_T + b - 1}{1 - \theta}$$

- Solving for θ,

$$\hat{\theta}_{\mathrm{MAP}} = \frac{N_H + a - 1}{N_H + N_T + a + b - 2}$$

Maximum A-Posteriori Estimation

Comparison of estimates in the coin flip example:

	Formula	$N_H = 2, N_T = 0$	$N_H = 55, N_T = 45$
$\hat{\theta}_{\mathrm{ML}}$	$\frac{N_H}{N_H+N_T}$	1	$\frac{55}{100} = 0.55$
$\hat{\theta}_{\mathrm{MAP}}$	$\frac{N_H+a-1}{N_H+N_T+a+b-2}$	$\frac{3}{4} = 0.75$	$\frac{56}{102} \approx 0.549$

$\hat{\theta}_{\mathrm{MAP}}$ assigns nonzero probabilities as long as $a, b > 1$.

Recap

- We took a probabilistic perspective on parameter estimation.
- We modeled a biased coin as a Bernoulli random variable with parameter θ, which we estimated using:
  - maximum likelihood estimation:
    $\hat{\theta}_{\mathrm{ML}} = \arg\max_\theta p(\mathcal{D} \mid \theta)$
  - Bayesian posterior:
    $p(\theta \mid \mathcal{D}) \propto p(\boldsymbol{\theta}) p(\mathcal{D} \mid \theta)$ by Bayes' Rule.
  - Maximum a-posteriori (MAP) estimation:
    $\hat{\theta}_{\mathrm{MAP}} = \arg\max_\theta \; p(\theta \mid \mathcal{D})$
- We also saw parameter estimation in context of a Naïve Bayes classifier.
- Today we will continue developing the probabilistic perspective:
  - Gaussian Discriminant Analysis (GDA): Use Gaussian generative model of the data for classification, similar in spirit to Naive bayes
  - Gaussian Mixture Model (GMM): Gaussian generative model view of clustering

GDA: the data

- N inputs, D continuous features

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_D^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_D^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \cdots & x_D^{(N)} \end{bmatrix} = \begin{bmatrix} (\mathbf{x}^{(1)})^\top \\ (\mathbf{x}^{(2)})^\top \\ \vdots \\ (\mathbf{x}^{(N)})^\top \end{bmatrix} \in \mathbb{R}^{N\times D}$$

- N integer targets

$$\mathbf{t} = \begin{bmatrix} t^{(1)} \\ t^{(2)} \\ \vdots \\ t^{(N)} \end{bmatrix} \in \{1, \dots, K\}^N$$

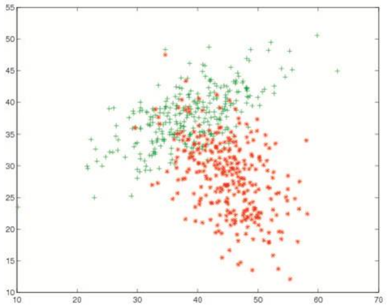
- Independent and identically distributed $\mathbf{x}^{(i)}, t^{(i)}$.

GDA: motivation

- Generative models - model $p(\mathbf{x}|t)$
- Instead of trying to separate classes, try to model what each class "looks like".
- Recall that $p(\mathbf{x}|t)$ may be very complex and requite many parameters to specify.
- Naive bayes used a conditional independence assumption. What else could we do? Choose a simple distribution.
- Gaussians are "simple" distributions and today we will look at models that use a Gaussian to specify $p(\mathbf{x}|t)$.

GDA: Gaussian model

- The Gaussian is a "simple" model of continuous data with elliptical shape.
- Gaussian Discriminant Analysis in its general form assumes that $p(\mathbf{x}|t)$ is distributed according to a multivariate normal (Gaussian) distribution
- If $p(\mathbf{x}|t)$ is elliptical, then Gaussian Discriminant Analysis is a good choice.
- Observation per patient: White blood cell count & glucose value.



GDA

- Assume the prior is categorical over K possible classes.

$$p(t) = \theta_t$$

  such that $\sum_{k=1}^{K} \theta_k = 1$.
- Model each class condtional $p(\mathbf{x}|t)$ with a multivariate Gaussian.

$$p(\mathbf{x}|t) = \frac{1}{(2\pi)^{d/2}|\Sigma_t|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_t)^T \Sigma_t^{-1}(\mathbf{x}-\boldsymbol{\mu}_t)\right]$$

  where $|\Sigma_t|$ denotes the determinant of the matrix, and D is dimension of $\mathbf{x}$.
- Note: I am ommitting the dependence of the probabilities on the parameters, but they are there.
- Note: each class k has associated mean vector $\boldsymbol{\mu}_k$ and covariance matrix Σ_k
- Note: Σ_k requires $D(D+1)/2$ parameters to specify, $\boldsymbol{\mu}_k$ requires D and θ requires $K-1$.
- Total parameters needed to specify the model: $KD(D+3)/2 + K - 1$.

GDA: Learning

- We will fit the parameters of GDA using maximum likelihood.
- Let $\phi = (\theta, \boldsymbol{\mu}_1, \Sigma_1, \ldots, \boldsymbol{\mu}_K, \Sigma_K)$, then log-likelihood function:

$$\ell(\phi) = \sum_{i=1}^{N} \log \theta_{t^{(i)}} + \underbrace{- \log(2\pi)^{d/2}}_{\text{constant}} - \log |\Sigma_{t^{(i)}}|^{1/2} - \frac{1}{2}(\mathbf{x}^{(i)} - \boldsymbol{\mu}_{t^{(i)}})^T \Sigma_{t^{(i)}}^{-1} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_{t^{(i)}})$$

Optional intuition building: why does $|\boldsymbol{\Sigma}|^{1/2}$ show up in the Gaussian density $p(\mathbf{x})$?

Hint: determinant is product of eigenvalues

GDA: MLE

- The maximum likelihood estimate of θ is the same as the homework:

$$\hat{\theta}_k = \frac{1}{N}\sum_{i=1}^{N} 1[t^{(i)} = k]$$

GDA: MLE

- The MLE of $\boldsymbol{\mu}_k$ is similar to the tutorial last week.

$$0 = \frac{\mathrm{d}\ell}{\mathrm{d}\boldsymbol{\mu}_k} = -\sum_{i=1}^{N} \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\mu}_k} \frac{1}{2} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_{t^{(i)}})^T \Sigma_{t^{(i)}}^{-1} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_{t^{(i)}})$$

$$= -\sum_{i=1}^{N} 1[t^{(i)} = k] \Sigma_k^{-1} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k) = 0$$

  Here we use the identity $\partial \mathbf{x}^\top \mathbf{A} \mathbf{x} / \partial \mathbf{x} = 2\mathbf{A}\mathbf{x}$ for symmetric $\mathbf{A}$.

- Solving we get the sample mean of the observed values of class k,

$$\hat{\boldsymbol{\mu}}_k = \sum_{i=1}^{N} \frac{1[t^{(i)} = k] \mathbf{x}^{(i)}}{\sum_{i=1}^{N} 1[t^{(i)} = k]}$$

GDA: MLE

- We can do a similar calculation for the covariance matrices Σ. The derivation in multivariate case is tedious, so we skip it. But it is good practice to derive this in one dimension. (See supplement at end of slides.)
- Setting the *partial* derivatives to zero, just like before, we get:

$$\hat{\Sigma}_k = \frac{1}{\sum_{i=1}^N 1[t^{(i)} = k]} \sum_{i=1}^N 1[t^{(i)} = k](\mathbf{x}^{(i)} - \hat{\mu}_k)(\mathbf{x}^{(i)} - \hat{\mu}_k)^T$$

  Note this is a bit different from the empirical covariance matrix we defined for PCA, but it's close enough!

GDA: MLE summary

- MLEs for GDA:

$$\begin{aligned}
\hat{\theta}_k &= \frac{1}{N}\sum_{i=1}^{N} 1[t^{(i)} = k] \\
\hat{\boldsymbol{\mu}}_k &= \frac{\sum_{i=1}^{N} 1[t^{(i)} = k]\cdot \mathbf{x}^{(i)}}{\sum_{i=1}^{N} 1[t^{(i)} = k]} \\
\hat{\Sigma}_k &= \frac{1}{\sum_{i=1}^{N} 1[t^{(i)} = k]}\sum_{i=1}^{N} 1[t^{(i)} = k](\mathbf{x}^{(i)} - \hat{\mu}_k)(\mathbf{x}^{(i)} - \hat{\mu}_k)^T
\end{aligned}$$

GDA: Decision boundary

- GDA decision boundary is based on class posterior.
- Given the MLEs $\hat{\phi} = (\hat{\theta}, \hat{\boldsymbol{\mu}}_1, \hat{\Sigma}_1, \ldots, \hat{\boldsymbol{\mu}}_K, \hat{\Sigma}_K)$, we make classification predictions by computing

$$y(\mathbf{x}) = \arg\max_t p_{\hat{\phi}}(t|\mathbf{x})$$

- Note: I am explicitly including the dependence on the MLE $\hat{\phi}$ to be clear which parameters we use to predict.
- Let's study the decision boundary in the binary special case.

GDA: Decision boundary, $K = 2$

- In the binary special case we have

$$y(\mathbf{x}) = 1 \text{ if } p_{\hat{\phi}}(1|\mathbf{x}) > p_{\hat{\phi}}(0|\mathbf{x}) \iff \log p_{\hat{\phi}}(1|\mathbf{x}) > \log 0.5$$

- This is starting to look a lot like a linear classifier.
- Let's investigate. The decision boundary is the set of points $\mathbf{x}$ where $\log p_{\hat{\phi}}(1|\mathbf{x}) = \log p_{\hat{\phi}}(0|\mathbf{x})$.

GDA: Decision boundary, $K = 2$

- Specifically,

$$\begin{aligned}\log p_{\hat{\phi}}(t|\mathbf{x}) &= \log p_{\hat{\phi}}(\mathbf{x}|t) + \log p_{\hat{\phi}}(t) - \log p_{\hat{\phi}}(\mathbf{x}) \\ &= -\frac{D}{2}\log(2\pi) - \frac{1}{2}\log|\hat{\Sigma}_t^{-1}| - \frac{1}{2}(\mathbf{x} - \hat{\boldsymbol{\mu}}_t)^T\hat{\Sigma}_t^{-1}(\mathbf{x} - \hat{\boldsymbol{\mu}}_t) \\ &\quad + \log\hat{\theta}_t - \log p_{\hat{\phi}}(\mathbf{x})\end{aligned}$$

- So, the decision boundaries of the GDA classifier are the points $\mathbf{x}$ where

$$(\mathbf{x} - \hat{\boldsymbol{\mu}}_1)^T\hat{\Sigma}_1^{-1}(\mathbf{x} - \hat{\boldsymbol{\mu}}_1) = (\mathbf{x} - \hat{\boldsymbol{\mu}}_0)^T\hat{\Sigma}_0^{-1}(\mathbf{x} - \hat{\boldsymbol{\mu}}_0) + C$$

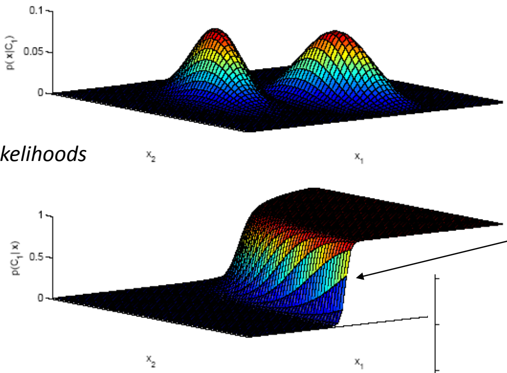
  for some constant (in $\mathbf{x}$) C.

- Re-writing this, we find a quadratic relation in $\mathbf{x} \implies$ quadratic (conic) decision boundary:

$$\mathbf{x}^T(\hat{\Sigma}_1^{-1} - \hat{\Sigma}_0^{-1})\mathbf{x} + 2\mathbf{x}^T(\hat{\Sigma}_0^{-1}\hat{\boldsymbol{\mu}}_0 - \hat{\Sigma}_1^{-1}\hat{\boldsymbol{\mu}}_1) + C' = 0$$

  Here C' is another constant (in $\mathbf{x}$).

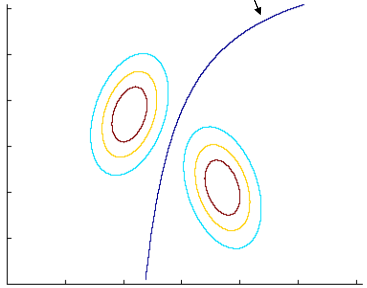
Decision boundary for different covariances



likelihoods

posterior for t_1

discriminant:
$P(t_1|\boldsymbol{x}) = 0.5$



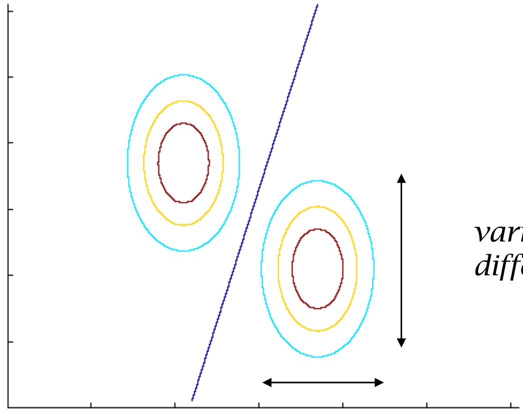
GDA: Decision boundary, $K = 2$

- If $\hat{\Sigma}_1^{-1} = \hat{\Sigma}_0^{-1}$, then we get a linear decision boundary!

$$\mathbf{x}^T \hat{\Sigma}_0^{-1} (\hat{\boldsymbol{\mu}}_0 - \hat{\boldsymbol{\mu}}_1) + C^{''} = 0$$

Here $C^{''}$ is yet another constant (in $\mathbf{x}$).

Decision boundary for the same covariance



variances may be different

Simplifying the Model

- It can sometimes be valuable to share the covariance matrix between classes, i.e. $\Sigma_k = \Sigma_l$.
  - For GDA, if $\mathbf{x}$ is high-dimensional, then covariance matrix has many parameters, i.e., $D(D+1)/2$, so sharing covariance matrices can reduce the total number of parameters.
- MLE in this case:

$$\hat{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)} - \hat{\mu}_{t^{(i)}})(\mathbf{x}^{(i)} - \hat{\mu}_{t^{(i)}})^T$$

- Linear decision boundary (at home: verify this mathematically!).
  - In Scikit-Learn this is called "Linear Discriminant Analysis" (LDA)

Gaussian Discriminative Analysis vs Logistic Regression

- Binary classification: If you examine $p(t|\mathbf{x})$ under GDA and assume $\Sigma_0 = \Sigma_1 = \Sigma$, you will find that it looks like this:

$$p(t|\mathbf{x}, \theta, \mu_0, \mu_1, \Sigma) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

  where $\mathbf{w}$ is an appropriate function of $(\theta, \mu_0, \mu_1, \Sigma)$, $\theta = p(t = 1)$.
- GDA is similar to logistic regression (LR), parameter estimates are computed differently.
- When should we prefer GDA to LR, and vice versa?

Gaussian Discriminative Analysis vs Logistic Regression

- GDA is a generative model, LR is a discriminative model.
- GDA makes stronger modeling assumption: assumes class-conditional data is multivariate Gaussian.
- If this is true, GDA is asymptotically efficient (best model in limit of large N)
- But LR is more robust, less sensitive to incorrect modeling assumptions (what loss is it optimizing?)
- Many class-conditional distributions lead to logistic classifier.
- When these distributions are non-Gaussian (true almost always), LR usually beats GDA

A Generative View of Clustering

What if we do not observe the targets?

A Generative View of Clustering

- We covered hard and soft k-means algorithm for clustering.
- Today: statistical formulation of clustering $\rightarrow$ principled, justification for updates
- We need a sensible measure of what it means to cluster the data well
  - This makes it possible to judge different methods
  - It may help us decide on the number of clusters
- An obvious approach is to imagine that the data was produced by a generative model
  - Then we adjust the model parameters to maximize the probability that it would produce exactly the data we observed

Latent Variable Models

- To incorporate the idea of clusters, model a joint distribution:

$$p(\mathbf{x}, z) = p(\mathbf{x}|z)p(z)$$

  between the data and an unobserved cluster id $z \in \{1, \dots, K\}$.
- The "label" or cluster id z is not observed, so we call it a latent variable. Use z instead of t.
- Because z is unobserved, we cannot just maximize $\log p(\mathbf{x}, z)$. Instead, we must maximize just the likelihood of the data $\mathbf{x}$:

$$p(\mathbf{x}) = \sum_z p(\mathbf{x}, z) = \sum_z p(\mathbf{x}|z)p(z)$$

- This is an instance of a mixture model or more generally, a latent variable model.

Gaussian Mixture Model (GMM)

Most common mixture model: Gaussian mixture model (GMM)

- A GMM represents a **distribution** as

$$p(\mathbf{x}) = \sum_{k=1}^{K} \theta_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)$$

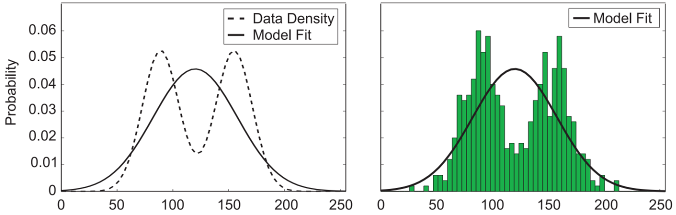
  with θ_k the mixing coefficients, where:

$$\sum_{k=1}^{K} \theta_k = 1 \quad \text{and} \quad \theta_k \geq 0 \quad \forall k$$

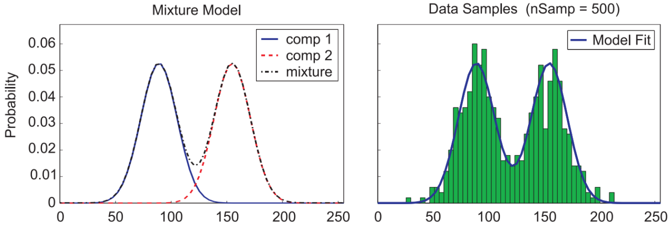
- In general mixture models are very powerful, but harder to optimize

Visualizing a Mixture of Gaussians – 1D Gaussians

- If you fit a Gaussian to data:

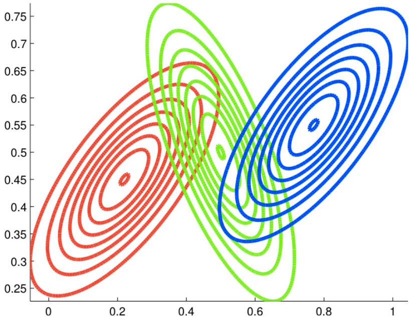


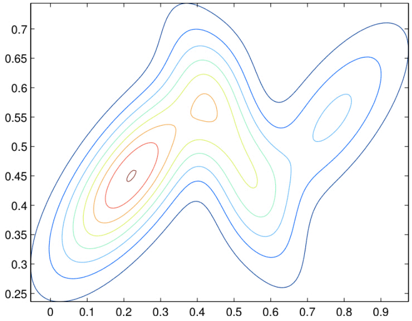
- Now, we are trying to fit a GMM (with $K = 2$ in this example):

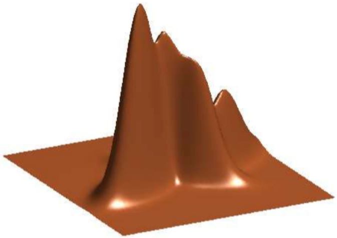


[Slide credit: K. Kutulakos]

Visualizing a Mixture of Gaussians – 2D Gaussians







Fitting GMMs: Maximum Likelihood

- Maximum likelihood maximizes

$$\ell(\phi) = \ln p(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)} | \phi) = \sum_{i=1}^{N} \ln \left(\sum_{k=1}^{K} \theta_k \mathcal{N}(\mathbf{x}^{(i)} | \boldsymbol{\mu}_k, \Sigma_k) \right)$$

  w.r.t $\phi = (\theta, \boldsymbol{\mu}_1, \Sigma_1, \ldots \boldsymbol{\mu}_K, \Sigma_K)$

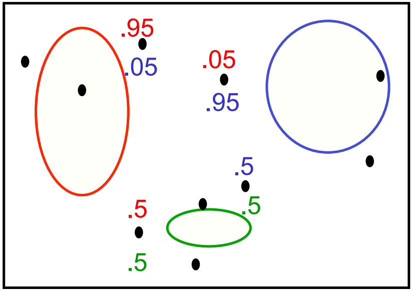
- Problems:
  - Singularities: Arbitrarily large likelihood when a Gaussian explains a single point
  - Identifiability: Solution is invariant to permutations
  - Non-convex
- How would you optimize this?
- Could try gradient descent, but don't forget to satisfy the constraints on θ_k and Σ_k.

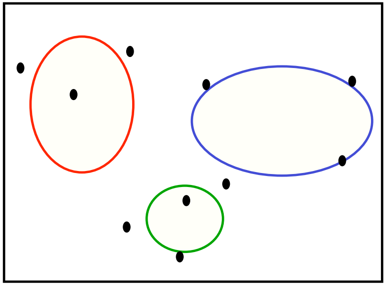
Expectation Maximization

- Typically a latent variable model is fit with the Expectation Maximization (EM) algorithm, or variants of it.
- The EM algorithm can be seen as a type of coordinate descent, just like K-means and our method for matrix completion.
- We won't go into details to justify the convergence of the algorithm, but I will show you the high-level algorithm for Gaussian mixture models and compare it to K-means.

Intuitively, How Can We Fit a Mixture of Gaussians?

1. E-step: Compute the posterior probability over z given our current model - i.e. how much do we think each Gaussian generates each datapoint.
2. M-step: Assuming that the data really was generated this way, change the parameters of each Gaussian to maximize the probability that it would generate the data it is currently responsible for.





Relation to k-Means

- The K-Means Algorithm:
  1. Assignment step: Assign each data point to the closest cluster
  2. Refitting step: Move each cluster center to the center of gravity of the data assigned to it
- The EM Algorithm:
  1. E-step: Compute the posterior probability over z given our current model
  2. M-step: Maximize the probability that it would generate the data it is currently responsible for.

EM Algorithm for GMM

- Initialize the means $\boldsymbol{\mu}_k$, covariances Σ_k and mixing coefficients θ_k
- Iterate until convergence:
  - E-step: Evaluate the responsibilities given current parameters

$$\gamma_k^{(i)} = p(z^{(i)}|\mathbf{x}) = \frac{\theta_k \mathcal{N}(\mathbf{x}^{(i)}|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^K \theta_j \mathcal{N}(\mathbf{x}^{(i)}|\boldsymbol{\mu}_j, \Sigma_j)}$$

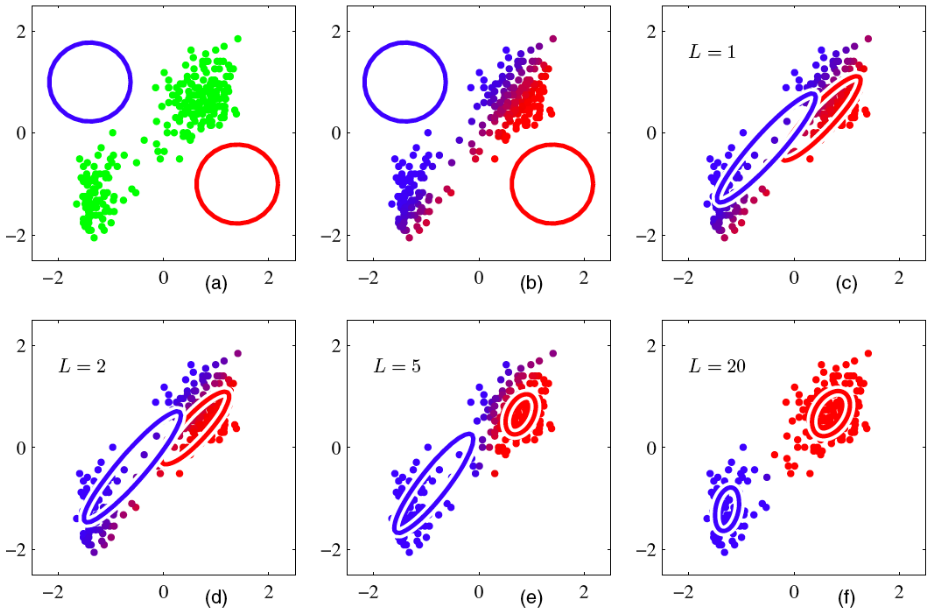
  - M-step: Re-estimate the parameters given current responsibilities

$$\begin{aligned}
\boldsymbol{\mu}_k &= \frac{1}{N_k} \sum_{i=1}^N \gamma_k^{(i)} \mathbf{x}^{(i)} \\
\Sigma_k &= \frac{1}{N_k} \sum_{i=1}^N \gamma_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^T \\
\theta_k &= \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{i=1}^N \gamma_k^{(i)}
\end{aligned}$$

  - Evaluate log likelihood and check for convergence.

EM Algorithm for GMM

- Can show that the EM algorithm monotonically improves the log-likelihood.



(a)
(b)
$L = 1$
(c)
$L = 2$
(d)
$L = 5$
(e)
$L = 20$
(f)

Mixture of Gaussians vs. K-means

- EM for mixtures of Gaussians is just like a soft version of K-means, with fixed priors and covariance
- Instead of hard assignments in the E-step, we do soft assignments based on the softmax of the squared Mahalanobis distance from each point to each cluster.
- Each center moved by weighted means of the data, with weights given by soft assignments
- In K-means, weights are 0 or 1.
- Confirm this at home!!!

Supplement: MLE for univariate Gaussian

$$
\begin{aligned}
0 = \frac{\partial \ell}{\partial \mu} &= -\frac{1}{\sigma^2} \sum_{i=1}^N \mathbf{x}^{(i)} - \mu \\
0 = \frac{\partial \ell}{\partial \sigma} &= \frac{\partial}{\partial \sigma}\left[\sum_{i=1}^N -\frac{1}{2}\log 2\pi - \log\sigma - \frac{1}{2\sigma^2}(\mathbf{x}^{(i)} - \mu)^2\right] \\
&= \sum_{i=1}^N -\frac{1}{2}\frac{\partial}{\partial \sigma}\log 2\pi - \frac{\partial}{\partial \sigma}\log\sigma - \frac{\partial}{\partial \sigma}\frac{1}{2\sigma}(\mathbf{x}^{(i)} - \mu)^2 \\
&= \sum_{i=1}^N 0 - \frac{1}{\sigma} + \frac{1}{\sigma^3}(\mathbf{x}^{(i)} - \mu)^2 \\
&= -\frac{N}{\sigma} + \frac{1}{\sigma^3}\sum_{i=1}^N (\mathbf{x}^{(i)} - \mu)^2
\end{aligned}
$$

$$
\begin{aligned}
\hat{\mu}_{\mathrm{ML}} &= \frac{1}{N}\sum_{i=1}^N \mathbf{x}^{(i)} \\
\hat{\sigma}_{\mathrm{ML}} &= \sqrt{\frac{1}{N}\sum_{i=1}^N (\mathbf{x}^{(i)} - \mu)^2}
\end{aligned}
$$