CPSC 540: Machine Learning

Mixture Models, Density Estimation, Factor Analysis

Mark Schmidt

University of British Columbia

Winter 2016
Assignment 2:
- 1 late day to hand it in now.

Assignment 3:
- Posted, due on February 23. Start early.
- Some additional hints will be added.
Multiple Kernel Learning

- Last time we discussed kernelizing L2-regularized linear models,
  \[
  \arg\min_{w \in \mathbb{R}^d} f(Xw, y) + \frac{\lambda}{2} \|w\|^2 \iff \arg\min_{z \in \mathbb{R}^n} f(Kz, y) + \frac{\lambda}{2} \|z\|^2_K,
  \]
  under fairly general conditions.
Multiple Kernel Learning

- Last time we discussed kernelizing L2-regularized linear models,

\[
\arg\min_{w \in \mathbb{R}^d} f(Xw, y) + \frac{\lambda}{2} \|w\|^2 \Leftrightarrow \arg\min_{z \in \mathbb{R}^n} f(Kz, y) + \frac{\lambda}{2} \|z\|^2_K,
\]

under fairly general conditions.

- What if we have multiple kernels and don’t know which to use?
  - Cross-validation.
Last time we discussed kernelizing L2-regularized linear models,
\[
\arg\min_{w \in \mathbb{R}^d} f(Xw, y) + \frac{\lambda}{2} \|w\|^2 \Leftrightarrow \arg\min_{z \in \mathbb{R}^n} f(Kz, y) + \frac{\lambda}{2} \|z\|^2_K,
\]
under fairly general conditions.

What if we have multiple kernels and don’t know which to use?
- Cross-validation.

What if we have multiple potentially-relevant kernels?
Last time we discussed kernelizing L2-regularized linear models,

$$\arg\min_{w \in \mathbb{R}^d} f(Xw, y) + \frac{\lambda}{2} \|w\|^2 \iff \arg\min_{z \in \mathbb{R}^n} f(Kz, y) + \frac{\lambda}{2} \|z\|^2_K,$$

under fairly general conditions.

What if we have multiple kernels and don’t know which to use?
- Cross-validation.

What if we have multiple potentially-relevant kernels?
- Multiple kernel learning:

$$\arg\min_{z_1 \in \mathbb{R}^n, z_2 \in \mathbb{R}^n, \ldots, z_k \in \mathbb{R}^n} f \left( \sum_{c=1}^{k} K_c z_c, y \right) + \frac{1}{2} \sum_{c=1}^{k} \lambda_c \|z_c\|_{K_c}.$$

- Defines a valid kernel and is convex if $f$ is convex.
Multiple Kernel Learning

- Last time we discussed kernelizing L2-regularized linear models,
  \[
  \arg\min_{w \in \mathbb{R}^d} f(Xw, y) + \frac{\lambda}{2} \|w\|^2 \iff \arg\min_{z \in \mathbb{R}^n} f(Kz, y) + \frac{\lambda}{2} \|z\|^2_K,
  \]
  under fairly general conditions.
- What if we have multiple kernels and don’t know which to use?
  - Cross-validation.
- What if we have multiple potentially-relevant kernels?
  - Multiple kernel learning:
    \[
    \arg\min_{z_1 \in \mathbb{R}^n, z_2 \in \mathbb{R}^n, \ldots, z_k \in \mathbb{R}^n} f \left( \sum_{c=1}^{k} K_c z_c, y \right) + \frac{1}{2} \sum_{c=1}^{k} \lambda_k \|z\|^2_{K_c}.
    \]
    - Defines a valid kernel and is convex if \( f \) is convex.
    - Group L1-regularization of parameters associated with each kernel.
      - Selects a sparse set of kernels.
    - Hierarchical kernel learning:
      - Use structured sparsity to search through exponential number of kernels.
Optimization Wrap-Up
Mixture Models
Gaussian Distributions
Learning with Hidden Values

Unconstrained and Smooth Optimization

- For typical unconstrained/smooth optimization of ML problems,

$$\arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(w^T x_i) + \frac{\lambda}{2} \|w\|^2.$$ 

- we discussed several methods:
  - **Gradient method:**
    - Linear convergence but $O(nd)$ iteration cost.
    - Faster versions like Nesterov/Newton exist.
Unconstrained and Smooth Optimization

For typical unconstrained/smooth optimization of ML problems,

$$\arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(w^T x_i) + \frac{\lambda}{2} \|w\|^2.$$  

we discussed several methods:

- **Gradient method:**
  - Linear convergence but $O(nd)$ iteration cost.
  - Faster versions like Nesterov/Newton exist.

- **Coordinate optimization:**
  - Faster than gradient method if iteration cost is $O(n)$.

- **Stochastic subgradient:**
  - Iteration cost is $O(d)$ but sublinear convergence rate.
  - SAG/SVRG improve to linear rate for finite datasets.
Constrained and Non-Smooth Optimization

For typical constrained/non-smooth optimization of ML problems, the “optimal” method for large $d$ is subgradient methods.
For typical constrained/non-smooth optimization of ML problems, the “optimal” method for large $d$ is subgradient methods. But we discussed better methods for specific cases:

- **Smoothing** which doesn’t work quite as well as we would like.
- **Projected-gradient** for “simple” constraints.
Constrained and Non-Smooth Optimization

- For typical constrained/non-smooth optimization of ML problems, the “optimal” method for large $d$ is subgradient methods.
- But we discussed better methods for specific cases:
  - **Smoothing** which doesn’t work quite as well as we would like.
  - **Projected-gradient** for “simple” constraints.
  - **Projected-Newton** for expensive $f_i$ and simple constraints.
  - **Proximal-gradient** if $g$ is “simple”.
  - **Proximal-Newton** for expensive $f_i$ and simple $g$. 
Constrained and Non-Smooth Optimization

- For typical constrained/non-smooth optimization of ML problems, the “optimal” method for large $d$ is subgradient methods.
- But we discussed better methods for specific cases:
  - **Smoothing** which doesn’t work quite as well as we would like.
  - **Projected-gradient** for “simple” constraints.
  - **Projected-Newton** for expensive $f_i$ and simple constraints.
  - **Proximal-gradient** if $g$ is “simple”.
  - **Proximal-Newton** for expensive $f_i$ and simple $g$.
  - **Coordinate optimization** if $g$ is separable.
  - **Stochastic subgradient** if $n$ is large.
  - **Dual optimization** for smoothing strongly-convex problems.
Constrained and Non-Smooth Optimization

- For typical constrained/non-smooth optimization of ML problems, the “optimal” method for large $d$ is subgradient methods.
- But we discussed better methods for specific cases:
  - **Smoothing** which doesn’t work quite as well as we would like.
  - **Projected-gradient** for “simple” constraints.
  - **Projected-Newton** for expensive $f_i$ and simple constraints.
  - **Proximal-gradient** if $g$ is “simple”.
  - **Proximal-Newton** for expensive $f_i$ and simple $g$.
  - **Coordinate optimization** if $g$ is separable.
  - **Stochastic subgradient** if $n$ is large.
  - **Dual optimization** for smoothing strongly-convex problems.
- With a few more tricks, you can almost always beat subgradient methods:
  - **Chambolle-Pock**: min-max problems.
  - **ADMM**: for “simple” regularized composed with affine function like $\|Ax\|_1$.
  - **Frank-Wolfe**: for nuclear-norm regularization.
  - **Mirror descent**: for probability-simplex constraints.
Even Bigger Problems?

- What about datasets that don’t fit on one machine?
  - We need to consider parallel and distributed optimization.
Even Bigger Problems?

- What about datasets that don’t fit on one machine?
  - We need to consider parallel and distributed optimization.
- Major issues:
  - Synchronization: we can’t wait for the slowest machine.
  - Communication: it’s expensive to transfer across machines.
Even Bigger Problems?

- What about datasets that don’t fit on one machine?
  - We need to consider parallel and distributed optimization.

- Major issues:
  - Synchronization: we can’t wait for the slowest machine.
  - Communication: it’s expensive to transfer across machines.

- “Embarassingly” parallel solution:
  - Split data across machines, each machine computes gradient of their subset.

- Fancier methods (key idea is usually that you just make step-size smaller):
  - Asynchronous stochastic gradient.
  - Parallel coordinate optimization.
  - Decentralized gradient.
Last Time: Density Estimation

- Last time we started discussing density estimation.
  - Unsupervised task of estimating $p(x)$.
- It can also be used for supervised learning:
  - Generative models estimate joint distribution over feature and labels,
    $$ p(y^i|x^i) \propto p(x^i, y^i) = p(x^i|y^i)p(y^i). $$
Last Time: Density Estimation

- Last time we started discussing density estimation.
  - Unsupervised task of estimating $p(x)$.
- It can also be used for supervised learning:
  - Generative models estimate joint distribution over feature and labels,
    \[
    p(y^i|x^i) \propto p(x^i, y^i) = p(x^i|y^i)p(y^i).
    \]
  - Estimating $p(x^i, y^i)$ is density estimation problem.
  - Estimating $p(y^i)$ and $p(x^i|y^i)$ are also density estimation problems.
Last Time: Density Estimation

- Last time we started discussing **density estimation**.
  - Unsupervised task of estimating $p(x)$.
- It can also be used for supervised learning:
  - **Generative models** estimate joint distribution over feature and labels,
    $$p(y^i|x^i) \propto p(x^i, y^i) = p(x^i|y^i)p(y^i).$$
- Estimating $p(x^i, y^i)$ is density estimation problem.
- Estimating $p(y^i)$ and $p(x^i|y^i)$ are also density estimation problems.
- Special cases:
  - Naive Bayes models $p(x^i|y^i)$ as product of independent distributions.
  - Linear discriminant analysis models $p(x^i|y^i)$ as a multivariate Gaussian.
- Currently unpopular, but may be coming back:
  - We believe that most human learning is unsupervised.
Last Time: Independent vs. General Discrete Distributions

- We considered density estimation with discrete variables,

\[ X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \]

and considered two extreme approaches:

- **Product of independent distributions:**

\[ p(x) = \prod_{j=1}^{d} p(x_j). \]

  Easy to fit but strong **independence assumption:**

  - Knowing \( x_j \) tells you nothing about \( x_k \).

- **General discrete distribution:**

\[ p(x) = \theta_x. \]

No assumptions but hard to fit: Parameter vector \( \theta_x \) for each possible \( x \).
We considered density estimation with discrete variables,

\[ X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \]

and considered two extreme approaches:

- **Product of independent distributions**:
  \[ p(x) = \prod_{j=1}^{d} p(x_j). \]
  
  Easy to fit but strong independence assumption:
  - Knowing \( x_j \) tells you nothing about \( x_k \).

- **General discrete distribution**:
  \[ p(x) = \theta_x. \]
  
  No assumptions but hard to fit:
  - Parameter vector \( \theta_x \) for each possible \( x \).

What lies between these extremes?
Consider a coin flipping scenario where we have two coins:
- Coin 1 has $\theta_1 = 0.5$ (fair) and coin 2 has $\theta_2 = 1$ (fixed).

This is called a mixture model: The probability is a convex combination ("mixture") of probabilities. Here we get a Bernoulli with $\theta = 0.75$, but other mixtures are more interesting...
Mixture of Bernoullis

- Consider a coin flipping scenario where we have two coins:
  - Coin 1 has $\theta_1 = 0.5$ (fair) and coin 2 has $\theta_2 = 1$ (fixed).
  - With 0.5 probability we look coin 1, otherwise we look at coin 2:
    
    $$p(x = 1|\theta_1, \theta_2) = p(z = 1)p(x = 1|\theta_1) + p(z = 2)p(x = 1|\theta_2)$$
    $$= 0.5\theta_1 + 0.5\theta_2,$$

  where $z$ is the choice of coin we flip.
Consider a coin flipping scenario where we have two coins:

- Coin 1 has $\theta_1 = 0.5$ (fair) and coin 2 has $\theta_2 = 1$ (fixed).

With 0.5 probability we look coin 1, otherwise we look at coin 2:

$$p(x = 1|\theta_1, \theta_2) = p(z = 1)p(x = 1|\theta_1) + p(z = 2)p(x = 1|\theta_2)$$

$$= 0.5\theta_1 + 0.5\theta_2,$$

where $z$ is the choice of coin we flip.

This is called a mixture model:

- The probability is a convex combination ("mixture") of probabilities.

Here we get a Bernoulli with $\theta = 0.75$, but other mixtures are more interesting...
Consider a mixture of the product of independent Bernoullis:

\[ p(x) = 0.5 \prod_{j=1}^{d} p(x_j | \theta_{1j}) + 0.5 \prod_{j=1}^{d} p(x_j | \theta_{2j}). \]

E.g., \( \theta_1 = [\theta_{11} \ \theta_{12} \ \theta_{13}] = [0 \ 0.7 \ 1] \) and \( \theta_2 = [1 \ 0.7 \ 0.8] \).

Conceptually, we now have two sets of coins:
- With probability 0.5 we throw the first set, otherwise we throw the second set.
Consider a mixture of the product of independent Bernoullis:

\[ p(x) = 0.5 \prod_{j=1}^{d} p(x_j|\theta_{1j}) + 0.5 \prod_{j=1}^{d} p(x_j|\theta_{2j}). \]

E.g., \( \theta_1 = [\theta_{11} \quad \theta_{12} \quad \theta_{13}] = [0 \quad 0.7 \quad 1] \) and \( \theta_2 = [1 \quad 0.7 \quad 0.8] \).

Conceptually, we now have two sets of coins:
- With probability 0.5 we throw the first set, otherwise we throw the second set.
- Product of independent distributions is special case where \( \theta_{1j} = \theta_{2j} \) for all \( j \):
  - We haven’t lost anything by taking a mixture.
Mixture of Independent Bernoullis

• Consider a mixture of the product of independent Bernoullis:

\[ p(x) = 0.5 \prod_{j=1}^{d} p(x_j | \theta_{1j}) + 0.5 \prod_{j=1}^{d} p(x_j | \theta_{2j}). \]

• E.g., \( \theta_1 = [\theta_{11} \ \theta_{12} \ \theta_{13}] = [0 \ 0.7 \ 1] \) and \( \theta_2 = [1 \ 0.7 \ 0.8] \).

• Conceptually, we now have two sets of coins:
  • With probability 0.5 we throw the first set, otherwise we throw the second set.
  • Product of independent distributions is special case where \( \theta_{1j} = \theta_{2j} \) for all \( j \):
    • We haven’t lost anything by taking a mixture.
  • But mixtures can model dependencies between variables \( x_j \):
    • If you know \( x_j \), it tells you something about which mixture \( x_k \) comes from.
    • E.g., if \( \theta_1 = [0 \ 0 \ 0] \) and \( \theta_2 = [1 \ 1 \ 1] \), seeing \( x_j = 1 \) tells you \( x_k = 1 \).
Mixture of Independent Bernoullis

- General mixture of independent Bernoullis:

\[
p(x) = \sum_{c=1}^{k} p(z = c)p(x | z = c),
\]

where every thing is conditioned on \( \theta_c \) values and

1. We have likelihood \( p(x | z = c) \) of \( x \) if it came from cluster \( c \).
2. Mixture weight \( p(z = c) \) is probability that \( c \) generated data.
Mixture of Independent Bernoullis

- General mixture of independent Bernoullis:

\[ p(x) = \sum_{c=1}^{k} p(z = c)p(x|z = c), \]

where every thing is conditioned on \( \theta_c \) values and

1. We have likelihood \( p(x|z = c) \) of \( x \) if it came from cluster \( c \).
2. Mixture weight \( p(z = c) \) is probability that \( c \) generated data.

- We typically model \( p(z = c) \) using a categorical distribution.
- With \( k \) large enough, we can model any discrete distribution.
  - Though \( k \) may not be much smaller than \( 2^d \) in the worst case.
Mixture of Independent Bernoullis

- General mixture of independent Bernoullis:

\[ p(x) = \sum_{c=1}^{k} p(z = c)p(x|z = c), \]

where every thing is conditioned on \( \theta_c \) values and

1. We have likelihood \( p(x|z = c) \) of \( x \) if it came from cluster \( c \).
2. Mixture weight \( p(z = c) \) is probability that \( c \) generated data.

- We typically model \( p(z = c) \) using a categorical distribution.
- With \( k \) large enough, we can model any discrete distribution.
  - Though \( k \) may not be much smaller than \( 2^d \) in the worst case.

- An important quantity is the responsibility,

\[ p(z = c|x) = \frac{p(x|z = c)p(z = c)}{\sum_{c'} p(x|z = c')p(z' = c')}, \]

the probability that \( x \) came from mixture \( c \).

- The responsibilities are often interpreted as a probabilistic clustering.
Mixture of Independent Bernoullis

Plotting mean vectors $\theta_c$ with 10 mixtures trained on MNIST:
(hand-written images of the numbers 0 through 9)
(pause)
Univariate Gaussian

Consider the case of a continuous variable $x \in \mathbb{R}$:

$$X = \begin{bmatrix} 0.53 \\ 1.83 \\ -2.26 \\ 0.86 \end{bmatrix}.$$
Univariate Gaussian

- Consider the case of a continuous variable \( x \in \mathbb{R} \):

\[
X = \begin{bmatrix}
0.53 \\
1.83 \\
-2.26 \\
0.86
\end{bmatrix}.
\]

- Even with 1 variable there are many possible distributions.
- Most common is the Gaussian (or "normal") distribution:

\[
p(x|\mu, \sigma^2) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left( -\frac{(x - \mu)^2}{2\sigma^2} \right) \text{ or } x \sim \mathcal{N}(\mu, \sigma^2).
\]
Univariate Gaussian

Why the Gaussian distribution?

- Central limit theorem: mean estimate converges to Gaussian.
- Data might actually follow Gaussian.
Why the Gaussian distribution?
- Central limit theorem: mean estimate converges to Gaussian.
- Data might actually follow Gaussian.
- Analytics properties: symmetry, closed-form solution for $\mu$ and $\sigma$:
  - Maximum likelihood for mean is $\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x^i$.
  - Maximum likelihood for variance is $\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x^i - \hat{\mu})^2$ (for $n > 1$).

https://en.wikipedia.org/wiki/Gaussian_function
Alternatives to Univariate Gaussian

- Why not the Gaussian distribution?
  - Negative log-likelihood is a quadratic function of $\mu$,

  \[-\log p(X|\mu, \sigma^2) = \sum_{i=1}^{n} p(x^i|\mu, \sigma^2) = \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x^i - \mu)^2 - \log(\sigma) + \text{const.} \]

  so as with least squares distribution is not robust to outliers.
Alternatives to Univariate Gaussian

- Why not the Gaussian distribution?
  - Negative log-likelihood is a quadratic function of $\mu$,

$$-\log p(X|\mu, \sigma^2) = \sum_{i=1}^{n} p(x^i|\mu, \sigma^2) = \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x^i - \mu)^2 - \log(\sigma) + \text{const.}$$

  so as with least squares distribution is not robust to outliers.
- More robust: Laplace distribution or student’s t-distribution
Alternatives to Univariate Gaussian

- Why not the Gaussian distribution?
  - Negative log-likelihood is a quadratic function of $\mu$,
    $$-\log p(X|\mu, \sigma^2) = \sum_{i=1}^{n} p(x^i|\mu, \sigma^2) = \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x^i - \mu)^2 - \log(\sigma) + \text{const.}$$
    
    so as with least squares distribution is not robust to outliers.
  - More robust: Laplace distribution or student’s t-distribution
  - Gaussian distribution is unimodal.
Alternatives to Univariate Gaussian

- Why not the Gaussian distribution?
  - Negative log-likelihood is a quadratic function of \( \mu \),
    
    \[
    -\log p(X|\mu, \sigma^2) = \sum_{i=1}^{n} p(x^i|\mu, \sigma^2) = \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x^i - \mu)^2 - \log(\sigma) + \text{const.}
    \]
    
    so as with least squares distribution is not robust to outliers.
  - More robust: Laplace distribution or student’s t-distribution
  - Gaussian distribution is unimodal.
  - Even with one variable we may want to do a mixture of Gaussians.
Multivariate Gaussian Distribution

The generalization to multiple variables is the multivariate normal/Gaussian,

\[ p(x | \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right), \quad \text{or} \quad x \sim \mathcal{N}(\mu, \Sigma), \]

where \( \mu \in \mathbb{R}^d \), \( \Sigma \in \mathbb{R}^{d \times d} \) and \( \Sigma \succ 0 \), and \( |\Sigma| \) is the determinant.

http://personal.kenyon.edu/hartlaub/MellonProject/Bivariate2.html
The generalization to multiple variables is the **multivariate normal/Gaussian**,  

\[
p(x | \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right), \quad \text{or } x \sim \mathcal{N}(\mu, \Sigma),
\]

where $\mu \in \mathbb{R}^d$, $\Sigma \in \mathbb{R}^{d \times d}$ and $\Sigma \succ 0$, and $|\Sigma|$ is the determinant.

**Why the multivariate Gaussian?**
- Inherits the good/bad properties of univariate Gaussian.
  - Closed-form MLE but unimodal and not robust to outliers.
**Multivariate Gaussian Distribution**

- The generalization to multiple variables is the *multivariate normal/Gaussian*,

\[
p(x|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right), \quad \text{or } x \sim \mathcal{N}(\mu, \Sigma),
\]

where \( \mu \in \mathbb{R}^d, \Sigma \in \mathbb{R}^{d \times d} \) and \( \Sigma \succ 0 \), and \( |\Sigma| \) is the determinant.

- **Why the multivariate Gaussian?**
  - Inherits the good/bad properties of univariate Gaussian.
    - Closed-form MLE but unimodal and not robust to outliers.
  - **Closed** under some common operations:
    - Products of Gaussians PDFs is Gaussian:
      \[
p(x_1|\mu_1, \Sigma_1)p(x_2|\mu_2, \Sigma_2) = p(\tilde{x}|\tilde{\mu}, \tilde{\Sigma}).
\]
    - Marginal distributions \( p(x_S|\mu, \Sigma) \) are Gaussians.
    - Conditional distributions \( p(x_S|x_{-S}, \mu, \Sigma) \) are Gaussians.
Product of Independent Gaussians

Consider a distribution that is a product of independent Gaussians,

\[ x_j \sim \mathcal{N}(\mu_j, \sigma^2_f), \]

then the joint distribution is a multivariate Gaussian,

\[ x_j \sim \mathcal{N}(\mu, \Sigma), \]

with \( \mu = (\mu_1, \mu_2, \ldots, \mu_d) \) and \( \Sigma \) diagonal with elements \( \sigma_j \).
Product of Independent Gaussians

Consider a distribution that is a product of independent Gaussians,

\[ x_j \sim \mathcal{N}(\mu_j, \sigma_j^2), \]

then the joint distribution is a multivariate Gaussian,

\[ x_j \sim \mathcal{N}(\mu, \Sigma), \]

with \( \mu = (\mu_1, \mu_2, \ldots, \mu_d) \) and \( \Sigma \) diagonal with elements \( \sigma_j \).

This follows from

\[
p(x|\mu, \Sigma) = p(x_j|\mu_j, \sigma_j^2) \propto \prod_{j=1}^{d} \exp \left( -\frac{(x_j - \mu_j)^2}{2\sigma_j^2} \right)
\]

\[
= \exp \left( -\frac{1}{2} \sum_{j=1}^{d} \frac{(x_j - \mu_j)^2}{\sigma_j^2} \right) \quad (e^a e^b = e^{a+b})
\]

\[
= \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right) \quad \text{(definition of } \Sigma).\]
What is the effect of diagonal $\Sigma$ in the independent Gaussian model?

- If $\Sigma = \alpha I$ the level curves are circles (1 parameter).
- If $\Sigma = D$ (diagonal) they axis-aligned ellipses ($d$ parameters).
- If $\Sigma$ is dense they do not need to be axis-aligned ($d(d + 1)/2$ parameters).
  (by symmetry, we need to estimate upper-triangular part of $\Sigma$)
Maximum Likelihood Estimation in Multivariate Gaussians

With a multivariate Gaussian we have

\[ p(x|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right), \]

so up to a constant our negative log-likelihood is

\[ \frac{1}{2} \sum_{i=1}^{n} (x^i - \mu)^T \Sigma^{-1} (x^i - \mu) + \frac{n}{2} \log |\Sigma|. \]
Maximum Likelihood Estimation in Multivariate Gaussians

- With a multivariate Gaussian we have

\[ p(x|\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right), \]

so up to a constant our negative log-likelihood is

\[ \frac{1}{2} \sum_{i=1}^{n} (x^i - \mu)^T \Sigma^{-1} (x^i - \mu) + \frac{n}{2} \log |\Sigma|. \]

- This is quadratic in \( \mu \), taking the gradient with respect to \( \mu \) and setting to zero:

\[ 0 = \sum_{i=1}^{n} \Sigma^{-1} (x^i - \mu), \text{ or that } \Sigma^{-1} \sum_{i=1}^{n} \mu = \Sigma^{-1} \sum_{i=1}^{n} x^i. \]
Maximum Likelihood Estimation in Multivariate Gaussians

- With a multivariate Gaussian we have
  \[ p(x|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right) , \]

  so up to a constant our negative log-likelihood is
  \[ \frac{1}{2} \sum_{i=1}^{n} (x^i - \mu)^T \Sigma^{-1} (x^i - \mu) + \frac{n}{2} \log |\Sigma| . \]

- This is quadratic in \( \mu \), taking the gradient with respect to \( \mu \) and setting to zero:
  \[ 0 = \sum_{i=1}^{n} \Sigma^{-1} (x^i - \mu) , \text{ or that } \Sigma^{-1} \sum_{i=1}^{n} \mu = \Sigma^{-1} \sum_{i=1}^{n} x^i . \]

- Noting that \( \sum_{i=1}^{n} \mu = n \mu \) and pre-multiplying by \( \Sigma \) we get \( \mu = \frac{1}{n} \sum_{i=1}^{n} x^i . \)
  - So \( \mu \) should be the average, and it doesn’t depend on \( \Sigma \).
Maximum Likelihood Estimation in Multivariate Gaussians

Re-parameterizing in terms of precision matrix $\Theta = \Sigma^{-1}$ we have

$$\frac{1}{2} \sum_{i=1}^{n} (x^i - \mu)^T \Sigma^{-1} (x^i - \mu) + \frac{n}{2} \log |\Sigma|$$

$$= \frac{1}{2} \sum_{i=1}^{n} \text{Tr} \left( (x^i - \mu)^T \Theta (x^i - \mu) \right) + \frac{n}{2} \log |\Theta^{-1}| \quad (y^T Ay = \text{Tr}(y^T Ay))$$

$$= \frac{1}{2} \sum_{i=1}^{n} \text{Tr} \left( (x^i - \mu)(x^i - \mu)^T \Theta \right) - \frac{n}{2} \log |\Theta| \quad (\text{Tr}(AB) = \text{Tr}(BA))$$
Maximum Likelihood Estimation in Multivariate Gaussians

- Re-parameterizing in terms of precision matrix $\Theta = \Sigma^{-1}$ we have

$$
\frac{1}{2} \sum_{i=1}^{n} (x^i - \mu)^T \Sigma^{-1} (x^i - \mu) + \frac{n}{2} \log |\Sigma|
$$

$$
= \frac{1}{2} \sum_{i=1}^{n} \text{Tr} ((x^i - \mu)^T \Theta (x^i - \mu)) + \frac{n}{2} \log |\Theta^{-1}| \quad (y^T A y = \text{Tr}(y^T A y))
$$

$$
= \frac{1}{2} \sum_{i=1}^{n} \text{Tr}((x^i - \mu)(x^i - \mu)^T \Theta) - \frac{n}{2} \log |\Theta| \quad (\text{Tr}(AB) = \text{Tr}(BA))
$$

- Changing trace/sum and using sample covariance $S = \frac{1}{n} \sum_{i=1}^{n} (x^i - \mu)(x^i - \mu)^T$, we have

$$
= \frac{1}{2} \text{Tr} \left( \sum_{i=1}^{n} (x^i - \mu)(x^i - \mu)^T \Theta \right) - \frac{n}{2} \log |\Theta| \quad (\sum_i \text{Tr}(A_i B) = \text{Tr}(\sum_i A_i B))
$$

$$
= \frac{n}{2} \text{Tr}(S \Theta) - \frac{n}{2} \log |\Theta|.
$$
Maximum Likelihood Estimation in Multivariate Gaussians

- So the NLL in terms of the precision matrix $\Theta$ is

$$\frac{n}{2}\text{Tr}(S\Theta) - \frac{n}{2}\log|\Theta|,$$

with

$$S = \frac{1}{n}\sum_{i=1}^{n}(x^i - \mu)(x^i - \mu)^T.$$
Maximum Likelihood Estimation in Multivariate Gaussians

- So the NLL in terms of the precision matrix $\Theta$ is
  \[
  \frac{n}{2} \text{Tr}(S\Theta) - \frac{n}{2} \log |\Theta|, \quad \text{with} \quad S = \frac{1}{n} \sum_{i=1}^{n} (x^i - \mu)(x^i - \mu)^T
  \]

- Weird-looking but has nice properties:
  - $\text{Tr}(S\Theta)$ is linear function of $\Theta$, with $\nabla_{\Theta} \text{Tr}(S\Theta) = S$.
  - Negative log-determinant is strictly-convex and has $\nabla_{\Theta} \log |\Theta| = \Theta^{-1}$.
    (generalization of $\nabla \log |x| = 1/x$ for $x > 0$.)
Maximum Likelihood Estimation in Multivariate Gaussians

- So the NLL in terms of the precision matrix $\Theta$ is

$$\frac{n}{2} \text{Tr}(S\Theta) - \frac{n}{2} \log |\Theta|, \text{ with } S = \frac{1}{n} \sum_{i=1}^{n} (x^i - \mu)(x^i - \mu)^T$$

- Weird-looking but has nice properties:
  - $\text{Tr}(S\Theta)$ is linear function of $\Theta$, with $\nabla_{\Theta} \text{Tr}(S\Theta) = S$.
  - Negative log-determinant is strictly-convex and has $\nabla_{\Theta} \log |\Theta| = \Theta^{-1}$. (generalization of $\nabla \log |x| = 1/x$ for $x > 0$.)

- Using the MLE $\hat{\mu}$ and setting the gradient matrix to zero we get

$$0 = nS - n\Theta^{-1}, \text{ or } \Theta = S^{-1}, \text{ or } \Sigma = S = \frac{1}{n} \sum_{i=1}^{n} (x^i - \hat{\mu})(x^i - \hat{\mu})^T.$$
Maximum Likelihood Estimation in Multivariate Gaussians

- So the NLL in terms of the precision matrix $\Theta$ is

\[
\frac{n}{2} \text{Tr}(S\Theta) - \frac{n}{2} \log |\Theta|, \text{ with } S = \frac{1}{n} \sum_{i=1}^{n} (x^i - \mu)(x^i - \mu)^T
\]

- Weird-looking but has nice properties:
  - $\text{Tr}(S\Theta)$ is linear function of $\Theta$, with $\nabla_{\Theta} \text{Tr}(S\Theta) = S$.
  - Negative log-determinant is strictly-convex and has $\nabla_{\Theta} \log |\Theta| = \Theta^{-1}$.
    (generalization of $\nabla \log |x| = 1/x$ for $x > 0$.

- Using the MLE $\hat{\mu}$ and setting the gradient matrix to zero we get

\[
0 = nS - n\Theta^{-1}, \text{ or } \Theta = S^{-1}, \text{ or } \Sigma = S = \frac{1}{n} \sum_{i=1}^{n} (x^i - \hat{\mu})(x^i - \hat{\mu})^T.
\]

- The constraint $\Sigma \succ 0$ means we need empirical covariance $S \succ 0$.
  - If $S$ is not invertible, NLL is unbounded below and no MLE exists.
If we define centered vectors $\tilde{x}^i = x^i - \mu$ then empirical covariance is

$$S = \frac{1}{n} \sum_{i=1}^{n} (x^i - \mu)(x^i - \mu)^T = \sum_{i=1}^{n} \tilde{x}^i (\tilde{x}^i)^T = \tilde{X}^T \tilde{X} \succeq 0,$$

so $S$ is positive semi-definite but not positive-definite by construction.

- If data has noise, it will be positive-definite with $n$ large enough.
- For $\Theta \succ 0$, note that for an upper-triangular $T$ we have

$$\log |T| = \log(\prod(eig(T))) = \log(\prod(diag(T))) = \text{Tr}(\log(diagT)),$$

where we’ve used Matlab notation.

- So to compute $\log |\Theta|$ for $\Theta \succ 0$, use Cholesky to turn into upper-triangular.
  - Bonus: Cholesky will fail if $\Theta \succ 0$ is not true, so it checks constraint.
MAP Estimation in Multivariate Gaussian

- We typically don’t regularize $\mu$, but you could add an L2-regularizer $\frac{\lambda}{2} \|\mu\|^2$.
- A classic “hack” for $\Sigma$ is to add a diagonal matrix to $S$ and use

$$
\Sigma = S + \lambda I,
$$

which satisfies $\Sigma \succ 0$ by construction.
MAP Estimation in Multivariate Gaussian

- We typically don’t regularize $\mu$, but you could add an L2-regularizer $\frac{\lambda}{2}\|\mu\|^2$.
- A classic “hack” for $\Sigma$ is to add a diagonal matrix to $S$ and use

$$\Sigma = S + \lambda I,$$

which satisfies $\Sigma \succ 0$ by construction.
- This corresponds to a regularizer that penalizes diagonal of the precision,

$$\text{Tr}(S\Theta) - \log |\Theta| + \lambda \text{Tr}(\Theta).$$

Gives sparse $\Theta$ and introduces independences. E.g., if it makes $\Theta$ diagonal then all variables are independent.

Can solve very large instances with proximal-Newton and other tricks.
MAP Estimation in Multivariate Gaussian

- We typically don’t regularize $\mu$, but you could add an L2-regularizer $\frac{\lambda}{2} \| \mu \|^2$.
- A classic “hack” for $\Sigma$ is to add a diagonal matrix to $S$ and use
  \[ \Sigma = S + \lambda I, \]
  which satisfies $\Sigma \succ 0$ by construction.
  - This corresponds to a regularizer that penalizes diagonal of the precision,
    \[ \text{Tr}(S\Theta) - \log |\Theta| + \lambda \text{Tr}(\Theta). \]
- Recent substantial interest in generalization called the graphical LASSO,
  \[ f(\Theta) = \text{Tr}(S\Theta) - \log |\Theta| + \lambda \| \Theta \|_1 = \text{Tr}(S\Theta + \lambda \Theta) - \log |\Theta|. \]
  where we are using the element-wise L1-norm.
  - Gives sparse $\Theta$. 

Gives sparse $\Theta$.
MAP Estimation in Multivariate Gaussian

- We typically don’t regularize $\mu$, but you could add an L2-regularizer $\frac{\lambda}{2}||\mu||^2$.
- A classic “hack” for $\Sigma$ is to add a diagonal matrix to $S$ and use
  \[
  \Sigma = S + \lambda I,
  \]
  which satisfies $\Sigma \succ 0$ by construction.
  - This corresponds to a regularizer that penalizes diagonal of the precision,
    \[
    \text{Tr}(S\Theta) - \log |\Theta| + \lambda \text{Tr}(\Theta).
    \]
- Recent substantial interest in generalization called the graphical LASSO,
  \[
  f(\Theta) = \text{Tr}(S\Theta) - \log |\Theta| + \lambda ||\Theta||_1 = \text{Tr}(S\Theta + \lambda \Theta) - \log |\Theta|.
  \]
  where we are using the element-wise L1-norm.
  - Gives sparse $\Theta$ and introduces independences.
    - E.g., if it makes $\Theta$ diagonal then all variables are independent.
    - Can solve very large instances with proximal-Newton and other tricks.
Alternatives to Multivariate Gaussian

- Why not the multivariate Gaussian distribution?
  - Still not robust, may want to consider multivariate Laplace or multivariate T.
Alternatives to Multivariate Gaussian

- Why not the multivariate Gaussian distribution?
  - Still not robust, may want to consider multivariate Laplace of multivariate T.
  - Still unimodal, may want to consider mixture of Gaussians.
(pause)
Learning with Hidden Values

- We often want to learn when some variables unobserved/missing/hidden/latent.
- For example, we could have a dataset

\[
X = \begin{bmatrix}
N & 33 & 5 \\
F & 10 & 1 \\
F & ? & 2 \\
M & 22 & 0
\end{bmatrix},
\quad y = \begin{bmatrix}
-1 \\
+1 \\
-1 \\
?
\end{bmatrix}.
\]

- Missing values are very common in real datasets.
Learning with Hidden Values

- We often want to learn when some variables unobserved/missing/hidden/latent.
- For example, we could have a dataset

\[ X = \begin{bmatrix} N & 33 & 5 \\ F & 10 & 1 \\ F & ? & 2 \\ M & 22 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} -1 \\ +1 \\ -1 \\ ? \end{bmatrix}. \]

- Missing values are very common in real datasets.
- Heuristic approach:
  1. Imputation: replace each ? with the most likely value.
  2. Estimation: fit model with these imputed values.
- Sometimes you alternate between these two steps ("hard EM").
- EM algorithm is a more theoretically-justified version of this.
We’ll focus on data that is missing at random (MAR):

The assumption that \( ? \) is missing does not depend on the missing value.
Missing at Random (MAR)

We’ll focus on data that is **missing at random (MAR):**

- The assumption that ? is missing does **not** depend on the missing value.
- Note that this definition doesn’t agree with intuitive notion of random:
  - variable that is *always* missing would be “missing at random”.
  - The intuitive/stronger version is **missing completely at random (MCAR).**
Missing at Random (MAR)

We’ll focus on data that is missing at random (MAR):

- The assumption that \( ? \) is missing does not depend on the missing value.
- Note that this definition doesn’t agree with intuitive notion of random:
  - variable that is always missing would be “missing at random”.
  - The intuitive/stronger version is missing completely at random (MCAR).

Examples of MCAR and MAR for digit classification:

- Missing random pixels/labels: MCAR.
- Hide the top half of every digit: MAR.
- Hide the labels of all the “2” examples: not MAR.

If you are not MAR, you need to model why data is missing.
Missing at Random (MAR)

- We’ll focus on data that is **missing at random (MAR):**
  - The assumption that ? is missing does **not depend on the missing value.**
  - Note that this definition doesn’t agree with intuitive notion of random:
    - variable that is *always* missing would be “missing at random”.
    - The intuitive/stronger version is **missing completely at random (MCAR).**

- Examples of MCAR and MAR for digit classification:
  - Missing random pixels/labels: MCAR.
Missing at Random (MAR)

- We’ll focus on data that is missing at random (MAR):
  - The assumption that \( ? \) is missing does not depend on the missing value.
  - Note that this definition doesn’t agree with intuitive notion of random:
    - variable that is always missing would be “missing at random”.
    - The intuitive/stronger version is missing completely at random (MCAR).

- Examples of MCAR and MAR for digit classification:
  - Missing random pixels/labels: MCAR.
  - Hide the the top half of every digit: MAR.
We’ll focus on data that is missing at random (MAR):

- The assumption that \( ? \) is missing does not depend on the missing value.
- Note that this definition doesn’t agree with intuitive notion of random:
  - variable that is always missing would be “missing at random”.
  - The intuitive/stronger version is missing completely at random (MCAR).

Examples of MCAR and MAR for digit classification:

- Missing random pixels/labels: MCAR.
- Hide the the top half of every digit: MAR.
- Hide the labels of all the “2” examples: not MAR.

If you are not MAR, you need to model why data is missing.
Summary

- **Generative models** use density estimation for supervised learning.
Summary

- **Generative models** use density estimation for supervised learning.
- **Mixture models** write probability as convex combination of probabilities.
  - Model dependencies between variables even if components are independent.
  - Perform a soft-clustering of examples.
- Multivariate Gaussian generalizes univariate Gaussian for multiple variables.
- Closed-form solution but unimodal and not robust.
- Missing at random: fact that variable is missing does not depend on its value.
- Next time: EM algorithm for hidden variables and probabilistic PCA.
Summary

- **Generative models** use density estimation for supervised learning.
- **Mixture models** write probability as convex combination of probabilities.
  - Model dependencies between variables even if components are independent.
  - Perform a soft-clustering of examples.
- **Multivariate Gaussian** generalizes univariate Gaussian for multiple variables.
  - Closed-form solution but unimodal and not robust.
Summary

- **Generative models** use density estimation for supervised learning.
- **Mixture models** write probability as convex combination of probabilities.
  - Model dependencies between variables even if components are independent.
  - Perform a soft-clustering of examples.
- **Multivariate Gaussian** generalizes univariate Gaussian for multiple variables.
  - Closed-form solution but unimodal and not robust.
- **Missing at random**: fact that variable is missing does not depend on its value.
- Next time: EM algorithm for hidden variables and probabilistic PCA.