8. Using the mixture model for collaborative filtering

Extension Note: Project 4 due date has been extended by 1 more day to **August 22 23:59UTC** .

Reporting log likelihood values on Netflix data

1.0/1.0 point (graded)

Now, run the EM algorithm on the incomplete data matrix from Netflix ratings `netflix_incomplete.txt` . As before, please use seeds from $[0, 1, 2, 3, 4]$ and report the best log likelihood you achieve with $K = 1$ and $K = 12$.

This may take on the order of a couple minutes for $K = 12$.

Report the maximum likelihood for each K using seeds $0, 1, 2, 3, 4$:

$\text{Log-likelihood}|_{K=1} =$ -1521060.9539852478 ✓ **Answer:** -1521060.9539

$\text{Log-likelihood}|_{K=12} =$ -1390234.4223469393 ✓ **Answer:** -1390234.4223

Submit You have used 3 of 10 attempts

Answers are displayed within the problem

Completing missing entries

1.0/1.0 point (graded)

Now that we have a mixture model, how do we use it to complete a partially observed rating matrix? Derive an expression for completing a particular row, say x_C where the observed values are $i \in C$.

In `em.py` implement the function `fill_matrix.py` that takes as input an incomplete data matrix `X` as well as a mixture model, and outputs a completed version of the matrix `X_pred` .

Available Functions: You have access to the NumPy python library as `np` , to the `GaussianMixture` class and to typing annotation `typing.Tuple` as `Tuple` . You also have access to `scipy.special.logsumexp` as `logsumexp`

```
    return post, log_likelihood

def fill_matrix(X: np.ndarray, mixture: GaussianMixture) -> np.ndarray:
    """Fills an incomplete matrix according to a mixture model

    Args:
        X: (n, d) array of incomplete data (incomplete entries = 0)
        mixture: a mixture of gaussians

    Returns
        np.ndarray: a (n, d) array with completed data
    """
    mu, dum_var, dum_p = mixture
    post, _ = estep(X, mixture)
    x_est = post @ mu
    return np.where(X == 0, x_est, X)
```

Press ESC then TAB or click outside of the code editor to exit

Correct

```
def fill_matrix(X: np.ndarray, mixture: GaussianMixture) -> np.ndarray:
    """Fills an incomplete matrix according to a mixture model

    Args:
        X: (n, d) array of incomplete data (incomplete entries =0)
        mixture: a mixture of gaussians

    Returns
        np.ndarray: a (n, d) array with completed data
    """
    n, d = X.shape
    X_pred = X.copy()
    K, _ = mixture.mu.shape

    for i in range(n):
        mask = X[i, :] != 0
        mask0 = X[i, :] == 0
        post = np.zeros(K)
        for j in range(K):
            log_likelihood = log_gaussian(X[i, mask], mixture.mu[j, mask],
                                          mixture.var[j])
            post[j] = np.log(mixture.p[j]) + log_likelihood
        post = np.exp(post - logsumexp(post))
        X_pred[i, mask0] = np.dot(post, mixture.mu[:, mask0])
    return X_pred


def log_gaussian(x: np.ndarray, mean: np.ndarray, var: float) -> float:
    """Computes the log probablity of vector x under a normal distribution

    Args:
        x: (d, ) array holding the vector's coordinates
        mean: (d, ) mean of the gaussian
        var: variance of the gaussian

    Returns:
        float: the log probability
    """
    d = len(x)
    log_prob = -d / 2.0 * np.log(2 * np.pi * var)
    log_prob -= 0.5 * ((x - mean)**2).sum() / var
    return log_prob
```

Test results

```
CORRECT
```

See full output

See full output

Solution:

To complete the row $x_i^{(u)}$ for all values of $i \notin C$ where C is the set of observed values, we have

$$x_i^{(u)} = \sum_{j=1}^{k} p\left(j|u\right) \mu_i^{(j)}$$

Submit You have used 1 of 20 attempts

Answers are displayed within the problem

Comparing with gold targets

1.0/1.0 point (graded)
Test the accuracy of your predictions against actual target values by loading the complete matrix `X_gold = np.loadtxt('netflix_complete.txt')` and measuring the root mean squared error between the two matrices using `common.rmse(X_gold, X_pred)`. Use your best mixture for $K = 12$ from the first question of this tab to generate the results.

$\mathrm{RMSE} =$ 0.4804908505400686 ✓ **Answer:** 0.4804908505400684

Correction note (Aug 8): Since the `mstep` function has been defined differently since Aug 8, the root mean squared error will also be changed. But note that the grader will also accept RMSE resulting from from the earlier and current versions.

Submit You have used 1 of 10 attempts

Answers are displayed within the problem

More Challenge: Collaborative Filtering Using Matrix Factorization and Neural Networks

Now that you have solved use the Expectation Maximization (EM) algorithm to perform collaborative filtering, you may also want to try solving the same task using matrix factorization as discussed in earlier in the course. Here are some steps you could follow in that direction:

1. Implement and test matrix factorization directly. (You will actually find it easier to implement, and will likely work better.)

2. Reformulate the matrix factorization model as a neural network model where you feed in one-hot vectors of the user and movie and predict outcome for each pair.

3. Incorporate additional feature information available about the user/movie into the neural network model.

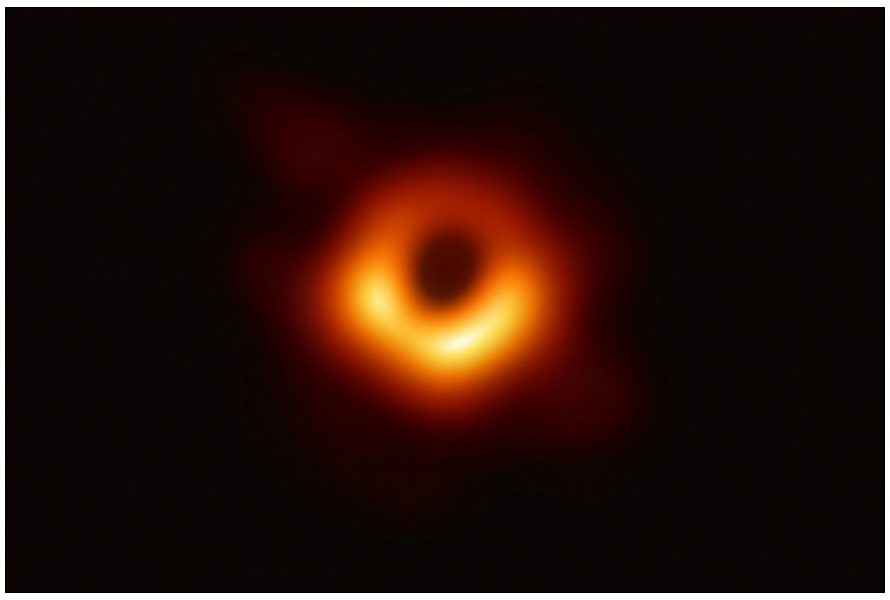
An Application of the EM algorithm: Black Hole Imaging

You may wonder where else the EM algorithm is used.

The following story will draw a connection between the EM algorithm and the first black hole "image" that was uncovered to the world on April 10, 2019. The material is mainly based on the work of Katherine L. Bouman during her PhD at MIT.

The Sparse Reconstruction Problem in Black Hole Imaging

Say we want to obtain a picture of a black hole. Could we just take a “good" camera and directly take a "picture" of it? As you may have guessed, the task is not that easy.

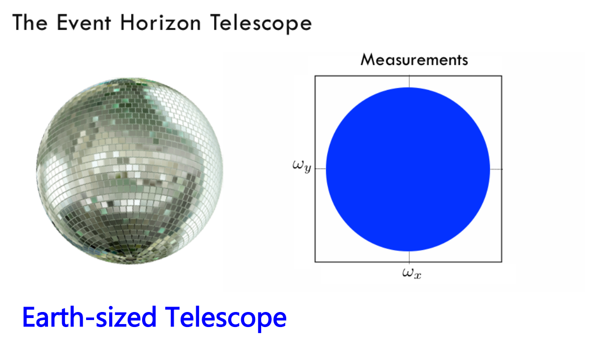


The main issue is that black holes are too far away from us human beings. In fact, the black hole in the picture above is roughly 55 million light years from earth: it is a supermassive black hole at the heart of Messier 87, or M87, a galaxy within the Virgo galaxy cluster (so even if you can travel in a spaceship that moves as fast as light, you will need 55 million years to get there from home). This long distance means that taking a picture of this blackhole is roughly as difficult of taking a picture of an orange on the Moon surface from earth.

However, current photographing technology only allows us to take a picture of the Moon surface in which each pixel captures the size of millions of oranges. A high-resolution telescope that could directly take a picture of an object that “small" in terms of its distance to us would need be the size of the entire earth.

Let us assume for now we have turned our entire earth to a telescope - then could we really observe the black hole? The answer is yes. Even though the black hole, as its name suggests, absorbs everything including light, near its outer marginal surface there is something observable - this is the so-called “event horizon." And this is really what the telescopes are really observing, and that's also why those telescopes of observing the black hole is called the Event Horizon Telescope (EHT).

Back to reality, we do not have such large telescopes. Instead, what we can do is to use several much smaller telescopes, at different locations (Mexico, Hawaii, Chile, Spain, South Pole, and Arizona), to receive the measurements from the black hole, and then "piece together" these to reconstruct the full image. The technique behind this multiple-telescope combination for imaging is called Very Long Baseline Interferometry (VLBI).



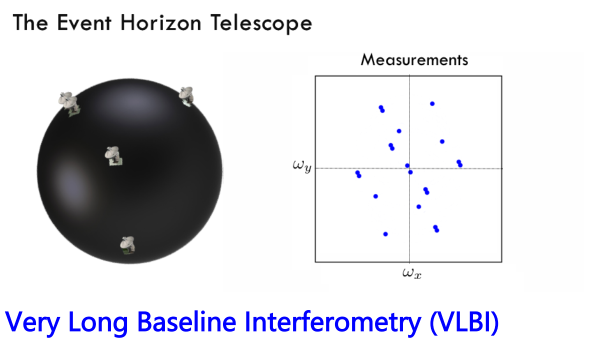

The Event Horizon Telescope
Measurements
ω_y
ω_x
Very Long Baseline Interferometry (VLBI)

*Image credit:*https://people.csail.mit.edu/klbouman/pw/papers_and_presentations/bouman_tedxslides.pptx
(Left) Using an Earth-sized EHT, we would obtain an image with dense measurements
(Right) Using VLBI, we obtain only sparse measurements

Using VLBI, we can only obtain sparse measurements from the black hole. (There are some ways to increase the amount of data we have. For example, since the Earth is rotating, the EHTs also move to different locations so as we are able to get more measurements.).

With the collected data, the problem becomes an algorithmic one: how do we reconstruct the "image" from these measurements, where by "image" we mean not the full and real one, but only one that is close to the true image with high probability.

The scale of this machine learning problem is enormous. The EHT teams took only 5 days to collect the roughly 3500TB of data. The researchers then took two years to clean the data, and train and adjust the machine learning algorithms in order to get the final image we now see.

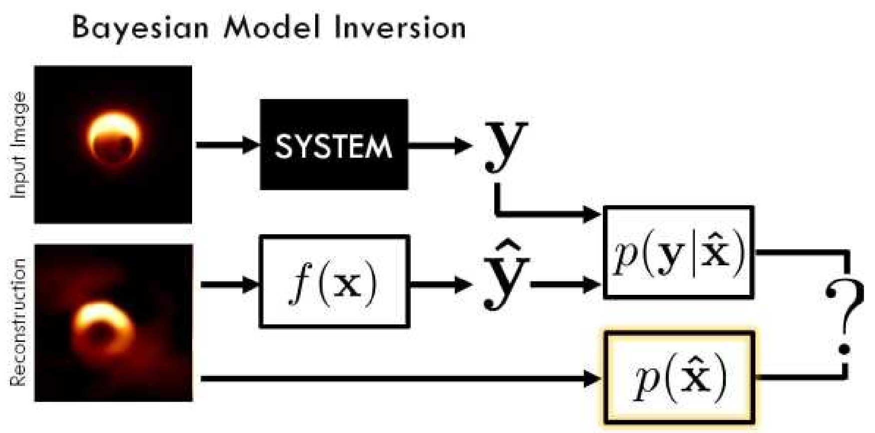
Single Image VLBI Reconstruction of Static Sources

For simplicity, we first consider the model of static sources, i.e., let us assume the image of the black hole would not change over time. Then, according to physics we know the image $\mathbf{x}$ and our measurements $\mathbf{y}$ has a functional dependence of f, that is:

$$f(\mathbf{x}) = \mathbf{y}$$

If we have dense measurements, we could perform the inverse function operation $f^{-1}(\mathbf{y})$ to get the desired image. However, we only have sparse measurements, so the goal here is to get an approximation $\hat{\mathbf{x}}$ of $\mathbf{x}$ from the measurements. Even for VLBI reconstruction of static sources, many of the traditional methods in image reconstruction does not work, because there are delays in the time when we get the measurements depending on the location of the EHTs. Since the EHTs are spread over the Earth, these delays can be viewed as random.

What about a Bayesian model? Following Katherine Bouman, we can start with the simplest model, the Gaussian Mixture Model (GMM), which we have seen in the lectures!



Specifically, we assume

$$\mathbf{y} \sim \mathcal{N}_{\mathbf{y}}(f(\mathbf{x}), \mathbf{R})$$

so that the measurements follows a Gaussian distribution of mean $f(\mathbf{x})$. We define each overlapping patch $\mathbf{z}$ of image $\mathbf{x}$ as being a sample from a GMM with C clusters. Here patch samples (patches of the image) include other planets, galaxies, simulated black hole images consistent with General Relativity, and even daily life pictures, and we denote them as the collection $\{\mathbf{z}_n : n = 1 : \ldots, N\}$. We assume these samples are from a GMM model with C clusters, that is,

$$\mathbf{z}_n = \mathbf{P}_n \mathbf{x} \sim \sum_{c=1}^{C} \pi_c \mathcal{N}_p(\mu_c, \mathbf{\Lambda}_c)$$

where matrix $\mathbf{P}_n$ extracts the n-th overlapping patch, $\mathbf{z}_n$, from $\mathbf{x}$, and π_c is the mixture component weight of the c-th cluster. From this relation we can write the probability of each patch $\mathbf{z}$ as

$$p(\mathbf{z}) = \sum_{c=1}^{C} \frac{\pi_c}{\sqrt{|2\pi\mathbf{\Lambda}|}} \exp\left(-\frac{1}{2}(\mathbf{z}-\mu_c)^T \mathbf{\Lambda}^{-1}(\mathbf{z}-\mu_c)\right)$$

We use EM algorithm to train the GMM patch prior. Namely, given N patches $\{\mathbf{z}_n : n = 1 : \ldots, N\}$, we iterate between the E-steps and M-steps as follows

- E-step: find the expected likelihood of a patch belonging to each cluster.

$$\xi_c(\mathbf{z}_n) = \frac{\pi_c \mathcal{N}_{\mathbf{z_n}}(\mu_c, \mathbf{\Lambda}_c)}{\sum_{k=1}^{C} \pi_k \mathcal{N}_{\mathbf{z_n}}(\mu_k, \mathbf{\Lambda}_k)}$$

- M-step: find the model parameters that maximize the expected value of the log likelihood function.

$$\mu_c = \frac{\sum_{n=1}^{N} \xi_c(\mathbf{z}_n)\mathbf{z}_n}{\sum_{n=1}^{N} \xi_c(\mathbf{z}_n)}, \mathbf{\Lambda}_c = \frac{\sum_{n=1}^{N} \xi_c(\mathbf{z}_n)(\mathbf{z}_n-\mu_c)(\mathbf{z}_n-\mu_c)^T}{\sum_{n=1}^{N} \xi_c(\mathbf{z}_n)}, \pi_c = \frac{1}{N}\sum_{n=1}^{N} \xi_c(\mathbf{z}_n)$$

Since we have defined the likelihood of an image patch, and not the likelihood of the image itself, we can no longer write a posterior distribution of $\mathbf{x}$ given $\mathbf{y}$. However, we still are able to write an optimizing cost function in the form

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} [\chi(\mathbf{x}, \mathbf{y}) - \beta R(\mathbf{x})]$$

in which $\chi(\mathbf{x}, \mathbf{y}) = \frac{1}{2}(\mathbf{y} - f(\mathbf{x}))^T \mathbf{R}^{-1}(\mathbf{y} - f(\mathbf{x}))$ denotes the deviation of $\mathbf{y}$ under conditional distribution $p(\mathbf{y}|\mathbf{x})$ and $R(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N} \log[p(\mathbf{P}_n\mathbf{x})]$ denotes the regularization term. Here $\beta = \frac{1}{2}$.

Definitely, we then need to do some optimization to find the $\arg\min$. Again the usual EM steps won't follow directly here but a similar iterative procedure called "Half Quadratic Splitting" works for this.

1. Solve for $\{\mathbf{z}^n\}$ given $\hat{\mathbf{x}}$: we first find the most likely cluster by evaluating the log-likelihood of each cluster

$$\hat{c}^n = \arg\max_{c} \left[\log \pi_c - \frac{1}{2}\log|2\pi\mathbf{\Lambda}_c| - \frac{1}{2}(\mathbf{P}_n\hat{\mathbf{x}} - \mu_c)^T \mathbf{\Lambda}_c^{-1}(\mathbf{P}_n\hat{\mathbf{x}} - \mu_c)\right]$$

We introduce a weighting parameter γ^{-1}, that indicates the variance of noise in the current estimate of $\hat{\mathbf{x}}$. Under the posterior distribution

$$p(\mathbf{z}^n | \mathbb{P}_n\hat{\mathbf{x}}; \mu_{\hat{c}^n}, \mathbf{\Lambda}_{\hat{c}^n}) \propto \mathcal{N}_{\mathbf{z}^n}(\mathbf{P}_n\hat{\mathbf{x}}, \gamma^{-1}\mathbb{1}) \mathcal{N}_{\mathbf{z}^n}(\mu_{\hat{c}^n}, \mathbf{\Lambda}_{\hat{c}^n})$$

we estimate the best patch using Wiener filtering

$$\mathbf{z}^n = \mu_{\hat{c}^n} + \boldsymbol{\Lambda}_{\hat{c}^n} \left(\gamma^{-1}\mathbb{1} + \boldsymbol{\Lambda}_{\hat{c}^n}\right) \left(\mathbf{P}_n \hat{\mathbf{x}} - \mu_{\hat{c}^n}\right)$$

2. Solve for $\hat{\mathbf{x}}$ given $\{\mathbf{z}^n\}$: We re-define the regularization term as

$$R(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^{N} \sum_{n=1}^{N} \left[\frac{\gamma}{2} \left(\mathbf{P}_n \mathbf{x} - \mathbf{z}^n\right)^T \left(\mathbf{P}_n \mathbf{x} - \mathbf{z}^n\right) - \log p\left(z^n\right)\right]$$

and then given $\{\mathbf{z}^n\}$ and if we restrict to linear measurements such that $f(\mathbf{x}) = \mathbf{F}\mathbf{x}$ we would have a closed form of $\hat{\mathbf{x}}$ as (you should be able to work this out yourself by doing the MAP minimization)

$$\hat{\mathbf{x}} = \left(\mathbf{F}^T \mathbf{R}^{-1} \mathbf{F} + \frac{\beta\gamma}{N} \sum_{n=1}^{N} \mathbf{P}_n^T \mathbf{P}_n\right)^{-1} \left(\mathbf{F}^T \mathbf{R}^{-1} \mathbf{y} + \frac{\beta\gamma}{N} \sum_{n=1}^{N} \mathbf{P}_n^T \mathbf{z}_n\right)$$

3. Iterate between step (1) and step (2) with increasing γ values of 1,4,8,16, 32, 64, 128, 256, and 512. Note when $\gamma \to \infty$ patches $\mathbf{P}_n \mathbf{x}$ are restricted to be equal to their auxiliary patch $\mathbf{z}_n$.

That's it! This is the methodology that lets you reconstruct nice-resolution astronomical images from VLBI measurements.

Endnotes

Of course, the methodology presented in the former section is not what the scientists applied exactly for getting the black hole image. There are a lot of issues we have not considered, for example, the image of black hole might change over time due to astronomical movements. In Section 4 of Katherine L. Bouman's PhD thesis, she addresses the issue by proposing more advanced model and algorithms based on Gaussian Hidden Markov Models (GHMM). The Markov model here is used for capturing the dynamics of the object.

The main takeaway of the story is that mixture models and EM algorithms are very powerful tools for machine learning tasks in science. Besides the exciting photographing of black holes, it is also widely used in many other applications, including financial modeling, choice model in revenue management, predictive maintenance, and so on. Compared to deep learning, the mixture models often give your better interpretability especially when you have a good understanding of the underlying model dynamics.

Hide

Discussion

Show Discussion

Topic: Unit 4 Unsupervised Learning (2 weeks) :Project 4: Collaborative Filtering via Gaussian Mixtures / 8. Using the mixture model for collaborative filtering