Alternating Minimization (and Friends)

Lecture 7: 6.883, Spring 2016

Suvrit Sra

Massachusetts Institute of Technology

Feb 29, 2016

Background: Coordinate Descent

For $x \in \mathbb{R}^n$ consider

$$\min f(x) = f(x_1, x_2, \ldots, x_n)$$

Ancient idea: optimize over individual coordinates

Coordinate descent

Coordinate descent

- For $k = 0, 1, \ldots$
  - Pick an index i from $\{1, \ldots, n\}$

Coordinate descent

Coordinate descent

- For $k = 0, 1, \ldots$
  - Pick an index i from $\{1, \ldots, n\}$
  - Optimize the ith coordinate

$$x_i^{k+1} \leftarrow \operatorname*{argmin}_{\xi \in \mathbb{R}} f(\underbrace{x_1^{k+1}, \ldots, x_{i-1}^{k+1}}_{\textbf{done}}, \underbrace{\xi}_{\textbf{current}}, \underbrace{x_{i+1}^{k}, \ldots, x_n^{k}}_{\textbf{todo}})$$

Coordinate descent

Coordinate descent

- For $k = 0, 1, \ldots$
  - Pick an index i from $\{1, \ldots, n\}$
  - Optimize the ith coordinate

$$x_i^{k+1} \leftarrow \underset{\xi \in \mathbb{R}}{\operatorname{argmin}}\, f(\underbrace{x_1^{k+1}, \ldots, x_{i-1}^{k+1}}_{\text{done}}, \underbrace{\xi}_{\text{current}}, \underbrace{x_{i+1}^{k}, \ldots, x_n^{k}}_{\text{todo}})$$

- Decide when/how to stop; *return* x^k

! x_i^{k+1} **overwrites** value in x_i^k (implementation)

Coordinate descent

- ♣ One of the simplest optimization methods
- ♣ Various ideas for next coordinate to optimize

Coordinate descent

- ♣ One of the simplest optimization methods
- ♣ Various ideas for next coordinate to optimize
- ♣ Old idea: Gauss-Seidel, Jacobi methods for $Ax = b$

Coordinate descent

- ♣ One of the simplest optimization methods
- ♣ Various ideas for next coordinate to optimize
- ♣ Old idea: Gauss-Seidel, Jacobi methods for $Ax = b$
- ♣ Can be "slow"; sometimes very competitive

Coordinate descent

- ♣ One of the simplest optimization methods
- ♣ Various ideas for next coordinate to optimize
- ♣ Old idea: Gauss-Seidel, Jacobi methods for $Ax = b$
- ♣ Can be "slow"; sometimes very competitive
- ♣ Gradient, stochastic gradient also "slow"

Coordinate descent

- ♣ One of the simplest optimization methods
- ♣ Various ideas for next coordinate to optimize
- ♣ Old idea: Gauss-Seidel, Jacobi methods for $Ax = b$
- ♣ Can be "slow"; sometimes very competitive
- ♣ Gradient, stochastic gradient also "slow"
- ♣ But scalable (eg: libsvm)

Coordinate descent

- ♣ One of the simplest optimization methods
- ♣ Various ideas for next coordinate to optimize
- ♣ Old idea: Gauss-Seidel, Jacobi methods for $Ax = b$
- ♣ Can be "slow"; sometimes very competitive
- ♣ Gradient, stochastic gradient also "slow"
- ♣ But scalable (eg: libsvm)
- ♣ Renewed interest; esp. stochastic CD

Coordinate descent

- ♣ One of the simplest optimization methods
- ♣ Various ideas for next coordinate to optimize
- ♣ Old idea: Gauss-Seidel, Jacobi methods for $Ax = b$
- ♣ Can be "slow"; sometimes very competitive
- ♣ Gradient, stochastic gradient also "slow"
- ♣ But scalable (eg: libsvm)
- ♣ Renewed interest; esp. stochastic CD
- ♣ Notice: in general CD is "derivative free"

Example: Least-squares

Assume $A \in \mathbb{R}^{m \times n}$

$$\min \ \|Ax - b\|_2^2$$

Example: Least-squares

Assume $A \in \mathbb{R}^{m \times n}$

$$\min \ \|Ax - b\|_2^2$$

Coordinate descent update

$$x_j \leftarrow \frac{\sum_{i=1}^m a_{ij}\left(b_i - \sum_{l \neq j} a_{il} x_l\right)}{\sum_{i=1}^m a_{ij}^2}$$

(dropped superscripts, since we overwrite)

Coordinate descent remarks

Advantages

- ◇ Each iteration usually cheap (single variable optimization)
- ◇ No extra storage vectors needed
- ◇ **No stepsize tuning** 🙂
- ◇ No other pesky parameters that must be tuned
- ◇ Simple to implement
- ◇ Works well for large-scale problems
- ◇ Currently quite popular; parallel versions exist

Coordinate descent remarks

Advantages

- ◇ Each iteration usually cheap (single variable optimization)
- ◇ No extra storage vectors needed
- ◇ **No stepsize tuning**
- ◇ No other pesky parameters that must be tuned
- ◇ Simple to implement
- ◇ Works well for large-scale problems
- ◇ Currently quite popular; parallel versions exist

Disadvantages

- ♠ Tricky if single variable optimization is hard
- ♠ Convergence theory can be complicated
- ♠ Can slow down near optimum
- ♠ Non-differentiable case more tricky

Block coordinate descent (BCD)

$$\begin{aligned} \min \quad & f(x) := f(x_1, \ldots, x_n) \\ & x \in \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_m. \end{aligned}$$

Block coordinate descent (BCD)

$$\begin{aligned} \min \quad & f(x) := f(x_1, \ldots, x_n) \\ & x \in \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_m. \end{aligned}$$

Gauss-Seidel update

$$x_i^{k+1} \leftarrow \operatorname*{argmin}_{\xi \in \mathcal{X}_i} f(\underbrace{x_1^{k+1}, \ldots, x_{i-1}^{k+1}}_{\text{done}}, \underbrace{\xi}_{\text{current}}, \underbrace{x_{i+1}^k, \ldots, x_m^k}_{\text{todo}})$$

Block coordinate descent (BCD)

$$\begin{aligned} \min \quad & f(x) := f(x_1, \ldots, x_n) \\ & x \in \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_m. \end{aligned}$$

Gauss-Seidel update

$$x_i^{k+1} \leftarrow \operatorname*{argmin}_{\xi \in \mathcal{X}_i} f(\underbrace{x_1^{k+1}, \ldots, x_{i-1}^{k+1}}_{\text{done}}, \underbrace{\xi}_{\text{current}}, \underbrace{x_{i+1}^{k}, \ldots, x_m^{k}}_{\text{todo}})$$

Jacobi update (easy to parallelize)

$$x_i^{k+1} \leftarrow \operatorname*{argmin}_{\xi \in \mathcal{X}_i} f(\underbrace{x_1^{k}, \ldots, x_{i-1}^{k}}_{\text{don't clobber}}, \underbrace{\xi}_{\text{current}}, \underbrace{x_{i+1}^{k}, \ldots, x_m^{k}}_{\text{todo}})$$

Two block BCD

$$\min\ f(x,y), \quad \text{s.t. } x \in \mathcal{X}, y \in \mathcal{Y}.$$

Theorem (Grippo & Sciandrone (2000)). Let f be continuously differentiable; and $\mathcal{X}$, $\mathcal{Y}$ be closed and convex sets. Assuming both subproblems have solutions, and that the sequence $\{(x^k, y^k)\}$ has limit points. Then, every limit point is stationary.

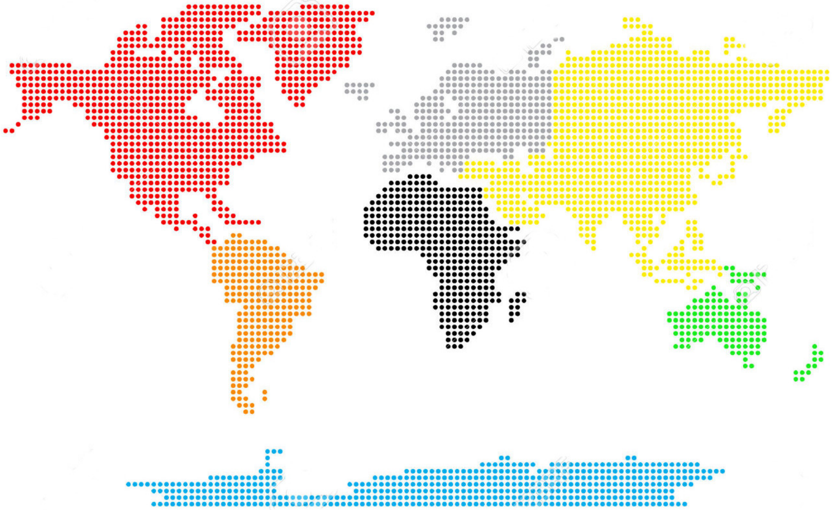
Two block BCD

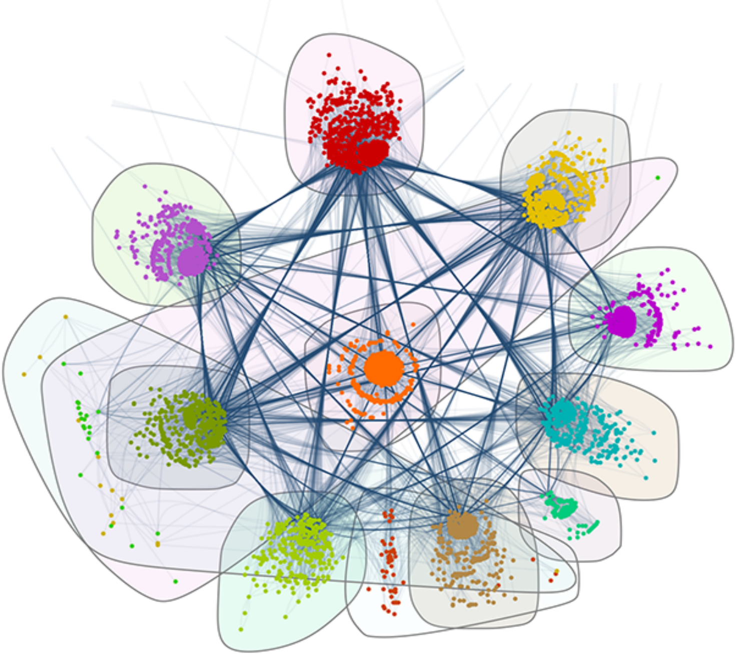
$$\min \; f(x,y), \quad \text{s.t. } x \in \mathcal{X}, y \in \mathcal{Y}.$$

Theorem (Grippo & Sciandrone (2000)). Let f be continuously differentiable; and $\mathcal{X}$, $\mathcal{Y}$ be closed and convex sets. Assuming both subproblems have solutions, and that the sequence $\{(x^k, y^k)\}$ has limit points. Then, every limit point is stationary.

- Subproblems need not have **unique solutions**
- BCD for 2 blocks aka **Alternating Minimization**

AltMin





Clustering

Original matrix

a	+	a	+	+
z	○	z	○	○
a	+	a	+	+
—	*	—	*	*
—	*	—	*	*
z	○	z	○	○

Clustering

Clustered matrix

a	a	+	+	+
z	z	○	○	○
a	a	+	+	+
—	—	*	*	*
—	—	*	*	*
z	z	○	○	○

After clustering and permutation

Clustering

Co-clustered matrix

a	a	+	+	+
a	a	+	+	+
z	z	○	○	○
z	z	○	○	○
—	—	*	*	*
—	—	*	*	*

After co-clustering and permutation

Clustering

- Let $X \in \mathbb{R}^{m \times n}$ be the input matrix
- Cluster columns of X
- Well-known k-means clustering problem can be written as

$$\min_{B,C} \quad \tfrac{1}{2}\|X - BC\|_F^2 \quad \text{s.t. } C^T C = \mathrm{Diag}(\text{sizes})$$

  where $B \in \mathbb{R}^{m \times k}$, and $C \in \{0, 1\}^{k \times n}$.
- Optimization problem with 2 blocks; $\min F(B, C)$

Clustering

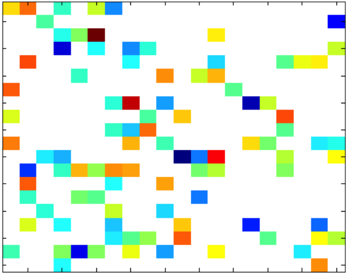
- Let $X \in \mathbb{R}^{m \times n}$ be the input matrix
- Cluster columns of X
- Well-known k-means clustering problem can be written as

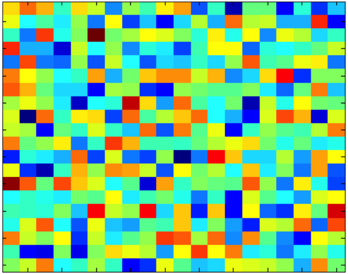
$$\min_{B,C} \quad \tfrac{1}{2}\|X - BC\|_F^2 \quad \text{s.t. } C^T C = \mathrm{Diag}(\text{sizes})$$

  where $B \in \mathbb{R}^{m \times k}$, and $C \in \{0, 1\}^{k \times n}$.
- Optimization problem with 2 blocks; $\min F(B, C)$

Exercise: Write co-clustering in matrix form
Hint: Write using 3 blocks

Matrix Completion

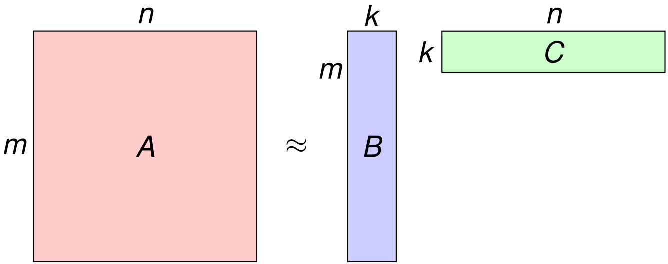




- Given matrix with missing entries, fill in the rest
- Recall Netflix million-$ prize problem
- Given User-Movie ratings, recommend movies to users

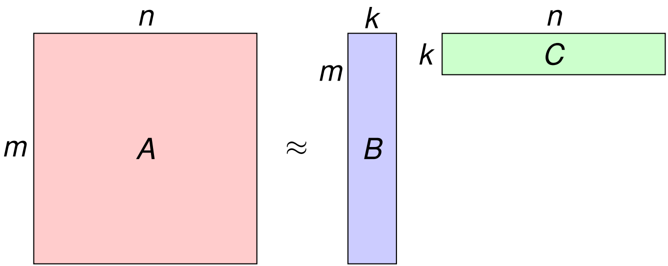
Matrix Completion

- Input: matrix A with missing entries
- "Predict" missing entries to "complete" the matrix
- Netflix: movies x users matrix; available entries were ratings given to movies by users
- Task: predict missing entries
- Winning methods based on **low-rank matrix completion**

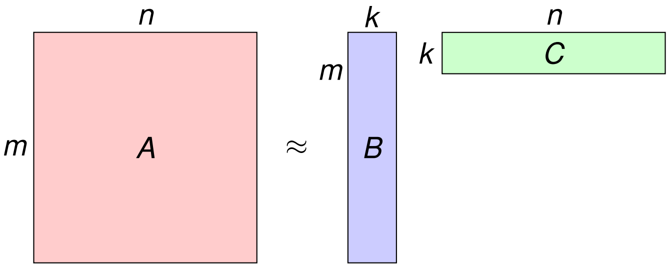


Matrix Completion

- Input: matrix A with missing entries
- "Predict" missing entries to "complete" the matrix
- Netflix: movies x users matrix; available entries were ratings given to movies by users
- Task: predict missing entries
- Winning methods based on **low-rank matrix completion**



Matrix Completion



Task: Recover matrix A given a sampling of its entries

Theorem: Can recover most low-rank matrices!

Matrix completion

$$\begin{aligned} \min \quad & \text{rank}(X) \\ \text{s.t. } X_{ij} = A_{ij}, \quad & \forall (i,j) \in \Omega = \text{Rating pairs} \end{aligned}$$

another formulation

$$\begin{aligned} \min \quad & \sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2 \\ \text{s.t. } & \text{rank}(X) \leq k. \end{aligned}$$

———— ○ ————

Both are **NP-Hard** problems

Matrix completion

$$\begin{aligned} \min \quad & \textcolor{red}{\mathrm{rank}}(X) \\ \text{s.t. } X_{ij} = A_{ij}, \quad & \forall (i,j) \in \Omega = \text{Rating pairs} \end{aligned}$$

another formulation

$$\min \quad \sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2$$

$$\text{s.t. } \mathrm{rank}(X) \le k.$$

——— ○ ———

Both are **NP-Hard** problems

convex relaxation

$$\mathrm{rank}(X) \le k \mapsto \left(\|X\|_* := \sum_{j=1}^{m} \sigma_j(X)\right) \le k$$

Candes and Recht prove that convex relaxation solves matrix completion (under assumptions on Ω and A)

Matrix completion

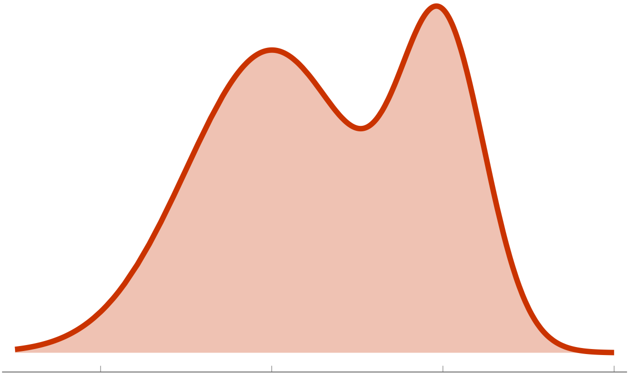
- convex relaxation does not scale well
- commonly used heuristic is Alternating Minimization
- Write $X = BC$ where B is $m \times k$, C is $k \times n$

$$\min_{B,C} \quad F(B, C) := \|P_\Omega(A) - P_\Omega(BC)\|_F^2,$$

where $[P_\Omega(X)]_{ij} = X_{ij}$ for $(i, j) \in \Omega$, and 0 otherwise.

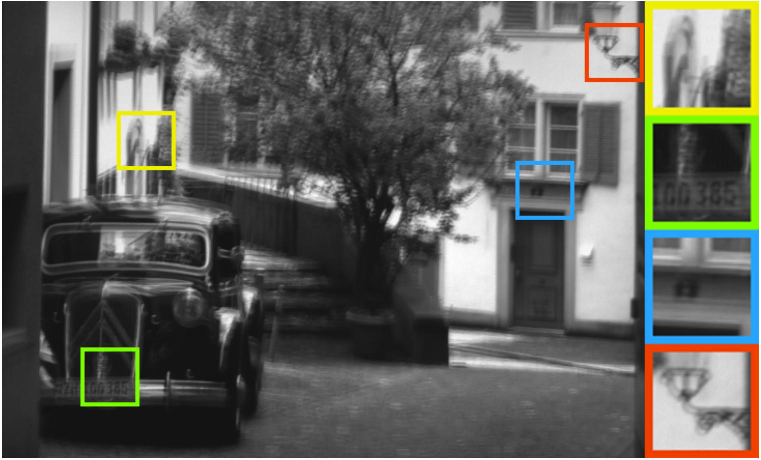
Result:

- Initialize B, C using SVD of $P_\Omega(A)$
- AltMin iterations to compute B and C
- Can be shown (Jain, Netrapalli, Sanghavi 2012) under assumptions on Ω (uniform sampling) and A (incoherence, most entries similar in magnitude) that AltMin generates B and C such that $\|A - BC\|_F \leq \epsilon$ after $O(\log(1/\epsilon))$ steps.



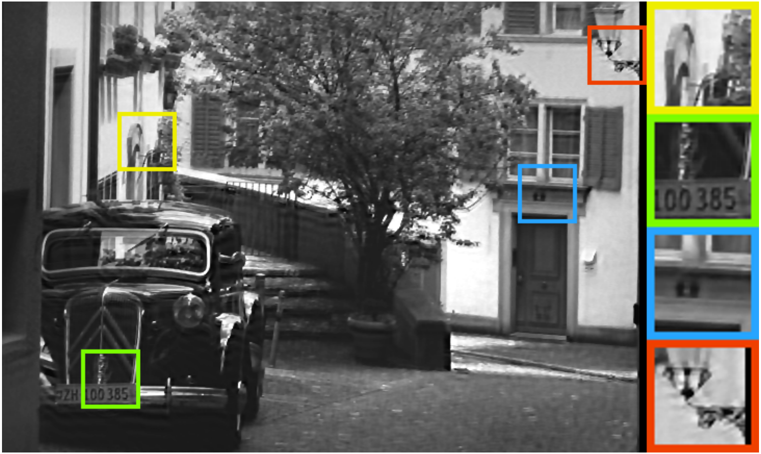
$$p(x) := \sum_{k=1}^{K} \pi_k p_{\mathcal{N}}(x; \Sigma_k, \mu_k)$$

Gaussian Mixture Model



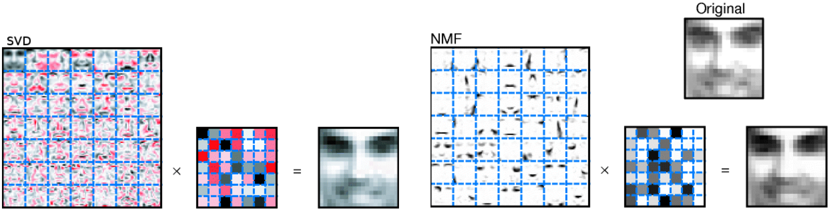
$$\frac{1}{2}\|a * x - y\|^2 + \lambda\Omega(x)$$

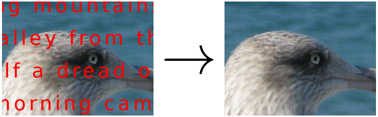
Image deblurring



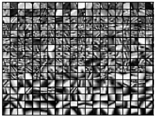
$$\frac{1}{2}\|a * x - y\|^2 + \lambda\Omega(x)$$

Image deblurring





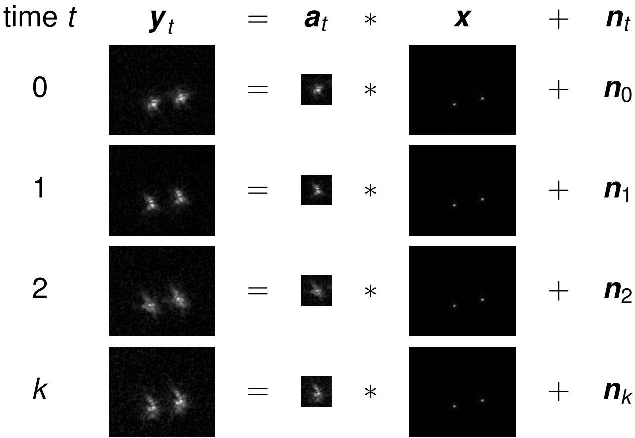
(Mairal et al., 2010)



$$\sum_{i=1}^{n} \frac{1}{2}\|\boldsymbol{y}_i - \boldsymbol{D}\boldsymbol{c}_i\|^2 + \Omega_1(\boldsymbol{c}_i) + \Omega_2(\boldsymbol{D})$$

Dict. learning, matrix factorization

Online matrix factorization



Non-online formulation

$$\begin{bmatrix} | & \vdots & | \\ y_1 & | & y_n \\ | & \vdots & | \end{bmatrix} \approx \begin{bmatrix} | & \vdots & | \\ a_1 & | & a_t \\ | & \vdots & | \end{bmatrix} * x$$

$$\boxed{\text{Rewrite: } a * x = Ax = Xa}$$

$$\begin{bmatrix} y_1 & y_2 & \cdots & y_t \end{bmatrix} \approx X \begin{bmatrix} a_1 & a_2 & \cdots & a_t \end{bmatrix}$$

$$\boxed{Y \approx XA}$$

Why online?

Example, 5000 frames of size 512×512
$Y_{262144 \times 5000} \approx X_{262144 \times 262144} A_{262144 \times 5000}$

Without structure $\approx$ 70 billion parameters!
With structure, $\approx$ 4.8 **million parameters!**

Why online?

Example, 5000 frames of size 512×512
$Y_{262144 \times 5000} \approx X_{262144 \times 262144} A_{262144 \times 5000}$

Without structure $\approx$ 70 billion parameters!
With structure, $\approx$ 4.8 **million parameters!**

Despite structure, alternating minimization **impractical**
Fix X, solve for A, requires updating $\approx$ 4.5 million params

Online matrix factorization

$$\min_{A_t, x} \quad \sum_{t=1}^{T} \tfrac{1}{2}\|y_t - A_t x\|^2 + \Omega(x) + \Gamma(A_t)$$

Online matrix factorization

$$\min_{A_t, x} \quad \sum_{t=1}^{T} \frac{1}{2}\|y_t - A_t x\|^2 + \Omega(x) + \Gamma(A_t)$$

Initialize guess $\boldsymbol{x}_0$
For $t = 1, 2, \ldots$
1. Observe image $\boldsymbol{y}_t$;

Online matrix factorization

$$\min_{A_t, x} \quad \sum_{t=1}^{T} \tfrac{1}{2}\|y_t - A_t x\|^2 + \Omega(x) + \Gamma(A_t)$$

Initialize guess $\boldsymbol{x}_0$
For $t = 1, 2, \ldots$

1. Observe image $\boldsymbol{y}_t$;
2. Use $\boldsymbol{x}_{t-1}$ to **estimate** $\boldsymbol{A}_t$

Online matrix factorization

$$\min_{A_t, x} \quad \sum_{t=1}^{T} \tfrac{1}{2}\|y_t - A_t x\|^2 + \Omega(x) + \Gamma(A_t)$$

Initialize guess $\boldsymbol{x}_0$
For $t = 1, 2, \ldots$
1. Observe image $\boldsymbol{y}_t$;
2. Use $\boldsymbol{x}_{t-1}$ to **estimate** $\boldsymbol{A}_t$
3. Solve **optimization subproblem** to obtain $\boldsymbol{x}_t$

Online matrix factorization

$$\min_{A_t,x} \quad \sum_{t=1}^{T} \tfrac{1}{2}\|y_t - A_t x\|^2 + \Omega(x) + \Gamma(A_t)$$

Initialize guess $\boldsymbol{x}_0$
For $t = 1, 2, \ldots$
1. Observe image $\boldsymbol{y}_t$;
2. Use $\boldsymbol{x}_{t-1}$ to **estimate** $\boldsymbol{A}_t$
3. Solve **optimization subproblem** to obtain $\boldsymbol{x}_t$

Video

Step 2. Model, estimate blur A_t — separate lecture

Step 3. convex subproblem — reuse convex subroutines

Do Steps 2, 3 **online** $\Longrightarrow$ realtime processing!

Some math: NMF

Nonnegative matrix factorization

We want a low-rank approximation $A \approx BC$

Nonnegative matrix factorization

We want a low-rank approximation $A \approx BC$

- SVD yields dense B and C
- B and C contain negative entries, even if $A \geq 0$
- SVD factors may not be that easy to interpret

Nonnegative matrix factorization

We want a low-rank approximation $A \approx BC$

- SVD yields dense B and C
- B and C contain negative entries, even if $A \geq 0$
- SVD factors may not be that easy to interpret

$$\boxed{\textbf{NMF} \text{ imposes } B \geq 0,\ C \geq 0}$$

Algorithms

$$\boxed{A \approx BC \quad \text{s.t. } B, C \geq 0}$$

Least-squares NMF

$$\min \quad \tfrac{1}{2}\|A - BC\|_F^2 \quad \text{s.t. } B, C \geq 0.$$

KL-Divergence NMF

$$\min \quad \sum\nolimits_{ij} a_{ij} \log \frac{(BC)_{ij}}{a_{ij}} - a_{ij} + (BC)_{ij} \quad \text{s.t. } B, C \geq 0.$$

Algorithms

$$\boxed{A \approx BC \quad \text{s.t. } B, C \geq 0}$$

Least-squares NMF

$$\min \quad \tfrac{1}{2}\|A - BC\|_F^2 \quad \text{s.t. } B, C \geq 0.$$

KL-Divergence NMF

$$\min \quad \sum\nolimits_{ij} a_{ij} \log \frac{(BC)_{ij}}{a_{ij}} - a_{ij} + (BC)_{ij} \quad \text{s.t. } B, C \geq 0.$$

- ♣ NP-Hard (Vavasis 2007) – no surprise
- ♣ Recently, Arora et al. showed that if the matrix A has a special "separable" structure, then actually globally optimal NMF is approximately solvable. More recent progress too!
- ♣ We look at only basic methods in this lecture

NMF Algorithms

- Hack: Compute TSVD; “zero-out” negative entries
- Alternating minimization (AM)
- Majorize-Minimize (MM)
- Global optimization (not covered)
- “Online” algorithms (not covered)

AltMin / AltDesc

$$\boxed{\min \quad F(B, C)}$$

Alternating Descent

1. Initialize B^0, $k \leftarrow 0$
2. Compute C^{k+1} s.t. $F(A, B^k C^{k+1}) \leq F(A, B^k C^k)$
3. Compute B^{k+1} s.t. $F(A, B^{k+1} C^{k+1}) \leq F(A, B^k C^{k+1})$
4. $k \leftarrow k + 1$, and repeat until stopping criteria met.

AltMin / AltDesc

$$\boxed{\min \quad F(B, C)}$$

Alternating Descent

1. Initialize B^0, $k \leftarrow 0$
2. Compute C^{k+1} s.t. $F(A, B^k C^{k+1}) \leq F(A, B^k C^k)$
3. Compute B^{k+1} s.t. $F(A, B^{k+1} C^{k+1}) \leq F(A, B^k C^{k+1})$
4. $k \leftarrow k+1$, and repeat until stopping criteria met.

$$F(B^{k+1}, C^{k+1}) \leq F(B^k, C^{k+1}) \leq F(B^k, C^k)$$

AltMin

Alternating Least Squares

$$C = \operatorname*{argmin}_{C} \quad \|A - B^k C\|_{\mathrm{F}}^2;$$

Alternating Least Squares

$$C = \operatorname*{argmin}_{C} \quad \|A - B^k C\|_F^2; \qquad\qquad C^{k+1} \leftarrow \max(0, C)$$

Alternating Least Squares

$$C = \operatorname*{argmin}_C \quad \|A - B^k C\|_F^2; \qquad C^{k+1} \leftarrow \max(0, C)$$

$$B = \operatorname*{argmin}_B \quad \|A - BC^{k+1}\|_F^2; \qquad B^{k+1} \leftarrow \max(0, B)$$

AltMin

Alternating Least Squares

$$C = \underset{C}{\operatorname{argmin}} \quad \|A - B^k C\|_F^2; \qquad C^{k+1} \leftarrow \max(0, C)$$

$$B = \underset{B}{\operatorname{argmin}} \quad \|A - BC^{k+1}\|_F^2; \qquad B^{k+1} \leftarrow \max(0, B)$$

ALS is fast, simple, often effective, but ...

AltMin

Alternating Least Squares

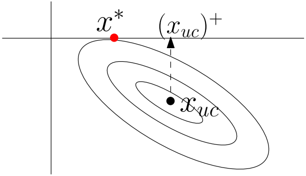
$$C = \operatorname*{argmin}_C \quad \|A - B^k C\|_F^2; \qquad C^{k+1} \leftarrow \max(0, C)$$

$$B = \operatorname*{argmin}_B \quad \|A - BC^{k+1}\|_F^2; \qquad B^{k+1} \leftarrow \max(0, B)$$

ALS is fast, simple, often effective, but ...

$$\|A - B^{k+1}C^{k+1}\|_F^2 \le \|A - B^k C^{k+1}\|_F^2 \le \|A - B^k C^k\|_F^2$$

descent **need not** hold



Alternating Minimization: correctly

Use alternating **nonnegative least-squares**

$$
\begin{aligned}
C^{k+1} &= \operatorname*{argmin}_{C} \quad \|A - B^k C\|_F^2 \quad \text{s.t.} \quad C \geq 0 \\
B^{k+1} &= \operatorname*{argmin}_{B} \quad \|A - BC^{k+1}\|_F^2 \quad \text{s.t.} \quad B \geq 0
\end{aligned}
$$

Advantages: Guaranteed descent. Theory of block-coordinate descent guarantees convergence to *stationary point*.

Disadvantages: more complex; slower than ALS

Just Descent

Majorize-Minimize (MM)

Consider $F(B,C) = \frac{1}{2}\|A - BC\|_F^2$: convex separately in B and C

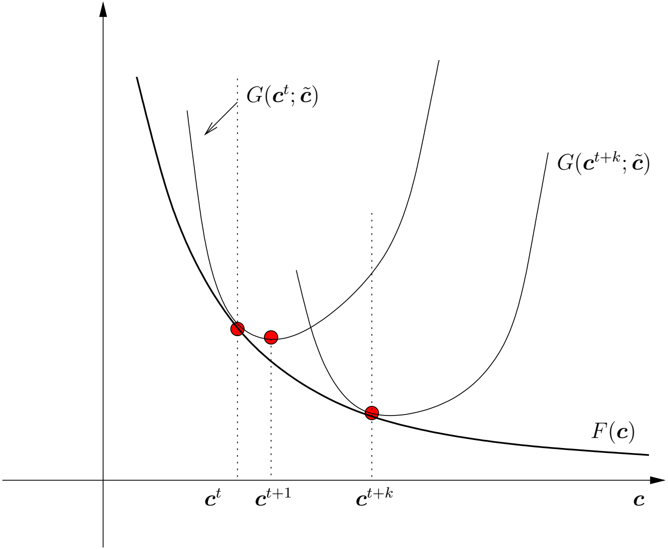
We use $F(C)$ to denote function restricted to C.

Since $F(C)$ *separable* (over cols of C), we just illustrate

$$\min_{c \geq 0} \quad f(c) = \frac{1}{2}\|a - Bc\|_2^2$$

Recall, our aim is: find C_{k+1} such that $F(B_k, C_{k+1}) \leq F(B_k, C_k)$

Majorize-Minimize (MM)



Descent technique

$$\boxed{\min_{c\geq 0} \quad f(c) = \tfrac{1}{2}\|a - Bc\|_2^2}$$

1. Find a function $g(c, \tilde{c})$ that satisfies:

$$\begin{aligned} g(c, c) &= f(c), \quad \text{for all} \quad c, \\ g(c, \tilde{c}) &\geq f(c), \quad \text{for all} \quad c, \tilde{c}. \end{aligned}$$

Descent technique

$$\min_{c \geq 0} \quad f(c) = \tfrac{1}{2}\|a - Bc\|_2^2$$

1. Find a function $g(c, \tilde{c})$ that satisfies:

$$\begin{aligned} g(c, c) &= f(c), \quad \text{for all} \quad c, \\ g(c, \tilde{c}) &\geq f(c), \quad \text{for all} \quad c, \tilde{c}. \end{aligned}$$

2. Compute $c^{t+1} = \operatorname{argmin}_{c \geq 0} g(c, c^t)$

Descent technique

$$\boxed{\min_{c \geq 0} \quad f(c) = \tfrac{1}{2}\|a - Bc\|_2^2}$$

1. Find a function $g(c, \tilde{c})$ that satisfies:

$$\begin{aligned} g(c, c) &= f(c), \quad \text{for all} \quad c, \\ g(c, \tilde{c}) &\geq f(c), \quad \text{for all} \quad c, \tilde{c}. \end{aligned}$$

2. Compute $c^{t+1} = \operatorname{argmin}_{c \geq 0} g(c, c^t)$
3. Then we have descent

Descent technique

$$\boxed{\min_{c \geq 0} \quad f(c) = \tfrac{1}{2}\|a - Bc\|_2^2}$$

1. Find a function $g(c, \tilde{c})$ that satisfies:

$$\begin{aligned} g(c, c) &= f(c), \quad \text{for all} \quad c, \\ g(c, \tilde{c}) &\geq f(c), \quad \text{for all} \quad c, \tilde{c}. \end{aligned}$$

2. Compute $c^{t+1} = \operatorname{argmin}_{c \geq 0} g(c, c^t)$
3. Then we have descent

$$f(c^{t+1})$$

Descent technique

$$\boxed{\min_{c \geq 0} \quad f(c) = \tfrac{1}{2}\|a - Bc\|_2^2}$$

1. Find a function $g(c, \tilde{c})$ that satisfies:

$$\begin{aligned} g(c, c) &= f(c), \quad \text{for all} \quad c, \\ g(c, \tilde{c}) &\geq f(c), \quad \text{for all} \quad c, \tilde{c}. \end{aligned}$$

2. Compute $c^{t+1} = \operatorname{argmin}_{c \geq 0} g(c, c^t)$
3. Then we have descent

$$f(c^{t+1}) \overset{\text{def}}{\leq} g(c^{t+1}, c^t)$$

Descent technique

$$\boxed{\min_{c \geq 0} \quad f(c) = \tfrac{1}{2}\|a - Bc\|_2^2}$$

1. Find a function $g(c, \tilde{c})$ that satisfies:

$$\begin{aligned} g(c,c) &= f(c), \quad \text{for all} \quad c, \\ g(c,\tilde{c}) &\geq f(c), \quad \text{for all} \quad c, \tilde{c}. \end{aligned}$$

2. Compute $c^{t+1} = \operatorname{argmin}_{c \geq 0} g(c, c^t)$
3. Then we have descent

$$f(c^{t+1}) \stackrel{\text{def}}{\leq} g(c^{t+1}, c^t) \stackrel{\text{argmin}}{\leq} g(c^t, c^t)$$

Descent technique

$$\boxed{\min_{c \geq 0} \quad f(c) = \tfrac{1}{2}\|a - Bc\|_2^2}$$

1. Find a function $g(c, \tilde{c})$ that satisfies:

$$\begin{aligned} g(c, c) &= f(c), \quad \text{for all} \quad c, \\ g(c, \tilde{c}) &\geq f(c), \quad \text{for all} \quad c, \tilde{c}. \end{aligned}$$

2. Compute $c^{t+1} = \text{argmin}_{c \geq 0}\, g(c, c^t)$
3. Then we have descent

$$f(c^{t+1}) \stackrel{\text{def}}{\leq} g(c^{t+1}, c^t) \stackrel{\text{argmin}}{\leq} g(c^t, c^t) \stackrel{\text{def}}{=} f(c^t).$$

Constructing g for f

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\sum_i \lambda_i x_i) \le \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \ge 0, \sum_i \lambda_i = 1}$$

Constructing g for f

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\sum_i \lambda_i x_i) \leq \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \geq 0, \sum_i \lambda_i = 1}$$

$$f(c) = \frac{1}{2}\sum_i (a_i - b_i^T c)^2 =$$

Constructing g for f

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\sum_i \lambda_i x_i) \leq \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \geq 0, \sum_i \lambda_i = 1}$$

$$f(c) = \frac{1}{2}\sum_i (a_i - b_i^T c)^2 = \frac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + (b_i^T c)^2$$

Constructing g for f

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\sum_i \lambda_i x_i) \le \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \ge 0, \sum_i \lambda_i = 1}$$

$$\begin{aligned} f(c) &= \tfrac{1}{2}\sum_i (a_i - b_i^T c)^2 = \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + (b_i^T c)^2 \\ &= \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum_i \big(\sum_j b_{ij} c_j\big)^2 \end{aligned}$$

Constructing g for f

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h\left(\sum_i \lambda_i x_i\right) \le \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \ge 0, \sum_i \lambda_i = 1}$$

$$\begin{aligned}
f(c) &= \tfrac{1}{2}\sum_i (a_i - b_i^T c)^2 = \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + (b_i^T c)^2 \\
&= \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum_i \left(\sum_j b_{ij} c_j\right)^2 \\
&= \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c
\end{aligned}$$

Constructing g for f

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\sum_i \lambda_i x_i) \le \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \ge 0, \sum_i \lambda_i = 1}$$

$$\begin{aligned}
f(c) = \tfrac{1}{2}\sum_i (a_i - b_i^T c)^2 &= \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + (b_i^T c)^2 \\
&= \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum_i \Big(\sum_j b_{ij} c_j\Big)^2 \\
&= \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum_i \Big(\sum_j \lambda_{ij} b_{ij} c_j / \lambda_{ij}\Big)^2
\end{aligned}$$

Constructing g for f

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h\left(\sum_i \lambda_i x_i\right) \le \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \ge 0,\ \sum_i \lambda_i = 1}$$

$$\begin{aligned}
f(c) &= \tfrac{1}{2}\sum_i (a_i - b_i^T c)^2 = \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + (b_i^T c)^2 \\
&= \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum_i \Big(\sum_j b_{ij} c_j\Big)^2 \\
&= \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum_i \Big(\sum_j \lambda_{ij} b_{ij} c_j / \lambda_{ij}\Big)^2 \\
&\overset{\text{cvx}}{\le} \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum_{ij} \lambda_{ij} \big(b_{ij} c_j / \lambda_{ij}\big)^2
\end{aligned}$$

Constructing g for f

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\sum_i \lambda_i x_i) \le \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \ge 0, \sum_i \lambda_i = 1}$$

$$\begin{aligned}
f(c) &= \tfrac{1}{2}\sum_i (a_i - b_i^T c)^2 = \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + (b_i^T c)^2 \\
&= \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum_i \Big(\sum_j b_{ij} c_j\Big)^2 \\
&= \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum_i \Big(\sum_j \lambda_{ij} b_{ij} c_j/\lambda_{ij}\Big)^2 \\
&\overset{\text{cvx}}{\le} \tfrac{1}{2}\sum_i a_i^2 - 2a_i b_i^T c + \tfrac{1}{2}\sum_{ij} \lambda_{ij} \big(b_{ij} c_j/\lambda_{ij}\big)^2 \\
&=: g(c, \tilde{c}), \quad \text{where} \quad \lambda_{ij} \quad \text{are convex coeffts}
\end{aligned}$$

Constructing g for f

$$f(c) = \tfrac{1}{2}\|a - Bc\|_2^2$$
$$g(c, \tilde{c}) = \tfrac{1}{2}\|a\|_2^2 - \sum\nolimits_i a_i b_i^T c + \tfrac{1}{2}\sum\nolimits_{ij} \lambda_{ij}\left(b_{ij}c_j/\lambda_{ij}\right)^2.$$

Only remains to *pick* λ_{ij} as functions of $\tilde{c}$

Constructing g for f

$$f(c) = \tfrac{1}{2}\|a - Bc\|_2^2$$
$$g(c, \tilde{c}) = \tfrac{1}{2}\|a\|_2^2 - \sum\nolimits_i a_i b_i^T c + \tfrac{1}{2}\sum\nolimits_{ij} \lambda_{ij}\left(b_{ij}c_j/\lambda_{ij}\right)^2.$$

Only remains to *pick* λ_{ij} as functions of $\tilde{c}$

$$\lambda_{ij} = \frac{b_{ij}\tilde{c}_j}{\sum_k b_{ik}\tilde{c}_k} = \frac{b_{ij}\tilde{c}_j}{b_i^T\tilde{c}}$$

Constructing g for f

$$f(c) = \tfrac{1}{2}\|a - Bc\|_2^2$$
$$g(c, \tilde{c}) = \tfrac{1}{2}\|a\|_2^2 - \sum\nolimits_i a_i b_i^T c + \tfrac{1}{2}\sum\nolimits_{ij} \lambda_{ij}(b_{ij}c_j/\lambda_{ij})^2.$$

Only remains to *pick* λ_{ij} as functions of $\tilde{c}$

$$\lambda_{ij} = \frac{b_{ij}\tilde{c}_j}{\sum_k b_{ik}\tilde{c}_k} = \frac{b_{ij}\tilde{c}_j}{b_i^T\tilde{c}}$$

Exercise: Verify that $g(c, c) = f(c)$;
Exercise: Let $f(c) = \sum_i a_i \log(a_i/(Bc)_i) - a_i + (Bc)_i$. Derive an auxiliary function $g(c, \tilde{c})$ for this $f(c)$.

NMF updates

Key step

$$c^{t+1} = \operatorname*{argmin}_{c \geq 0} g(c, c^t).$$

Exercise: Solve $\partial g(c, c^t)/\partial c_p = 0$ to obtain

$$c_p = c_p^t \frac{[B^T a]_p}{[B^T B c^t]_p}$$

This yields the "multiplicative update" algorithm of Lee/Seung (1999).

MM algorithms

- We exploited convexity of x^2
- Expectation Maximization (EM) algorithm exploits convexity of $-\log x$
- Other choices possible, e.g., by varying λ_{ij}
- Our technique one variant of repertoire of *Majorization-Minimization* (MM) algorithms
- gradient-descent also an MM algorithm
- Related to *d.c. programming*
- MM algorithms subject of a separate lecture!

EM algorithm

Assume $p(x) = \sum_{j=1}^{K} \pi_j p(x; \theta_j)$ is mixture density.

EM algorithm

Assume $p(x) = \sum_{j=1}^{K} \pi_j p(x; \theta_j)$ is mixture density.

$$\ell(\mathcal{X}; \Theta) := \sum_{i=1}^{n} \ln\Big(\sum_{j=1}^{K} \pi_j p(x_i; \theta_j)\Big).$$

EM algorithm

Assume $p(x) = \sum_{j=1}^{K} \pi_j p(x; \theta_j)$ is mixture density.

$$\ell(\mathcal{X}; \Theta) := \sum_{i=1}^{n} \ln\Big(\sum_{j=1}^{K} \pi_j p(x_i; \theta_j)\Big).$$

Use convexity of $-\log t$ to compute lower-bound

$$\ell(\mathcal{X}; \Theta) \geq \sum_{ij} \beta_{ij} \ln\left(\pi_j p(x_i; \theta_j)/\beta_{ij}\right).$$

EM algorithm

Assume $p(x) = \sum_{j=1}^{K} \pi_j p(x; \theta_j)$ is mixture density.

$$\ell(\mathcal{X}; \Theta) := \sum_{i=1}^{n} \ln\Big(\sum_{j=1}^{K} \pi_j p(x_i; \theta_j)\Big).$$

Use convexity of $-\log t$ to compute lower-bound

$$\ell(\mathcal{X}; \Theta) \geq \sum_{ij} \beta_{ij} \ln\left(\pi_j p(x_i; \theta_j)/\beta_{ij}\right).$$

E-Step: Optimize over β_{ij}, to set them to *posterior* probabilities:

$$\beta_{ij} := \frac{\pi_j p(x_i; \theta_j)}{\sum_l \pi_l p(x_i; \theta_l)}.$$

M-Step optimizes the bound over Θ, using above β values

Other Alternating methods

- Alternating Projections
- Alternating Reflections
- (Nonconvex) ADMM (e.g., arXiv:1410.1390)
- (Nonconvex) Douglas-Rachford (e.g., Borwein's webpage!)
- AltMin for global optimization (we saw)
- BCD with more than 2 blocks
- ADMM with more than 2 blocks
- Several others...

Alternating Proximal Method

$$\min \quad L(x, y) := f(x, y) + g(x) + h(y).$$

Assume: ∇f Lipschitz cont. on bounded subsets of $\mathbb{R}^m \times \mathbb{R}^n$
g: lower semicontinuous on $\mathbb{R}^m$
h: lower semicontinuous on $\mathbb{R}^n$.
Example: $f(x, y) = \frac{1}{2}\|x - y\|^2$

Alternating Proximal Method

$$\min \quad L(x, y) := f(x, y) + g(x) + h(y).$$

Assume: ∇f Lipschitz cont. on bounded subsets of $\mathbb{R}^m \times \mathbb{R}^n$
g: lower semicontinuous on $\mathbb{R}^m$
h: lower semicontinuous on $\mathbb{R}^n$.
Example: $f(x, y) = \frac{1}{2}\|x - y\|^2$

Alternating Proximal Method

$$x_{k+1} \in \operatorname{argmin} \left\{ L(x, y_k) + \tfrac{1}{2} c_k \|x - x_k\|^2 \right\}$$
$$y_{k+1} \in \operatorname{argmin} \left\{ L(x_{k+1}, y) + \tfrac{1}{2} c_k' \|y - y_k\|^2 \right\},$$

here c_k, c_k' are suitable sequences of positive scalars.

[arXiv:0801.1780. Attouch, Bolte, Redont, Soubeyran. *Proximal alternating minimization and projection methods for nonconvex problems.*]