Proximal methods:

Consider

\[ f(x) = g(x) + h(x) \]

- convex, diff, domain is \( \mathbb{R}^n \)

\[ \text{argmin}_{x} \left[ w(x) + (\nabla w)(z - x_n) + \frac{1}{2t} \| z - x_n \|^2 \right] \]

\[ = x_n - t \nabla w \]
Strategy:

approx g, ignore h

\[ x_{n+1} = \arg \min_z \left[ g(x_n) + (\nabla g)^T (z - x_n) + \frac{1}{2t} \| z - x_n \|^2 ight] + h(z) \]

\[ = \arg \min_z \left[ \frac{1}{2t} \| z - (x_n - t \nabla g) \|^2 + h(z) \right] \]

[you should check this -- expand terms!]

so \( z \) should be close to g.d. pred

- make \( h(z) \) smaller.
Define:

$$\text{prox}_{h,t}(x) = \arg\min_{z} \frac{1}{2t} \|z - x\|_2^2 + h(z)$$

Proximal gradient descent:

$$x_{n+1} = \text{prox}_{h,t_n}(x_n - t_n \nabla g(x_n))$$

At its most useful when $h$ is such that $\text{prox}_{h,t_n}$ is easy

(Quite common — examples follow)
ISTA:
the lasso problem.

\[
\text{minimize} \quad \frac{1}{2} \left\| y - X \beta \right\|_2^2 + \lambda \left\| \beta \right\|_1
\]

we solved this by splitting and ADMM:

\[
\text{prox}_t (\beta) = \arg \min_{\beta} \frac{1}{2} \left\| \beta - z \right\|_2^2 + \lambda \left\| z \right\|_1
\]

(Which should look familiar)

Notice \( \text{prox}_t (\beta) = \arg \min_{z} \frac{1}{2} \sum_{u} (\beta_u - z_u)^2 + \lambda \sum_{u} \text{abs}(z_u) \)

so it splits over the components of \( z, \beta \).

consider one term

\[
\arg \min_{z_u} \frac{1}{2t} (\beta_u - z_u)^2 + \lambda \text{abs}(z_u)
\]
Some occurs when

$$0 \in \partial_{Z_u}$$

so

$$0 \in -\frac{1}{t}(\beta_u - Z_u) + \lambda \partial_{Z_u} \text{abs}(Z_u).$$

Cases:

\[
\begin{align*}
Z_u & > 0: \quad \text{then} \quad -\frac{1}{t}(\beta_u - Z_u) + \lambda = 0 \\
\Rightarrow & \quad Z_u = \beta_u - t\lambda. \\
& \quad \text{and so} \\
\beta_u & > t\lambda
\end{align*}
\]

\[
\begin{align*}
Z_u & < 0: \\
Z_u & = \beta_u + t\lambda
\end{align*}
\]

\[
\begin{align*}
Z_u & = 0: \\
0 & \in -\frac{1}{t}(\beta_u - Z_u) + \lambda \text{[-1, 1]} \\
& \quad \text{a whole interval!}
\end{align*}
\]

\[
\begin{align*}
0 & \in \beta_u + t\lambda \text{[-1, 1]} \\
\Rightarrow & \quad \lambda \leq \beta_u \leq \lambda t \\
\Rightarrow & \quad \lambda \leq \beta_u \leq \lambda t
\end{align*}
\]
This should all look fairly familiar.

**Notation**

\[ S_{\lambda t}(\beta) \]

Shrinkage operator:

\[
[S_{\lambda t}(\beta)]_u = \begin{cases} 
\beta_u - \lambda t, & \beta_u > \lambda t \\
0, & -\lambda t < \beta_u < \lambda t \\
\beta_u + \lambda t, & \beta_u < -\lambda t.
\end{cases}
\]

So proximal gradient descent is

\[
\beta_{n+1} = S_{\lambda t} [\beta_n + t X^T (y - X\beta)]
\]

(Iterative soft-thresholding algorithm — ISTA)
Convergence:

Assume: \( g \) convex, diff, \( \text{dom}(g) = \mathbb{R}^n \), \( \nabla g \) is Lipschitz, const \( L \).

In convex, \( \text{prox} \) can be evaluated.

Proximal gradient descent with fixed step size \( t \leq \frac{1}{L} \) satisfies

\[
f(x_n) - f^* \leq \frac{\|x_0 - x^*\|_2^2}{2 + n}
\]

Notice: as before, this bounds function values, rather than arguments matches gradient descent...
Recall $\nabla g(\beta) = -X^T(y - X\beta)$, hence proximal gradient update is:

$$\beta^+ = S_{\lambda t}(\beta + tX^T(y - X\beta))$$

Often called the iterative soft-thresholding algorithm (ISTA).\(^1\) Very simple algorithm

Example of proximal gradient (ISTA) vs. subgradient method convergence curves

\(^1\)Beck and Teboulle (2008), “A fast iterative shrinkage-thresholding algorithm for linear inverse problems”
Matrix completion:

we have $Y \in \mathbb{R}^{(m \times n)}$

But observe only some entries.
we want to fill in missing entries
- general case isn't possible
- but assume $Y$ is known to have
  small nuclear norm

(ky-Fan norm)

$\| B \|_{KF} = \sum \sigma_i(B)$

where $\sigma_i$ are singular values
of $B$ AND $\sigma_i > 0$

General idea: this norm favors
matrices with low rank
- This is a generalization of the case where \( Y = AB \) and entries are missing.

(in this case, rank of \( Y \) is known)

- Classic example of \( Y = AB \):
  - affine structure from motion
  - with missing observations

General case:
- recommender systems
- smoothing word counts
Reminder: imagine we know $y_{ij}$ for $i,j \in K$

then

$$\min_{a_i,k,b_{ij}} \sum_{i,j \in K} \left[ y_{ij} - \frac{1}{k} a_i e^{-b_{ij}} \right]^2$$

(least squares solve for $A, B$, changing only for known entries of $Y_{ij}$)

is straightforward.

Iterate

- fix $A$, solve for $B$ (linear sys)
- fix $B$, solve for $A$

works-ish, but has annoying features.

- can diverge (uncommon)
- can be slow (common)
- you have to know rank.
Recommender Systems

Video (Say?)

\[ i \to j \]

how much user \( i \) liked video \( j \).

\[ \text{User} \]

This matrix should have low rank.

- Many pairs of users will have about the same tastes.
  \[ \rightarrow \text{many pairs of rows will be similar} \]
- Two "similar" [horror, history, etc.] videos will be liked by about the same set of people.
  \[ \rightarrow \text{many pairs of cols will be similar} \]
AND:

- many cells were not observed
- knowing values of those cells is worthwhile.
Another example

PCA with data deleted "at random"

\[ X = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix} \]

test \[ \# \rightarrow \]

Have \[ \text{obs} \# \rightarrow \]

Assume \[ \mathbf{1}^T X = 0 \]

(i.e. data is zero mean)

- It is easy to get \underline{means} \underline{w}
  
- Missing observations:
  
- Covar presents challenges.
  
But covar presents challenges.

(possible, but nuisance)
Alternative:
- PCA for $X$ would yield $Y$ and $T$.

\[ X = \begin{bmatrix} \vdots & \vdots \end{bmatrix} \begin{bmatrix} \vdots & \vdots \end{bmatrix} \begin{bmatrix} \vdots & \vdots \end{bmatrix} \begin{bmatrix} \vdots \end{bmatrix} \]

- We know for most datasets, rank is small, but we don't know what it is.
- Possibility: choose rank.
- Factor $X$ w/ SVD (missing data? as above)

Alternative:
- Complete $X$
- Then obtain PCA w/ SVD.
Smoothing word counts:

$\text{word}_i$:

- How many times \text{word}_j \text{ appears in } \text{doc}_i$

Document

- In principle, we could know all entries of this Document-Word matrix.
- But 0-counts are somewhat unreliable.
- Many words have similar meanings:
  - "car", "motorcar", "automobile"
If "car" appears, there should likely be a small count for "motorcar" etc.

This means that the matrix should have low rank, too.

We are now interested in

$$\min_B \left[ \text{difference between observed } y \text{ and } \text{CSP } B \right]^2 + \lambda \| B \|_{KF}.$$ 

Write

$$P_{2}(B) = \begin{cases} B_{ij} & \text{if } (i,j) \text{ was observed} \\ 0 & \text{otherwise} \end{cases}$$
Then:

$$\min_B \frac{1}{2} \left\| P_{\Omega} (Y) - P_{\Omega} (B) \right\|_F^2 + \lambda \| B \|_{KF}$$

(notice \( P_{\Omega} (B) \) might return every value,)

if all of \( Y \) is observed

$$\| M \|_F^2 = \sum_{ij} M_{ij}^2 = Tr (M^T M) \leftarrow \text{Frobenius norm}.$$ 

$$f(B) = \frac{1}{2} \left\| P_{\Omega} (Y) - P_{\Omega} (B) \right\|_F^2 + \lambda \| B \|_{KF}$$ 

\[ \uparrow \]

$$g(B)$$

\[ \uparrow \]

$$h(B)$$

for proximal, need: gradient, prox

$$\text{gradient } \nabla_B (g) = -(P_{\Omega} (Y) - P_{\Omega} (B))$$

Q: What happened to \( \frac{\partial P_{\Omega}}{\partial B} \)?
\[ \text{prox}_{\ell_0} (B) = \arg \min_Z \left[ \frac{1}{2\lambda} \| B - Z \|_F^2 + \lambda \| Z \|_{K_F} \right] \]

Claim:

\[ \text{prox}_{\ell_0} (B) = S_{\lambda t} (B) \]

(matrix soft thresholding)

\[ S_{\lambda t} (B) = U \Sigma_{\lambda t} V^T \]

where

\[ B = U \Sigma V^T \] (SVD)

and

\[ \Sigma_{\lambda t} \] is diagonal,

\[ \left[ \Sigma_{\lambda t} \right]_{\mu \nu} = \max \left( \Sigma_{\mu \nu} - \lambda t \cdot 0 \right) \]

(proof in Tibshirani notes on web page).
\[ B_{n+1} = S_{\lambda t} \left( B_n + t \left( P_{\Omega}^+ (Y) - P_{\Omega}^- (B) \right) \right). \]

can choose \( t = 1 \), get

\[ B_{n+1} = S_{\lambda t} \left( P_{\Omega}^+ (Y) + P_{\Omega}^- (B) \right) \]

(where \( P_{\Omega}^+ (B) \) projects onto \underline{unobserved set})

This is \underline{soft-update} (ref in notes).
Now assume we want

$$\min_{x \in C} g(x)$$

where $g(x)$ is $\text{diff}$, convex and defined on $\mathbb{R}^n$

$C$ is convex

Write:

$$\min_{x \in \mathbb{R}^n} g(x) + I_C(x)$$

where

$$I_C(x) = \begin{cases} 0 & \text{if } x \in C \\ \infty & \text{otherwise} \end{cases}$$

Now:

$$\text{prox}_t(x) = \arg\min_z \frac{1}{2t} \| (x - z)^2_z + I_C(x)$$

$$= \arg\min_{z \in \mathbb{R}^n} \| x - z \|_2^2$$

$$= \text{proj}_{\mathbb{R}^n}(x)$$
so that

$$\text{prox}_c(x) = \begin{cases} \frac{P_c(x)}{\|P_c(x)\|} \cdot \text{project } x \text{ onto } c \\ \text{closest point in } C \text{ to } x \end{cases}$$

So

$$x_{n+1} = \text{proj}_c \left( x_n - t \nabla g(x_n) \right)$$

**Projected gradient descent**

useful if you can compute $P_c$

- box, sphere, etc.