Final exam review

CS 446

* Hawk weighting
* Handwritten cheatsheet front & back US letter
* Wasserstein Distance
Selected lecture slides
Hoeffding’s inequality

**Theorem (Hoeffding’s inequality).** Given IID $Z_i \in [a, b]$,

$$
\Pr \left[ \frac{1}{n} \sum_{i} Z_i - \mathbb{E}Z_1 \geq \epsilon \right] \leq \exp \left( \frac{-2n\epsilon^2}{(b-a)^2} \right).
$$

Alternatively, with probability at least $1 - \delta$,

$$
\frac{1}{n} \sum_{i=1}^{n} Z_i \leq \mathbb{E}Z_1 + (b - a)\sqrt{\frac{\ln(1/\delta)}{2n}}.
$$

**Remarks.**

- Can flip inequality by replacing $Z_i$ with $-Z_i$.
- Using the second (“inverted”) form: with probability at least $1 - \delta$,

  $$
  R(h) \leq \hat{R}(h) + \sqrt{\ln(1/\delta)/2n}.
  $$

- Alternatively: setting $\delta = 10^{-k}$, with probability at least $99\%$, we have

  $$
  R(h) \leq \hat{R}(h) + \sqrt{k\ln(10)/2n}.
  $$

  To add more bits of confidence, we must increase sample size $n$ linearly.

- Hoeffding captures a “concentration of measure” phenomenon: probability mass concentrates within $[-1/\sqrt{n}, +1/\sqrt{n}]$. 


Hoeffding’s inequality

**Theorem (Hoeffding’s inequality).** Given IID $Z_i \in [a, b]$, 

$$
\Pr \left[ \frac{1}{n} \sum_{i} Z_i - E Z_1 \geq \epsilon \right] \leq \exp \left( \frac{-2n\epsilon^2}{(b-a)^2} \right).
$$

Alternatively, with probability at least $1 - \delta$,

$$
\frac{1}{n} \sum_{i=1}^{n} Z_i \leq E Z_1 + (b - a) \sqrt{\frac{\ln(1/\delta)}{2n}}.
$$

**Remarks.**

- Can flip inequality by replacing $Z_i$ with $-Z_i$.
- Using the second ("inverted") form: with probability at least $1 - \delta$,

$$
\mathcal{R}(h) \leq \hat{\mathcal{R}}(h) + \sqrt{\frac{\ln(1/\delta)}{2n}}.
$$

- Alternatively: setting $\delta = 10^{-k}$, with probability at least $99.99 \cdots 9\%$ ($k$ 9s), we have $\mathcal{R}(h) \leq \hat{\mathcal{R}}(h) + \sqrt{k \ln(10)/(2n)}$: to add more bits of confidence, we must increase sample size $n$ linearly.
- Hoeffding captures a “concentration of measure” phenomenon: probability mass concentrates within $[-1/\sqrt{n}, +1/\sqrt{n}]$. 

Theorem. Let predictors \((h_1, \ldots, h_k)\) be given. With probability \(\geq 1 - \delta\) over an IID draw \(((X_i, Y_i))_{i=1}^n\),

\[
R(h_j) \leq \hat{R}(h_j) + \sqrt{\frac{\ln k + \ln(1/\delta)}{2n}} \quad \forall j.
\]

Remarks.
- If we choose \((h_1, \ldots, h_k)\) before seeing \(((X_i, Y_i))_{i=1}^n\), we can use this bound.
- **Example:** train \(k\) classifiers, pick the best on validation set!
- This approach “produce bound for all possible algo outputs” may seem sloppy, but it’s the best we have!
- Letting \(F = (h_1, \ldots, h_k)\) denote our set of predictors, the bound is: with probability \(\geq 1 - \delta\), every \(f \in F\) satisfies

\[
R(f) \leq \hat{R}(f) + \sqrt{\frac{\ln |F| + \ln 1/\delta}{2n}}.
\]

In the next sections, we’ll handle \(|F| = \infty\) by replacing \(\ln |F|\) with complexity(\(F\)), whose meaning will vary.
VC dimension overview

**Theorem.** With probability at least $1 - \delta$, every $f \in \mathcal{F}$ satisfies

$$\mathcal{R}(f) \leq \hat{\mathcal{R}}(f) + \tilde{O}\left(\sqrt{\frac{\text{VC}(\mathcal{F})}{n} \ln(1/\delta)}\right),$$

where $\text{VC}(\mathcal{F})$, the **Vapnik-Chervonenkis dimension** of $\mathcal{F}$ is the largest number of points where $\mathcal{F}$ can realize all labelings:

$$\text{VC}(\mathcal{F}) := \sup \left\{ n \in \mathbb{Z} : \exists (x_1, \ldots, x_n), \right\}$$
Theorem. With probability at least $1 - \delta$, every $f \in \mathcal{F}$ satisfies

$$R(f) \leq \hat{R}(f) + \tilde{O}\left(\sqrt{\frac{VC(\mathcal{F}) + \ln(1/\delta)}{n}}\right),$$

where $VC(\mathcal{F})$, the **Vapnik-Chervonenkis dimension** of $\mathcal{F}$ is the largest number of points where $\mathcal{F}$ can realize all labelings:

$$VC(\mathcal{F}) := \sup \left\{ n \in \mathbb{Z} : \exists (x_1, \ldots, x_n), \forall (y_1, \ldots, y_n), \right\}$$
Theorem. With probability at least $1 - \delta$, every $f \in \mathcal{F}$ satisfies

$$\mathcal{R}(f) \leq \hat{\mathcal{R}}(f) + \tilde{O}\left(\sqrt{\frac{\text{VC}(\mathcal{F}) + \ln(1/\delta)}{n}}\right),$$

where $\text{VC}(\mathcal{F})$, the Vapnik-Chervonenkis dimension of $\mathcal{F}$ is the largest number of points where $\mathcal{F}$ can realize all labelings:

$$\text{VC}(\mathcal{F}) := \sup \left\{ n \in \mathbb{Z} : \exists (x_1, \ldots, x_n), \forall (y_1, \ldots, y_n), \exists f \in \mathcal{F}, \hat{\mathcal{R}}_{0/1}(f) = 0 \right\}.$$
Theorem. With probability at least $1 - \delta$, every $f \in \mathcal{F}$ satisfies

$$R(f) \leq \hat{R}(f) + \tilde{O}\left(\sqrt{\frac{\text{VC}(\mathcal{F}) + \ln(1/\delta)}{n}}\right),$$

where $\text{VC}(\mathcal{F})$, the Vapnik-Chervonenkis dimension of $\mathcal{F}$ is the largest number of points where $\mathcal{F}$ can realize all labelings:

$$\text{VC}(\mathcal{F}) := \sup \left\{ n \in \mathbb{Z} : \exists (x_1, \ldots, x_n), \forall (y_1, \ldots, y_n), \exists f \in \mathcal{F}, \hat{R}_{0/1}(f) = 0 \right\}.$$

Remarks.

- $|\mathcal{F}|$ can be infinite!
- Definition only requires some set of points we can label in every way; this set is unrelated to the IID sample for the bound.
- Say that $\mathcal{F}$ shatters $(x_1, \ldots, x_n)$ when it can realize all labelings.
Rademacher complexity

**Definition.** Given examples \((x_1, \ldots, x_n)\) and functions \(\mathcal{F}\),

\[
\text{Rad}(\mathcal{F}) = E_{\epsilon} \max_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(x_i),
\]

where \((\epsilon_1, \ldots, \epsilon_n)\) are IID Rademacher rv

\((\Pr[\epsilon_i = 1] = \Pr[\epsilon_i = -1] = \frac{1}{2})\).
Rademacher complexity

**Definition.** Given examples \((x_1, \ldots, x_n)\) and functions \(\mathcal{F}\),

\[
\text{Rad}(\mathcal{F}) = \mathbb{E}_\varepsilon \max_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \varepsilon_i f(x_i),
\]

where \((\varepsilon_1, \ldots, \varepsilon_n)\) are IID Rademacher rv
\((\Pr[\varepsilon_i = 1] = \Pr[\varepsilon_i = -1] = \frac{1}{2})\).

**Remarks.**

- We should make \(\mathcal{F}\) as tight as possible; e.g., for SVM, we’ll incorporate \(C\).
Rademacher complexity

**Definition.** Given examples \((x_1, \ldots, x_n)\) and functions \(F\),

\[
\text{Rad}(F) = \mathbb{E}_{\epsilon} \max_{f \in F} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(x_i),
\]

where \((\epsilon_1, \ldots, \epsilon_n)\) are IID Rademacher rv

\((\Pr[\epsilon_i = 1] = \Pr[\epsilon_i = -1] = \frac{1}{2})\).

**Remarks.**

- Interpretation: ability of \(F\) to fit random signs.
- We should make \(F\) as tight as possible; e.g., for SVM, we’ll incorporate \(C\).
Rademacher complexity

**Definition.** Given examples \((x_1, \ldots, x_n)\) and functions \(\mathcal{F}\),

\[
\text{Rad}(\mathcal{F}) = \mathbb{E}_\varepsilon \max_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \varepsilon_i f(x_i),
\]

where \((\varepsilon_1, \ldots, \varepsilon_n)\) are IID Rademacher rv

\((\Pr[\varepsilon_i = 1] = \Pr[\varepsilon_i = -1] = \frac{1}{2})\).

**Remarks.**

- Interpretation: ability of \(\mathcal{F}\) to fit random signs.

- We whould make \(\mathcal{F}\) as tight as possible;
  e.g., for SVM, we’ll incorporate \(C\).

- Compared to VC: depends on \((x_1, \ldots, x_n)\), doesn’t require classification, is sensitive to scale of \(f\).
**Rademacher complexity**

**Definition.** Given examples \((x_1, \ldots, x_n)\) and functions \(\mathcal{F}\),

\[
\text{Rad}(\mathcal{F}) = \mathbb{E}_{\epsilon} \max_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(x_i),
\]

where \((\epsilon_1, \ldots, \epsilon_n)\) are IID Rademacher rv
\((\Pr[\epsilon_i = 1] = \Pr[\epsilon_i = -1] = \frac{1}{2})\).

**Remarks.**

- Interpretation: ability of \(\mathcal{F}\) to fit random signs.
- We should make \(\mathcal{F}\) as tight as possible;
  e.g., for SVM, we’ll incorporate \(C\).
- Compared to VC: depends on \((x_1, \ldots, x_n)\), doesn’t require classification, is sensitive to scale of \(f\).
- The general form (not presented here) can handle including labels, multiclass, amongst other things.
Theorem (simplified Rademacher generalization).

Let predictors $F$ and a distribution on $(X, Y)$ be given. Suppose for (almost) any $(x, y)$, loss $\ell$ satisfies:

- There exists $\rho \geq 0$ so that for any $f, g \in F$, 
  $$|\ell(f(x), y) - \ell(g(x), y)| \leq \rho |f(x) - g(x)|$$
  ("$\rho$-Lipschitz").

- There exists $[a, b]$ so that $\ell(f(x), y) \in [a, b]$ for any $f \in F$.

With probability $\geq 1 - \delta$, every $f \in F$ satisfies

$$R_\ell(f) \leq \hat{R}_\ell(f) + 2\rho \text{Rad}(F) + 3(b - a)\sqrt{\ln(2/\delta)/n}.$$
Theorem (simplified Rademacher generalization).
Let predictors $\mathcal{F}$ and a distribution on $(\mathbf{X}, Y)$ be given.
Generalization via Rademacher complexity

**Theorem** (simplified Rademacher generalization).
Let predictors $\mathcal{F}$ and a distribution on $(X, Y)$ be given. Suppose for (almost) any $(x, y)$, loss $\ell$ satisfies:

- There exists $\rho \geq 0$ so that for any $f, g \in \mathcal{F}$,
  \[ |\ell(f(x), y) - \ell(g(x), y)| \leq \rho |f(x) - g(x)| \]  
  ("$\rho$-Lipschitz").

- There exists $[a, b]$ so that $\ell(f(x), y) \in [a, b]$ for any $f \in \mathcal{F}$.

With probability $\geq 1 - \delta$, every $f \in \mathcal{F}$ satisfies

\[ R^\ell(f) \leq \hat{R}^\ell(f) + 2\rho \text{Rad}(\mathcal{F}) + 3(b - a)\sqrt{\ln(2/\delta) / 2n} . \]

Remarks.
- Can get a bound in terms of $1/\lambda_n$ for linear SVM and ridge regression. (Homework problems?)
- Kernel SVM okay as well.
**Theorem** (simplified Rademacher generalization).
Let predictors $\mathcal{F}$ and a distribution on $(X, Y)$ be given. Suppose for (almost) any $(x, y)$, loss $\ell$ satisfies:

- There exists $\rho \geq 0$ so that for any $f, g \in \mathcal{F}$,
  $$|\ell(f(x), y) - \ell(g(x), y)| \leq \rho |f(x) - g(x)| \text{ ("$\rho$-Lipschitz")}. $$

**Remarks.**

- Can get a bound in terms of $1/(\lambda n)$ for linear SVM and ridge regression. (Homework problems?)
- Kernel SVM okay as well.
Generalization via Rademacher complexity

**Theorem** (simplified Rademacher generalization).
Let predictors $\mathcal{F}$ and a distribution on $(X, Y)$ be given. Suppose for (almost) any $(x, y)$, loss $\ell$ satisfies:

- There exists $\rho \geq 0$ so that for any $f, g \in \mathcal{F}$,
  \[ |\ell(f(x), y) - \ell(g(x), y)| \leq \rho |f(x) - g(x)| \]  ("$\rho$-Lipschitz").

- There exists $[a, b]$ so that $\ell(f(x), y) \in [a, b]$ for any $f \in \mathcal{F}$.
Theorem (simplified Rademacher generalization).
Let predictors $\mathcal{F}$ and a distribution on $(X, Y)$ be given. Suppose for (almost) any $(x, y)$, loss $\ell$ satisfies:

- There exists $\rho \geq 0$ so that for any $f, g \in \mathcal{F}$,
  \[ |\ell(f(x), y) - \ell(g(x), y)| \leq \rho |f(x) - g(x)| \]  
  ("$\rho$-Lipschitz").

- There exists $[a, b]$ so that $\ell(f(x), y) \in [a, b]$ for any $f \in \mathcal{F}$.

With probability $\geq 1 - \delta$, every $f \in \mathcal{F}$ satisfies

\[
\mathcal{R}_\ell(f) \leq \hat{\mathcal{R}}_\ell(f) + 2\rho \text{Rad}(\mathcal{F}) + 3(b - a)\sqrt{\frac{\ln(2/\delta)}{2n}}.
\]
Generalization via Rademacher complexity

**Theorem** (simplified Rademacher generalization).
Let predictors $F$ and a distribution on $(X, Y)$ be given. Suppose for (almost) any $(x, y)$, loss $\ell$ satisfies:

- There exists $\rho \geq 0$ so that for any $f, g \in F$,
  $$|\ell(f(x), y) - \ell(g(x), y)| \leq \rho |f(x) - g(x)|$$
  ("$\rho$-Lipschitz").

- There exists $[a, b]$ so that $\ell(f(x), y) \in [a, b]$ for any $f \in F$.

With probability $\geq 1 - \delta$, every $f \in F$ satisfies

$$R_\ell(f) \leq \hat{R}_\ell(f) + 2\rho \text{Rad}(F) + 3(b - a)\sqrt{\frac{\ln(2/\delta)}{2n}}.$$

**Remarks.**

- Can get a bound in terms of $1/(\lambda n)$ for linear SVM and ridge regression. (Homework problems?)

- Kernel SVM okay as well.
Rademacher complexity examples

**Definition.** Given examples \((x_1, \ldots, x_n)\) and functions \(\mathcal{F}\),

\[
\text{Rad}(\mathcal{F}) = \mathbb{E}_{\mathbf{\epsilon}} \max_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(x_i),
\]

where \((\epsilon_1, \ldots, \epsilon_n)\) are IID Rademacher rv

\((\Pr[\epsilon_i = 1] = \Pr[\epsilon_i = -1] = \frac{1}{2})\).

**Examples.**
Rademacher complexity examples

**Definition.** Given examples \((x_1, \ldots, x_n)\) and functions \(F\),

\[
\text{Rad}(F) = \mathbb{E}_{\epsilon} \max_{f \in F} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(x_i),
\]

where \((\epsilon_1, \ldots, \epsilon_n)\) are IID Rademacher rv
(\(\Pr[\epsilon_i = 1] = \Pr[\epsilon_i = -1] = \frac{1}{2}\)).

**Examples.**

▶ If \(\|x\| \leq R\), then \(\text{Rad}(\{x \mapsto x^T \mathbf{w} : \|\mathbf{w}\| \leq W\}) \leq \frac{RW}{\sqrt{n}}\).

For SVM, we can set \(W = \sqrt{2/\lambda}\).
**Definition.** Given examples \((x_1, \ldots, x_n)\) and functions \(\mathcal{F}\),

\[
\text{Rad}(\mathcal{F}) = \mathbb{E}_{\epsilon} \max_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(x_i),
\]

where \((\epsilon_1, \ldots, \epsilon_n)\) are IID Rademacher rv

\((\Pr[\epsilon_i = 1] = \Pr[\epsilon_i = -1] = \frac{1}{2})\).

**Examples.**

- If \(\|x\| \leq R\), then \(\text{Rad}(\{x \mapsto x^T w : \|w\| \leq W\}) \leq \frac{RW}{\sqrt{n}}\).
  
  For SVM, we can set \(W = \sqrt{2/\lambda}\).

- For deep networks, we have \(\text{Rad}(\mathcal{F}) \leq \text{Lipschitz} \cdot \sqrt{\text{Junk}/n}\); still very loose.
Unsupervised learning

Now we only receive \((x_i)_{i=1}^n\), and the goal is...?
Unsupervised learning

Now we only receive \((x_i)_{i=1}^{n}\), and the goal is...?

- Encoding data in some compact representation (and decoding this).
- Data analysis; recovering “hidden structure” in data (e.g., recovering cliques or clusters).
- Features for supervised learning.
- ...?
Unsupervised learning

Now we only receive \((x_i)_{i=1}^n\), and the goal is...?

- Encoding data in some compact representation (and decoding this).
- Data analysis; recovering “hidden structure” in data (e.g., recovering cliques or clusters).
- Features for supervised learning.
- ...?

The task is less clear-cut. In 2019 we still have people trying to formalize it!
1. SV triples: \((s, u, v)\) satisfies \(Mv = s\) and \(M^T u = sv\).

2. Thin decomposition SVD: 

\[ M = \sum_{i=1}^{r} s_i u_i v_i^T. \]

3. Full factorization SVD: 

\[ M = USV^T. \]

4. "Operational" view of SVD: for \(M \in \mathbb{R}^{n \times d}\),

\[
\begin{bmatrix}
\uparrow & \uparrow \\
v_1 & \cdots & v_r & v_{r+1} & \cdots & v_d \\
\downarrow & \downarrow
\end{bmatrix}
\cdot
\begin{bmatrix}
s_1 & 0 & \cdots & 0 & 0 & s_r & 0 & \cdots \end{bmatrix}
\cdot
\begin{bmatrix}
\uparrow & \uparrow \\
u_1 & \cdots & u_r & u_{r+1} & \cdots & u_n \\
\downarrow & \downarrow
\end{bmatrix}^T.
\]

First part of \(U, V\) span the col / row space (respectively), second part the left / right nullspaces (respectively).

New: let \((U_k, S_k, V_k)\) denote the truncated SVD with \(U_k \in \mathbb{R}^{d \times k}\) (first \(k\) columns of \(U\)), similarly for the others.
1. SV triples: \((s, u, v)\) satisfies \(Mv = su\), and \(M^Tu = sv\).
1. SV triples: \((s, u, v)\) satisfies \(Mv = su\), and \(M^T u = sv\).

2. Thin decomposition SVD: \(M = \sum_{i=1}^{r} s_i u_i v_i^T\).
1. SV triples: \((s, u, v)\) satisfies \(Mv = su\), and \(M^Tu = sv\).

2. Thin decomposition SVD: \(M = \sum_{i=1}^{r} s_i u_i v_i^T\).

3. Full factorization SVD: \(M = USV^T\).
1. SV triples: \((s, u, v)\) satisfies \(Mv = su\), and \(M^T u = sv\).

2. Thin decomposition SVD: \(M = \sum_{i=1}^{r} s_i u_i v_i^T\).

3. Full factorization SVD: \(M = USV^T\).

4. “Operational” view of SVD: for \(M \in \mathbb{R}^{n \times d}\),

\[
\begin{bmatrix}
    u_1 & \cdots & u_r & u_{r+1} & \cdots & u_n
\end{bmatrix}
\begin{bmatrix}
    \begin{array}{cc}
        s_1 & 0 \\
        \vdots & \ddots \\
        0 & s_r
    \end{array}
\end{bmatrix}
\begin{bmatrix}
    v_1 & \cdots & v_r & v_{r+1} & \cdots & v_d
\end{bmatrix}^T.
\]
1. SV triples: $(s, u, v)$ satisfies $Mv = su$, and $M^Tu = sv$.
2. Thin decomposition SVD: $M = \sum_{i=1}^{r} s_i u_i v_i^T$.
3. Full factorization SVD: $M = USV^T$.
4. “Operational” view of SVD: for $M \in \mathbb{R}^{n \times d},$

$$\begin{bmatrix}
\uparrow & \cdots & \uparrow \\
\downarrow & \cdots & \downarrow \\
\begin{bmatrix} u_1 & \cdots & u_r & u_{r+1} & \cdots & u_n \end{bmatrix}
\end{bmatrix} \cdot \begin{bmatrix}
s_1 & 0 \\
\cdots & \cdots & 0 \\
0 & \cdots & s_r \\
0 & \cdots & 0
\end{bmatrix} \cdot \begin{bmatrix}
\uparrow & \cdots & \uparrow \\
\downarrow & \cdots & \downarrow \\
\begin{bmatrix} v_1 & \cdots & v_r & v_{r+1} & \cdots & v_d \end{bmatrix}
\end{bmatrix}^T.
$$

First part of $U, V$ span the col / row space (respectively), second part the left / right nullspaces (respectively).
1. SV triples: \((s, u, v)\) satisfies \(Mv = su\), and \(M^T u = sv\).

2. Thin decomposition SVD: \(M = \sum_{i=1}^{r} s_i u_i v_i^T\).

3. Full factorization SVD: \(M = USV^T\).

4. “Operational” view of SVD: for \(M \in \mathbb{R}^{n \times d}\),

\[
\begin{bmatrix}
\uparrow & \cdots & \uparrow \\
\downarrow & & \downarrow \\
\vdots & & \vdots \\
\uparrow & & \uparrow \\
\end{bmatrix}
\begin{bmatrix}
\uparrow & \cdots & \uparrow \\
\downarrow & & \downarrow \\
\vdots & & \vdots \\
\uparrow & & \uparrow \\
\end{bmatrix},
\begin{bmatrix}
s_1 & \cdots & 0 \\
0 & \cdots & 0 \\
0 & s_r & 0 \\
\end{bmatrix},
\begin{bmatrix}
\uparrow & \cdots & \uparrow \\
\downarrow & & \downarrow \\
\vdots & & \vdots \\
\uparrow & & \uparrow \\
\end{bmatrix}
\begin{bmatrix}
v_1 & \cdots & v_r & v_{r+1} & \cdots & v_d \\
\end{bmatrix}^T
\]

First part of \(U, V\) span the col / row space (respectively), second part the left / right nullspaces (respectively).

**New:** let \((U_k, S_k, V_k)\) denote the truncated SVD with \(U_k \in \mathbb{R}^{d \times k}\) (first \(k\) columns of \(U\)), similarly for the others.
PCA properties

**Theorem.** Let $X \in \mathbb{R}^{n \times d}$ with SVD $X = USV^T$ and integer $k \leq r$ be given.

$$
\min_{D \in \mathbb{R}^{k \times d}, \ E \in \mathbb{R}^{d \times k}} \| X - XED \|_F^2 = \min_{D \in \mathbb{R}^{d \times k}} \| X - XDD^T \|_F^2
$$

$$
= \| X - XV_k V_k^T \|_F^2 = \sum_{i=k+1}^{r} s_i^2.
$$

Additionally,

$$
\min_{D \in \mathbb{R}^{d \times k}} \| X - XDD^T \|_F^2 = \| X \|_F^2 - \max_{D \in \mathbb{R}^{d \times k}} \| XD \|_F^2
$$

$$
= \| X \|_F^2 - \| XV_k \|_F^2 = \| X \|_F^2 - \sum_{i=1}^{k} s_i^2.
$$
\[ \| X - XV_{kn}V_{kn}^T \|_F^2 \]

\[ = \text{tr} \left( (X - XV_{kn}V_{kn}^T)^T (X - XV_{kn}V_{kn}^T) \right) \]

\[ = \text{tr} (X^T X) - \text{tr} (X^T XV_{kn}V_{kn}^T) \]

\[ - \text{tr} \left( (XV_{kn}V_{kn}^T)^T X \right) + \text{tr} \left( XV_{kn}V_{kn}^T (XV_{kn}V_{kn}^T)^T \right) \]

\[ = \| X \|_F^2 \]

\[ \| XV_{kn}V_{kn}^T \|_F \]
\[ \text{tr} \left( \left( X V_k V_h^T \right)^T X \right) \]

\[ = \text{tr} \left( \left( V_h V_k^T X^T \right) X \right) \]

\[ = \text{tr} \left( X^T X V_k V_h^T \right) \]

\[ (AB)^T = B^T A^T \]
**Theorem.** Let $X \in \mathbb{R}^{n \times d}$ with SVD $X = USV^T$ and integer $k \leq r$ be given.

$$\min_{D \in \mathbb{R}^{k \times d}, E \in \mathbb{R}^{d \times k}} \|X - XED\|_F^2 = \min_{D \in \mathbb{R}^{d \times k}, D^TD = I} \|X - XDD^T\|_F^2$$

$$= \|X - XV_kV_k^T\|_F^2 = \sum_{i=k+1}^{r} s_i^2.$$ 

Additionally,

$$\min_{D \in \mathbb{R}^{d \times k}, D^TD = I} \|X - XDD^T\|_F^2 = \|X\|_F^2 - \max_{D \in \mathbb{R}^{d \times k}, D^TD = I} \|XD\|_F^2$$

$$= \|X\|_F^2 - \|XV_k\|_F^2 = \|X\|_F^2 - \sum_{i=1}^{k} s_i^2.$$ 

**Remark 1.** SVD is unique, but $\sum_{i=1}^{r} s_i^2$ unique.
PCA properties

**Theorem.** Let $X \in \mathbb{R}^{n \times d}$ with SVD $X = USV^T$ and integer $k \leq r$ be given.

$$\min_{D \in \mathbb{R}^{k \times d}} \|X - XED\|_F^2 = \min_{D \in \mathbb{R}^{d \times k}, D^T D = I} \|X - XDD^T\|_F^2$$

$$= \|X - XV_k V_k^T\|_F^2 = \sum_{i=k+1}^{r} s_i^2.$$

Additionally,

$$\min_{D \in \mathbb{R}^{d \times k}, D^T D = I} \|X - XDD^T\|_F^2 = \|X\|_F^2 - \max_{D \in \mathbb{R}^{d \times k}, D^T D = I} \|XD\|_F^2$$

$$= \|X\|_F^2 - \|XV_k\|_F^2 = \|X\|_F^2 - \sum_{i=1}^{k} s_i^2.$$

**Remark 1.** SVD is unique, but $\sum_{i=1}^{r} s_i^2$ unique.

**Remark 2.** As written, this is not a convex optimization problem!
PCA properties

**Theorem.** Let \( X \in \mathbb{R}^{n \times d} \) with SVD \( X = U S V^T \) and integer \( k \leq r \) be given.

\[
\min_{D \in \mathbb{R}^{k \times d}, E \in \mathbb{R}^{d \times k}} \| X - XED \|_F^2 = \min_{D \in \mathbb{R}^{d \times k}} \| X - XDD^T \|_F^2 = \| X - XV_k V_k^T \|_F^2 = \sum_{i=k+1}^{r} s_i^2.
\]

Additionally,

\[
\min_{D \in \mathbb{R}^{d \times k}, D^T D = I} \| X - XDD^T \|_F^2 = \| X \|_F^2 - \max_{D \in \mathbb{R}^{d \times k}, D^T D = I} \| XD \|_F^2 = \| X \|_F^2 - \sum_{i=1}^{k} s_i^2.
\]

**Remark 1.** SVD is unique, but \( \sum_{i=1}^{r} s_i^2 \) unique.

**Remark 2.** As written, this is not a convex optimization problem!

**Remark 3.** The second form is interesting...
Some treatments replace $X$ with $X - \mathbf{1} \mu^T$, with mean $\mu = \frac{1}{n} \sum_{i=1}^{i=n} x_i$. 
Centered PCA

Some treatments replace $X$ with $X - 1\mu^T$, with mean

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i.$$ 

$$\frac{1}{n} X^T X \in \mathbb{R}^{d \times d}$$

is data covariance;
Centered PCA

Some treatments replace $X$ with $X - 1\mu^T$, with mean $\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$.

$$\frac{1}{n}X^T X \in \mathbb{R}^{d \times d}$$ is data covariance;

$$\frac{1}{n} (XD)^T (XD)$$ is data covariance after projection;
**Centered PCA**

Some treatments replace $X$ with $X - 1\mu^T$, with mean $\mu = \frac{1}{n} \sum_{i=1} x_i$.

$$\frac{1}{n} X^T X \in \mathbb{R}^{d \times d}$$ is data covariance;

$$\frac{1}{n} (XD)^T (XD)$$ is data covariance after projection;

lastly

$$\frac{1}{n} \| XD \|^2_F = \frac{1}{n} \text{tr} \left( (XD)^T XD \right) = \frac{1}{n} \sum_{i=1}^k (XD e_i)^T (XD e_i),$$

therefore PCA is maximizing the resulting per-coordinate variances!
Lloyd’s method revisited

1. Choose initial clusters \((S_1, \ldots, S_k)\).

2. Repeat until convergence:
   2.1 **(Recenter.)** Set \(\mu_j := \text{mean}(S_j)\) for \(j \in (1, \ldots, k)\).
   2.2 **(Reassign).** Update \(S_j := \{x_i : \mu(x_i) = \mu_j\}\) for \(j \in (1, \ldots, k)\).
      ("\(\mu(x_i)\)" means "center closest to \(x_i\); break ties arbitrarily).
Lloyd’s method revisited

1. Choose initial clusters \((S_1, \ldots, S_k)\).

2. Repeat until convergence:
   
   2.1 \textbf{(Recenter.)} Set \(\mu_j := \text{mean}(S_j)\) for \(j \in (1, \ldots, k)\).
   
   2.2 \textbf{(Reassign).} Update \(S_j := \{x_i : \mu(x_i) = \mu_j\}\) for \(j \in (1, \ldots, k)\).
   
   \("\mu(x_i)"\ means \("\text{center closest to } x_i\"; \text{ break ties arbitrarily}).

Geometric perspective:

- Centers define a \textbf{Voronoi diagram/partition}:
  for each \(\mu_j\), define cell \(V_j := \{x \in \mathbb{R}^d : \mu(x) = \mu_j\}\)
  \((\text{break ties arbitrarily}).

- Reassignment leaves assignment consistent with Voronoi cells.

- Recentering might shift data outside Voronoi cells, \textbf{except} if we’ve converged!

- See http://mjt.cs.illinois.edu/htv/ for an interactive demo.
Theorem.

- For all $t$, $\phi(C_t; A_{t-1}) \geq \phi(C_t; A_t) \geq \phi(C_{t+1}; A_t)$.
- The method terminates.
Does Lloyd’s method solve the original problem?

**Theorem.**

- For all $t$, $\phi(C_t; A_{t-1}) \geq \phi(C_t; A_t) \geq \phi(C_{t+1}; A_t)$.
- The method terminates.

**Proof.**

- This first property is from the earlier theorem and the definition of the algorithm:

  $\phi(C_t; A_t) = \phi(C_t; A(C_{t-1})) = \min_{A \in \mathcal{A}} \phi(C_t; A) \leq \phi(C_t, A_{t-1})$,

  $\phi(C_{t+1}; A_t) = \phi(C(A_t); A_t) = \min_{C \in \mathcal{C}} \phi(C; A_t) \leq \phi(C_t, A_t)$,

- Previous property implies: cost is nonincreasing.

Combined with termination condition:

  all but final partition visited at most once.

There are finitely many partitions of $(x_i)_{i=1}^n$.  

□
Does Lloyd’s method solve the original problem?

Theorem.

- For all $t$, $\phi(C_t; A_{t-1}) \geq \phi(C_t; A_t) \geq \phi(C_{t+1}; A_t)$.
- The method terminates.

Proof.

- This first property is from the earlier theorem and the definition of the algorithm:

  \[
  \phi(C_t; A_t) = \phi(C_t; A(C_{t-1})) = \min_{A \in A} \phi(C_t; A) \leq \phi(C_t, A_{t-1}),
  \]

  \[
  \phi(C_{t+1}; A_t) = \phi(C(A_t); A_t) = \min_{C \in C} \phi(C; A_t) \leq \phi(C_t, A_t),
  \]

- Previous property implies: cost is nonincreasing.

Combined with termination condition:
- all but final partition visited at most once.
- There are finitely many partitions of $(x_i)_{i=1}^n$.

(That didn’t answer the question... )
Seriously: does Lloyd’s method solve the original problem?

- In practice, Lloyd’s method seems to optimize well;
- In theory, output can have \textit{unboundedly poor cost}.

\[ \begin{array}{c c c}
\bullet & \times & \bullet \\
\circ & \circ & \bullet \\
\bullet & \times & \bullet \\
\end{array} \]

(Suppose width is \( c > 1 \) and height is 1.)

**K-means cost:**

\[ \sum_{i=1}^{n} \left\| x_i - \mu(x_i) \right\|_2^2 \]
Seriously: does Lloyd’s method solve the original problem?

- In practice, Lloyd’s method seems to optimize well;
  In theory, output can have **unboundedly poor cost**.

  ![Diagram](image)

(Suppose width is $c > 1$ and height is 1.)

- In practice, method takes few iterations;
  in theory: can take $2^{\Omega(\sqrt{n})}$ iterations!

  (Examples of this are painful; but note, problem is NP-hard, and convergence proof used number of partitions... )
Seriously: does Lloyd’s method solve the original problem?

- In practice, Lloyd’s method seems to optimize well; in theory, output can have **unboundedly poor cost.**
  - ![Diagram](attachment:image.png)
  - (Suppose width is $c > 1$ and height is 1.)

- In practice, method takes few iterations; in theory: can take $2^{\Omega(\sqrt{n})}$ iterations!
  - (Examples of this are painful; but note, problem is NP-hard, and convergence proof used number of partitions...)

So: in practice, yes; in theory, don’t know...
Application: vector quantization.

**Vector quantization** with $k$-means.

- Let $(x_i)_{i=1}^n$ be given.
- run $k$-means to obtain $(\mu_1, \ldots, \mu_k)$.
- Replace each $(x_i)_{i=1}^n$ with $(\mu(x_i))_{i=1}^n$.

Encoding size reduces from $O(nd)$ to $O(kd + n \ln(k))$.

**Examples.**

- Audio compression.
- Image compression.
patch quantization, width 10, 32 exemplars
patch quantization, width 10, 128 exemplars
patch quantization, width 10, 512 exemplars
patch quantization, width 25, 32 exemplars
patch quantization, width 25, 128 exemplars
patch quantization, width 25, 256 exemplars
patch quantization, width 50, 8 exemplars
patch quantization, width 50, 64 exemplars
Initialization matters!

- **Easy choices:**
  - $k$ random points from dataset.
  - Random partition.

- **Standard choice** (theory and practice): "$D^2$-sampling"/kmeans++
  1. Choose $\mu_1$ uniformly at random from data.
  2. for $j \in (2, \ldots, k)$:
     2.1 Choose $x_i \propto \min_{l<j} \|x_i - \mu_l\|_2^2$.

- kmeans++ is *randomized furthest-first traversal*; regular furthest-first fails with outliers.

- Scikits-learn and Matlab both default to kmeans++.
We’ve had one main “meta-algorithm” this semester:

- (Regularized) ERM principle: pick the model that minimizes an average loss over training data.
Maximum likelihood: abstract formulation

We’ve had one main “meta-algorithm” this semester:

- (Regularized) ERM principle: pick the model that minimizes an average loss over training data.

We’ve also discussed another:

the “Maximum likelihood estimation (MLE)” principle:

- Pick a set of probability models for your data: \( \mathcal{P} := \{p_\theta : \theta \in \Theta\} \).
  - \( p_\theta \) will denote both densities and masses; the literature is similarly inconsistent.
  - Given samples \((z_i)_{i=1}^n\), pick the model that maximized the likelihood

\[
\max_{\theta \in \Theta} \mathcal{L}(\theta) = \max_{\theta \in \Theta} \prod_{i=1}^n p_\theta(z_i) = \max_{\theta \in \Theta} \sum_{i=1}^n \ln p_\theta(z_i),
\]

where the \( \ln(\cdot) \) is for mathematical convenience, and \( z_i \) can be a labeled pair \((x_i, y_i)\) or just \( x_i \).
Example 1: coin flips.

- We flip a coin of bias $\theta \in [0, 1]$.
- Write down $x_i = 0$ for tails, $x_i = 1$ for heads; then

  $$p_\theta(x_i) = x_i\theta + (1 - x_i)(1 - \theta),$$

or alternatively

  $$p_\theta(x_i) = \theta^{x_i} (1 - \theta)^{1-x_i}.$$

The second form will be more convenient.
Example 1: coin flips.

- We flip a coin of bias \( \theta \in [0, 1] \).
- Write down \( x_i = 0 \) for tails, \( x_i = 1 \) for heads; then
  \[
  p_{\theta}(x_i) = x_i \theta + (1 - x_i)(1 - \theta),
  \]
or alternatively
  \[
  p_{\theta}(x_i) = \theta^{x_i} (1 - \theta)^{1-x_i}.
  \]
The second form will be more convenient.

- Writing \( H := \sum_i x_i \) and \( T := \sum_i (1 - x_i) = n - H \) for convenience,
  \[
  \mathcal{L}(\theta) = \sum_{i=1}^{n} (x_i \ln \theta + (1 - x_i) \ln(1 - \theta)) = H \ln \theta + T \ln(1 - \theta).
  \]
Example 1: coin flips.

- We flip a coin of bias $\theta \in [0, 1]$.
- Write down $x_i = 0$ for tails, $x_i = 1$ for heads; then
  
  $p_{\theta}(x_i) = x_i\theta + (1 - x_i)(1 - \theta),$

  or alternatively
  
  $p_{\theta}(x_i) = \theta^{x_i}(1 - \theta)^{1-x_i}.$

  The second form will be more convenient.
- Writing $H := \sum_i x_i$ and $T := \sum_i (1 - x_i) = n - H$ for convenience,

  $L(\theta) = \sum_{i=1}^n (x_i \ln \theta + (1 - x_i) \ln(1 - \theta)) = H \ln \theta + T \ln(1 - \theta).$

  Differentiating and setting to 0,

  $0 = \frac{H}{\theta} - \frac{T}{1 - \theta},$

  which gives $\theta = \frac{H}{T + H} = \frac{H}{N}.$
- In this way, we’ve justified a natural algorithm.
Example 2: mean of a Gaussian

Suppose $x_i \sim \mathcal{N}(\mu, \sigma^2)$, so $\theta = (\mu, \sigma^2)$, and

$$\ln p_\theta(x_i) = \ln \frac{\exp \left( -\frac{(x_i - \mu)^2}{2\sigma^2} \right)}{\sqrt{2\pi\sigma^2}} = -\frac{(x_i - \mu)^2}{2\sigma^2} - \frac{\ln(2\pi\sigma^2)}{2}.$$
Example 2: mean of a Gaussian

- Suppose \( x_i \sim \mathcal{N}(\mu, \sigma^2) \), so \( \theta = (\mu, \sigma^2) \), and

\[
\ln p_{\theta}(x_i) = \ln \frac{\exp \left( -\frac{(x_i - \mu)^2}{2\sigma^2} \right) }{\sqrt{2\pi\sigma^2}} = -\frac{(x_i - \mu)^2}{2\sigma^2} - \ln(2\pi\sigma^2) / 2.
\]

- Therefore

\[
\mathcal{L}(\theta) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2 + \text{stuff without } \mu;
\]

applying \( \nabla_{\mu} \) and setting to zero gives \( \mu = \frac{1}{n} \sum_{i} x_i \).

- A similar derivation gives \( \sigma^2 = \frac{1}{n} \sum_{i} (x_i - \mu)^2 \).
Example 4: Naive Bayes

Let’s try a simple prediction setup, with (Bayes) optimal classifier

$$\arg \max_{y \in Y} p(Y = y | X = x).$$

(We haven’t discussed this concept a lot, but it’s widespread in ML.)
Example 4: Naive Bayes

- Let’s try a simple prediction setup, with (Bayes) optimal classifier
  \[
  \arg \max_{y \in Y} p(Y = y | X = x).
  \]
  (We haven’t discussed this concept a lot, but it’s widespread in ML.)
- One way to proceed is to learn \( p(Y | X) \) exactly; that’s a pain.
Example 4: Naive Bayes

Let’s try a simple prediction setup, with (Bayes) optimal classifier

\[
\arg \max_{y \in Y} p(Y = y | X = x).
\]

(We haven’t discussed this concept a lot, but it’s widespread in ML.)

One way to proceed is to learn \( p(Y | X) \) exactly; that’s a pain.

Let’s assume coordinates of \( X = (X_1, \ldots, X_d) \) are independent given \( Y \):

\[
p(Y = y | X = x) = \frac{p(Y = y, X = x)}{p(X = x)} = \frac{p(X = x | Y = y)p(Y = y)}{p(X = x)}
\]

\[
= \frac{p(Y = y) \prod_{j=1}^{d} p(X_j = x_j | Y = y)}{p(X = x)},
\]

and

\[
\arg \max_{y \in Y} p(Y = y | X = x) = \arg \max_{y \in Y} p(Y = y) \prod_{j=1}^{d} p(X_j = x_j | Y = y).
\]
Example 4: Naive Bayes (part 2)

\[
\arg \max_{y \in Y} p(Y = y|X = x) = \arg \max_{y \in Y} \frac{p(Y = y)}{p(X = x)} \prod_{j=1}^{d} p(X = x_j|Y = y).
\]
Example 4: Naive Bayes (part 2)

\[
\arg\max_{y \in \mathcal{Y}} p(Y = y|X = \mathbf{x}) = \arg\max_{y \in \mathcal{Y}} p(Y = y) \prod_{j=1}^{d} p(X = x_j|Y = y).
\]

Examples where this helps:

- Suppose \( \mathbf{X} \in \{0, 1\}^d \) has an arbitrary distribution; it’s specified with \( 2^d - 1 \) numbers. The factored form above needs \( d \) numbers. To see how this can help, suppose \( \mathbf{x} \in \{0, 1\}^d \); instead of having to learn a probability model of \( 2^d \) possibilities, we now have to learn \( d + 1 \) models each with 2 possibilities (binary labels).

- HW5 will use the standard “Iris dataset”. This data is continuous, Naive Bayes would approximate univariate distributions.
Suppose data is drawn from \( k \) Gaussians, meaning
\[
Y = j \sim \text{Discrete}(\pi_1, \ldots, \pi_k),
\]
\[
X = x | Y = j \sim \mathcal{N}(\mu_j, \Sigma_j),
\]
and the parameters are \( \theta = ((\pi_1, \mu_1, \Sigma_1), \ldots, (\pi_k, \mu_k, \Sigma_k)) \).

(Note: this is a **generative** model, and we have a way to sample.)
Gaussian Mixture Model

- Suppose data is drawn from $k$ Gaussians, meaning
  \begin{align*}
  Y = j &\sim \text{Discrete}(\pi_1, \ldots, \pi_k), \\
  X = x|Y = j &\sim \mathcal{N}(\mu_j, \Sigma_j),
  \end{align*}
  and the parameters are $\theta = ((\pi_1, \mu_1, \Sigma_1), \ldots, (\pi_k, \mu_k, \Sigma_k))$.
  (Note: this is a \textbf{generative} model, and we have a way to sample.)

- The probability density (with parameters $\theta = ((\pi_j, \mu_j, \Sigma_j))_{j=1}^k$) at a given $x$ is
  \begin{align*}
  p_\theta(x) &= \sum_{j=1}^k p_\theta(x|y = j)p_\theta(y = j) \\
  &= \sum_{j=1}^k p_{\mu_j, \Sigma_j}(x|Y = j)p_{\theta}(y = j) \\
  &= \prod_{j=1}^k \pi_j \
  \end{align*}
  and the likelihood problem is
  \begin{align*}
  \mathcal{L}(\theta) &= \sum_{i=1}^n \ln \sum_{j=1}^k \frac{\pi_j}{\sqrt{(2\pi)^d|\Sigma|}} \exp \left(-\frac{1}{2}(x_i - \mu_j)^T \Sigma^{-1} (x_i - \mu_j) \right).\end{align*}
  \[\text{The ln and the exp are no longer next to each other; we can't just take the derivative and set the answer to 0.}\]
Pearson’s crabs.

Statistician Karl Pearson wanted to understand the distribution of “forehead breadth to body length” for 1000 crabs

Doesn’t look Gaussian!
Pearson’s crabs.

Statistician Karl Pearson wanted to understand the distribution of “forehead breadth to body length” for 1000 crabs.

Doesn’t look Gaussian!
Pearson’s crabs.

Statistician Karl Pearson wanted to understand the distribution of “forehead breadth to body length” for 1000 crabs.

Pearson fit a **mixture of two Gaussians**.
Pearson’s crabs.

Statistician Karl Pearson wanted to understand the distribution of “forehead breadth to body length” for 1000 crabs

Pearson fit a mixture of two Gaussians.

Remark. Pearson did not use E-M. For this he invented the “method of moments” and obtained a solution by hand.
Gaussian mixture likelihood with responsibility matrix $R$

Let’s replace $\sum_{i=1}^{n} \ln \sum_{j=1}^{k} \pi_j p_{\mu_j, \Sigma_j}(x_i)$ with

$$\sum_{i=1}^{n} \sum_{j=1}^{k} R_{ij} \ln \left( \pi_j p_{\mu_j, \Sigma_j}(x_i) \right)$$

where $\boldsymbol{R} \in \mathcal{R}_{n,k} := \{ \boldsymbol{R} \in [0, 1]^{n \times k} : \boldsymbol{R} \mathbf{1}_k = \mathbf{1}_n \}$ is a responsibility matrix.
Gaussian mixture likelihood with \textit{responsibility matrix} $\mathbf{R}$

Let’s replace \( \sum_{i=1}^{n} \ln \sum_{j=1}^{k} \pi_{j} p_{\mu_{j}, \Sigma_{j}}(\mathbf{x}_{i}) \) with

\[
\sum_{i=1}^{n} \sum_{j=1}^{k} R_{ij} \ln \left( \pi_{j} p_{\mu_{j}, \Sigma_{j}}(\mathbf{x}_{i}) \right)
\]

where \( \mathbf{R} \in \mathcal{R}_{n,k} := \{ \mathbf{R} \in [0, 1]^{n \times k} : \mathbf{R}1_{k} = 1_{n} \} \) is a \textit{responsibility matrix}.

Holding $\mathbf{R}$ fixed and optimizing $\theta$ gives

\[
\pi_{j} := \frac{\sum_{i=1}^{n} R_{ij}}{\sum_{i=1}^{n} \sum_{l=1}^{k} R_{il}} = \frac{\sum_{i=1}^{n} R_{ij}}{n};
\]

\[
\mu_{j} := \frac{\sum_{i=1}^{n} R_{ij} \mathbf{x}_{i}}{\sum_{i=1}^{n} R_{ij}} = \frac{\sum_{i=1}^{n} R_{ij} \mathbf{x}_{i}}{n \pi_{j}},
\]

\[
\Sigma_{j} := \frac{\sum_{i=1}^{n} R_{ij}(\mathbf{x}_{i} - \mu_{j})(\mathbf{x}_{i} - \mu_{j})^{T}}{n \pi_{j}}.
\]

(Should use new mean in $\Sigma_{j}$ so that all derivatives $0$.)
We introduced an **assignment matrix** $A \in \{0, 1\}^{n \times k}$:

- For each $x_i$, define $\mu(x_i)$ to be a closest center:
  \[ \|x_i - \mu(x_i)\| = \min_j \|x_i - \mu_j\|. \]

- For each $i$, set $A_{ij} = 1[\mu(x_i) = \mu_j]$. 

What fulfills the same role for $L$?
Generalizing the assignment matrix to GMMs

We introduced an assignment matrix \( A \in \{0, 1\}^{n \times k} \):

- For each \( x_i \), define \( \mu(x_i) \) to be a closest center:
  \[
  \|x_i - \mu(x_i)\| = \min_j \|x_i - \mu_j\|.
  \]

- For each \( i \), set \( A_{ij} = 1[\mu(x_i) = \mu_j] \).

- **Key property:** by this choice,
  \[
  \phi(C; A) = \sum_{i=1}^{n} \sum_{j=1}^{k} A_{ij} \|x_i - \mu_j\|^2 = \sum_{i=1}^{n} \min_j \|x_i - \mu_j\|^2 = \phi(C);
  \]

therefore we can decrease \( \phi(C) = \phi(C; A) \)

first by optimizing \( C \) to get \( \phi(C'; A) \leq \phi(C; A) \),

then setting \( A \) as above to get

\[
\phi(C') = \phi(C'; A') \leq \phi(C'; A) \leq \phi(C; A) = \phi(C).
\]

**In other words:** we minimize \( \phi(C) \) via \( \phi(C; A) \).
Generalizing the assignment matrix to GMMs

We introduced an **assignment matrix** $A \in \{0, 1\}^{n \times k}$:

- For each $x_i$, define $\mu(x_i)$ to be a closest center:
  $$\|x_i - \mu(x_i)\| = \min_j \|x_i - \mu_j\|.$$

- For each $i$, set $A_{ij} = 1[\mu(x_i) = \mu_j]$.

- **Key property:** by this choice,
  $$\phi(C; A) = \sum_{i=1}^{n} \sum_{j=1}^{k} A_{ij} \|x_i - \mu_j\|^2 = \sum_{i=1}^{n} \min_j \|x_i - \mu_j\|^2 = \phi(C);$$

therefore we can decrease $\phi(C) = \phi(C; A)$ first by optimizing $C$ to get $\phi(C'; A) \leq \phi(C; A)$, then setting $A$ as above to get

$$\phi(C') = \phi(C'; A') \leq \phi(C'; A) \leq \phi(C; A) = \phi(C).$$

**In other words:** we minimize $\phi(C)$ via $\phi(C; A)$.

What fulfills the same role for $\mathcal{L}$?
Define **augmented likelihood**

\[
\mathcal{L}(\theta; R) := \sum_{i=1}^{n} \sum_{j=1}^{k} R_{ij} \ln \frac{p_\theta(x_i, y_i = j)}{R_{ij}},
\]

with **responsibility matrix** \( R \in \mathcal{R}_{n,k} := \{ R \in [0, 1]^{n \times k} : R1_k = 1_n \} \).

Alternate two steps:

- **E-step:** set \((R_t)_{ij} := p_{\theta_{t-1}}(y_i = j | x_i)\).
- **M-step:** set \(\theta_t = \arg \max_{\theta \in \Theta} \mathcal{L}(\theta; R_t)\).
Define **augmented likelihood**

\[
\mathcal{L}(\theta; R) := \sum_{i=1}^{n} \sum_{j=1}^{k} R_{ij} \ln \frac{p_{\theta}(x_i, y_i = j)}{R_{ij}},
\]

with **responsibility matrix** \( R \in \mathcal{R}_{n,k} := \{ R \in [0, 1]^{n \times k} : R1_k = 1_n \} \).

Alternate two steps:

- **E-step**: set \((R_t)_{ij} := p_{\theta_{t-1}}(y_i = j| x_i)\).
- **M-step**: set \(\theta_t = \arg \max_{\theta \in \Theta} \mathcal{L}(\theta; R_t)\).

**Soon**: we’ll see this gives nondecreasing likelihood!
Initialization: a standard choice is \( \pi_j = 1/k \), \( \Sigma_j = I \), and \( (\mu_j)_{j=1}^k \) given by \( k \)-means.

- **E-step:** Set \( R_{ij} = p_\theta(y_i = j|x_i) \), meaning

\[
R_{ij} = p_\theta(y_i = j|x_i) = \frac{p_\theta(y_i = j, x_i)}{p_\theta(x_i)} = \frac{\pi_j p_\mu_j, \Sigma_j(x_i)}{\sum_{l=1}^k \pi_l p_\mu_l, \Sigma_l(x_i)}.
\]

- **M-step:** solve \( \arg \max_{\theta \in \Theta} \mathcal{L}(\theta; R) \), meaning

\[
\pi_j := \frac{\sum_{i=1}^n R_{ij}}{\sum_{i=1}^n \sum_{l=1}^k R_{il}} = \frac{\sum_{i=1}^n R_{ij}}{n}, \\
\mu_j := \frac{\sum_{i=1}^n R_{ij} x_i}{\sum_{i=1}^n R_{ij}} = \frac{\sum_{i=1}^n R_{ij} x_i}{n \pi_j}, \\
\Sigma_j := \frac{\sum_{i=1}^n R_{ij}(x_i - \mu_j)(x_i - \mu_j)^\top}{n \pi_j}.
\]

(These are as before.)
Demo: elliptical clusters

E...
Demo: elliptical clusters

E... M...
Demo: elliptical clusters

E... M... E...
Demo: elliptical clusters

E... M... E... M...
Demo: elliptical clusters
Demo: elliptical clusters

E... M... E... M... E... M...
Demo: elliptical clusters

E... M... E... M... E... M... E...
Demo: elliptical clusters
Demo: elliptical clusters
Demo: elliptical clusters
Demo: elliptical clusters

E... M... E... M... E... M... E... M...
Demo: elliptical clusters
Demo: elliptical clusters
Demo: elliptical clusters
Demo: elliptical clusters
Demo: elliptical clusters
Demo: elliptical clusters
Demo: elliptical clusters
Demo: elliptical clusters
Demo: elliptical clusters
Demo: elliptical clusters
Demo: elliptical clusters
Demo: elliptical clusters

E... M... E... M... E... M... E... M...
Demo: elliptical clusters
Demo: elliptical clusters
Demo: elliptical clusters
Demo: elliptical clusters
Demo: elliptical clusters
Theorem.
Suppose \((R_0, \theta_0) \in \mathcal{R}_{n,k} \times \Theta\) arbitrary, thereafter \((R_t, \theta_t)\) given by E-M:

\[
(R_t)_{ij} := p_{\theta_{t-1}}(y = j | x_i), \quad \text{and} \quad \theta_t := \arg \max_{\theta \in \Theta} \mathcal{L}(\theta; R_t)
\]

Then

\[
\mathcal{L}(\theta_t; R_t) \leq \max_{R \in \mathcal{R}_{n \times k}} \mathcal{L}(\theta_t; R) = \mathcal{L}(\theta_t; R_{t+1}) = \mathcal{L}(\theta_t)
\]

\[
\leq \mathcal{L}(\theta_{t+1}; R_{t+1}).
\]

In particular, \(\mathcal{L}(\theta_t) \leq \mathcal{L}(\theta_{t+1}).\)
**Theorem.**
Suppose \((R_0, \theta_0) \in \mathcal{R}_{n,k} \times \Theta\) arbitrary, thereafter \((R_t, \theta_t)\) given by E-M:

\[
(R_t)_{ij} := p_{\theta_{t-1}}(y = j | x_i). \quad \text{and} \quad \theta_t := \arg\max_{\theta \in \Theta} \mathcal{L}(\theta; R_t)
\]

Then

\[
\mathcal{L}(\theta_t; R_t) \leq \max_{R \in \mathcal{R}_{n \times k}} \mathcal{L}(\theta_t; R) = \mathcal{L}(\theta_t; R_{t+1}) = \mathcal{L}(\theta_t) \\
\leq \mathcal{L}(\theta_{t+1}; R_{t+1}).
\]

In particular, \(\mathcal{L}(\theta_t) \leq \mathcal{L}(\theta_{t+1}).\)

**Remarks.**

- We proved a similar guarantee for \(k\)-means, which is also an *alternating minimization* scheme.
- Similarly, MLE for Gaussian mixtures is NP-hard; it is also known to need exponentially many samples in \(k\) to information-theoretically recover the parameters.
E-M for GMMs **still works** if we freeze or constrain some parameters.
Parameter constraints.

E-M for GMMs still works if we freeze or constrain some parameters.

Examples:

- **No weights**: initialize $\pi = (\frac{1}{k}, \ldots, \frac{1}{k})$ and never update it.

- **Diagonal covariance matrices**: update everything as before, except $\Sigma_j := \text{diag}((\sigma_j)_1^2, \ldots, (\sigma_j)_d^2)$ where

$$ (\sigma_j)_l^2 := \frac{\sum_{i=1}^n R_{ij} (x_i - \mu_j)_l^2}{n \pi_j}, $$

that is: we use coordinate-wise sample variances weighted by $R$. Why is this a good idea?
E-M for GMMs **still works** if we freeze or constrain some parameters.

**Examples:**

- **No weights:** initialize $\pi = (\frac{1}{k}, \ldots, \frac{1}{k})$ and never update it.

- **Diagonal covariance matrices:** update everything as before, except $\Sigma_j := \text{diag}((\sigma_j)_1^2, \ldots, (\sigma_j)_d^2)$ where

  $$(\sigma_j)_l^2 := \frac{\sum_{i=1}^n R_{ij} (x_i - \mu_j)_l^2}{n \pi_j};$$

  that is: we use coordinate-wise sample variances weighted by $R$. **Why is this a good idea?**

  Computation (of inverse), sample complexity, . . .
Singularities

E-M with GMMs suffers from **singularities**: trivial situations where the likelihood goes to $\infty$ but the solution is bad.

> Suppose:
> $d = 1$, $k = 2$, $\pi_j = \frac{1}{2}$,
> $n = 3$ with $x_1 = -1$ and $x_2 = +1$ and $x_3 = +3$.
> Initialize with $\mu_1 = 0$ and $\sigma_1 = 1$,
> but $\mu_2 = +3 = x_3$ and $\sigma_2 = \frac{1}{100}$.
> Then $\sigma_2 \rightarrow 0$ and $\mathcal{L} \uparrow \infty$. 
Recall the \textit{generative story} for GMMs:

\[ Y \sim \text{Discrete}(\pi_1, \ldots, \pi_k) \quad \text{pick a Gaussian;} \]
\[ X|Y = j \sim \mathcal{N}(\mu_j, \Sigma_j) \quad \text{pick a point.} \]

\( Y \) is \textit{latent/hidden/unobserved}, \( X \) is \textit{observed}. 
Recall the **generative story** for GMMs:

\[
Y \sim \text{Discrete}(\pi_1, \ldots, \pi_k) \quad \text{pick a Gaussian;}
\]

\[
X | Y = j \sim \mathcal{N}(\mu_j, \Sigma_j) \quad \text{pick a point.}
\]

\( Y \) is **latent/hidden/unobserved**, \( X \) is **observed**.

This model consists of random variables \((X, Y)\) with a specific **conditional dependence structure**.
Graphical models

Recall the **generative story** for GMMs:

\[
Y \sim \text{Discrete}(\pi_1, \ldots, \pi_k) \quad \text{pick a Gaussian;}
\]

\[
X|Y = j \sim \mathcal{N}(\mu_j, \Sigma_j) \quad \text{pick a point.}
\]

\(Y\) is **latent/hidden/unobserved**, \(X\) is **observed**.

This model consists of random variables \((X, Y)\) with a specific **conditional dependence structure**.

A **graphical model** is a compact way of representing a family of r.v.’s, most notably their conditional dependencies. Typically, the graphical model gives us

- a way to write down the (joint) probability distribution,
- guidance on how to sample.
Basic rules (there are many more):

- Nodes denote random variables. (Here we have \((X, Y)\).)
- Edges denote conditional dependence. (Here \(X\) depends on \(Y\).)
- Shaded nodes (e.g., \(X\)) are observed; unshaded (\(Y\)) are unobserved.
Graphical model for GMMs

**Basic rules** (there are many more):

- Nodes denote random variables. (Here we have $(X, Y)$.)
- Edges denote conditional dependence. (Here $X$ depends on $Y$.)
- Shaded nodes (e.g., $X$) are observed; unshaded ($Y$) are unobserved.

Likelihood of observations $(X_1, \ldots, X_n)$ drawn from GMM:

$$p(X_1, \ldots, X_n) = \sum_{j_1 \in \{1, \ldots, k\}, \ldots, j_n \in \{1, \ldots, k\}} p(X_1, \ldots, X_n, Y_1 = j_1, \ldots, Y_n = j_n)$$

$$= \sum_{j_1 \in \{1, \ldots, k\}} \prod_{i=1}^{n} p(Y_i = j_i) p(X_i | Y_i = j_i) = \prod_{i=1}^{n} \sum_{j_i = 1}^{k} p(Y_i = j_i) p(X_i | Y_i = j_i).$$
Recall the **Naive Bayes model:**
both inputs and outputs \((X, Y)\) are observed (both are shaded!);
coordinates \((X_1, \ldots, X_d)\) are *conditionally independent given* \(Y\)
(as indicated by the arrows!).
Why do people use graphical models?

- Easy to interpret how data inter-depends, flows.
- Easy to add nodes and edges based on observations and beliefs.
- MLE, E-M, and others provide a well-weathered toolbox to fit them to data, sample, etc.
- Were very popular in the natural sciences (easy to encode domain knowledge); not yet clear how deep networks are displacing them (how to encode prior knowledge with deep networks?).
Basic generative story

Generative networks provide a way to \textbf{sample} from any distribution.

1. Sample $z \sim \mu$, where $\mu$ denotes an efficiently sampleable distribution (e.g., uniform or Gaussian).
2. Output $g(z)$, where $g : \mathbb{R}^d \rightarrow \mathbb{R}^m$ is a deep network.

\textbf{Notation:} let $g\#\mu$ (pushforward of $\mu$ through $g$) denote this distribution.
Basic generative story

Generative networks provide a way to sample from any distribution.

1. Sample $z \sim \mu$, where $\mu$ denotes an efficiently sampleable distribution (e.g., uniform or Gaussian).
2. Output $g(z)$, where $g : \mathbb{R}^d \rightarrow \mathbb{R}^m$ is a deep network.

Notation: let $g\#\mu$ (pushforward of $\mu$ through $g$) denote this distribution.

Brief remarks:

- Can this model any target distribution $\nu$? Yes, (roughly) for the same reason that $g$ can approximate any $f : \mathbb{R}^d \rightarrow \mathbb{R}^m$.
- Graphical models let us sample and estimate probabilities; what about here? Nope.
Encoding/decoding was one motivation for unsupervised learning. PCA gives the optimal linear encoding:

\[ X = USV^T \]

\[ \min_{E \in \mathbb{R}^{d \times k}, D \in \mathbb{R}^{k \times d}} \| X - XED \|_F^2 = \| X - XV_kV_k^T \|_F^2. \]
Encoding/decoding was one motivation for unsupervised learning. PCA gives the optimal linear encoding:

**PCA Theorem (part of it).** Given $X = USV^T$ and $k \leq r$, 

$$
\min_{E \in \mathbb{R}^{d \times k}, D \in \mathbb{R}^{k \times d}} \|X - XED\|_F^2 = \|X - XV_k V_k^T\|_F^2.
$$
Encoding with deep networks

Let encoders $\mathcal{E}$ and decoders $\mathcal{D}$ denote families of deep networks from $\mathbb{R}^d$ to $\mathbb{R}^k$ and back.

$$\min_{f \in \mathcal{E}} \min_{g \in \mathcal{D}} \frac{1}{n} \sum_{i=1}^{n} \| x_i - g(f(x_i)) \|^2_2.$$
Encoding with deep networks

Let encoders $\mathcal{E}$ and decoders $\mathcal{D}$ denote families of deep networks from $\mathbb{R}^d$ to $\mathbb{R}^k$ and back.

$$\min_{f \in \mathcal{E}} \frac{1}{n} \sum_{i=1}^{n} \| x_i - g(f(x_i)) \|_2^2.$$ 

Remarks.

- This is called an autoencoder.
- $\mathbb{R}^k$ is the latent space, $f(x) \in \mathbb{R}^k$ is latent representation of $x$.
- As with PCA, can relate weights of $f$ and $g$; e.g., $g$ uses the transposes of $f$’s matrices. (Not necessary, though.)
- Does small $k$ enforce positive error? No; powerful $\mathcal{E}$ and $\mathcal{D}$ can do $f(x_i) = \frac{1}{i}$ and $g(\frac{1}{i}) = x_i$.
- Therefore we must regularize in some other way!
- Does squared error matter? Not really; it’s a choice like everything else in ML.
Another way to visualize generative networks

Given a sample from a distribution (even $g \neq \mu$), here’s the “kernel density” / “Parzen window” estimate of its density:

1. Start with random draw $(x_i)_{i=1}^n$.
2. “Place bumps at every $x_i$”:
   Define $\hat{p}(x) := \frac{1}{n} \sum_{i=1}^n k \left( \frac{x-x_i}{h} \right)$, where $k$ is a kernel function (not the SVM one!), $h$ is the “bandwidth”; for example:

   - **Gaussian**: $k(z) \propto \exp\left(-\frac{\|z\|^2}{2}\right)$;
   - **Epanechnikov**: $k(z) \propto \max\{0, 1 - \|z\|^2\}$. 
Another way to visualize generative networks

Given a sample from a distribution (even $g \neq \mu$), here's the "kernel density" / "Parzen window" estimate of its density:

1. Start with random draw $(x_i)_{i=1}^n$.
2. "Place bumps at every $x_i$":
   Define $\hat{p}(x) := \frac{1}{n} \sum_{i=1}^n k\left( \frac{x-x_i}{h} \right)$, where $k$ is a kernel function (not the SVM one!), $h$ is the "bandwidth"; for example:
   - Gaussian: $k(z) \propto \exp\left(-\|z\|^2/2\right)$;
   - Epanechnikov: $k(z) \propto \max\{0, 1 - \|z\|^2\}$. 
(Variational) Autoencoders

- **Autoencoder:**

  \[ x_i \xrightarrow{f \text{ map}} \text{latent } z_i = f(x_i) \xrightarrow{g \text{ map}} \hat{x}_i = g(z_i). \]

  Objective: \[ \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, \hat{x}_i). \]
(Variational) Autoencoders

▶ Autoencoder:

\[ \mathbf{x}_i \xrightarrow{\text{map}} \text{latent } \mathbf{z}_i = f(\mathbf{x}_i) \xrightarrow{\text{map}} \hat{\mathbf{x}}_i = g(\mathbf{z}_i). \]

Objective: \[ \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{x}_i, \hat{\mathbf{x}}_i). \]

▶ Variational Autoencoder:

\[ \mathbf{x}_i \xrightarrow{\text{map}} \text{latent distribution } \mu_i = f(\mathbf{x}_i) \xrightarrow{\text{pushforward}} \hat{\mathbf{x}}_i \sim g\#\mu_i. \]

Objective: \[ \frac{1}{n} \sum_{i=1}^{n} \left[ \ell(\mathbf{x}_i, \hat{\mathbf{x}}_i) + \lambda \text{KL}(\mu, \mu_i) \right]. \]
\[ f_1(x) \leftarrow \text{mean} \]
\[ f_2(x) \leftarrow \log \text{variance} \]

\[ z = f_1(x) + \text{torch.randn}(\text{ch}, 1) \times \exp\left(\frac{f_2(x)}{2}\right) \]

\[ g(z) \]
Generative network setup and training.

- We are given \((x_i)_{i=1}^n \sim \nu\).
- We want to find \(g\) so that \((g(z_i))_{i=1}^n \approx (x_i)_{i=1}^n\), where \((z_i)_{i=1}^n \sim \mu\).

**Problem:** this isn’t as simple as fitting \(g(z_i) \approx x_i\).
Generative network setup and training.

- We are given \((x_i)_{i=1}^n \sim \nu\).
- We want to find \(g\) so that \((g(z_i))_{i=1}^n \approx (x_i)_{i=1}^n\), where \((z_i)_{i=1}^n \sim \mu\).

**Problem:** this isn't as simple as fitting \(g(z_i) \approx x_i\).

**Solutions:**

- **VAE:** For each \(x_i\), construct distribution \(\mu_i\), so that \(\hat{x}_i \sim g \# \mu_i\) and \(x_i\) are close, as are \(\mu_i\) and \(\mu\).
  To generate fresh samples, get \(z \sim \mu\) and output \(g(z)\).

- **GAN:** Pick a distance notion between distributions (or between samples \((g(z_i))_{i=1}^n\) and \((x_i)_{i=1}^n\)) and pick \(g\) to minimize that!
GAN overview

**GAN approach:** we minimize $D(\nu, g\#\mu)$ directly, where “$D$” is some notion of distance/divergence:

- [ ] **Jensen-Shannon Divergence** (original GAN paper).
- [ ] **Wasserstein distance** (influential follow-up).
GAN overview

**GAN approach:** we minimize $D(\nu \parallel g \# \mu)$ directly, where “$D$” is some notion of distance/divergence:

- **Jensen-Shannon Divergence** (original GAN paper).
- **Wasserstein distance** (influential follow-up).

Each distance is computed with an *alternating/adversarial* scheme:

1. We have some current choice $g_t$, and use it to produce a sample $(\hat{x}_i)_{i=1}^n$ with $\hat{x}_i = g_t(z_i)$.
2. We train a *discriminator/critic* $f_t$ to find differences between $(\hat{x}_i)_{i=1}^n$ and $(x_i)_{i=1}^n$.
3. We then pick a new *generator* $g_{t+1}$, trained to fool $f_t$!
Original GAN formulation

Let \( p, p_g \) denote density of data and generator, \( \tilde{p} = \frac{p}{2} + \frac{p_g}{2} \).

Original GAN minimizes **Jensen-Shannon Divergence**:

\[
2 \cdot JS(p, p_g) = KL(p, \tilde{p}) + KL(p_g, \tilde{p})
\]

\[
= \int p(x) \ln \frac{p(x)}{\tilde{p}(x)} \, dx + \int p_g(x) \ln \frac{p_g(x)}{\tilde{p}(x)} \, dx
\]

\[
= \mathbb{E}_p \ln \frac{p(x)}{\tilde{p}(x)} + \mathbb{E}_{p_g} \ln \frac{p_g(x)}{\tilde{p}(x)}.
\]
Original GAN formulation

Let $p, p_g$ denote density of data and generator, $\tilde{p} = \frac{p}{2} + \frac{p_g}{2}$.

Original GAN minimizes **Jensen-Shannon Divergence**:

$$2 \cdot JS(p, p_g) = KL(p, \tilde{p}) + KL(p_g, \tilde{p})$$

$$= \int p(x) \ln \frac{p(x)}{\tilde{p}(x)} \, dx + \int p_g(x) \ln \frac{p_g(x)}{\tilde{p}(x)} \, dx$$

$$= E_p \ln \frac{p(x)}{\tilde{p}(x)} + E_{p_g} \ln \frac{p_g(x)}{\tilde{p}(x)}.$$

But we’ve been saying we can’t write down $p_g$?
Let $p, p_g$ denote density of data and generator, $\tilde{p} = \frac{p}{2} + \frac{p_g}{2}$.

Original GAN minimizes **Jensen-Shannon Divergence**: 

$$2 \cdot JS(p, p_g) = KL(p, \tilde{p}) + KL(p_g, \tilde{p})$$

$$= \int p(x) \ln \frac{p(x)}{\tilde{p}(x)} \, dx + \int p_g(x) \ln \frac{p_g(x)}{\tilde{p}(x)} \, dx$$

$$= \mathbb{E}_{p} \ln \frac{p(x)}{\tilde{p}(x)} + \mathbb{E}_{p_g} \ln \frac{p_g(x)}{\tilde{p}(x)}.$$

But we’ve been saying we can’t write down $p_g$?

Original GAN approach applies alternating minimization to

$$\inf_{g \in G} \sup_{f \in \mathcal{F}} \left[ \frac{1}{n} \sum_{i=1}^{n} \ln \left( f(x_i) \right) + \frac{1}{m} \sum_{j=1}^{m} \ln \left( 1 - f(g(z_j)) \right) \right].$$
Original GAN formulation and algorithm.

**Original GAN objective:**

\[
\inf_{g \in G} \sup_{f \in F} \left[ \frac{1}{n} \sum_{i=1}^{n} \ln \left( f(x_i) \right) + \frac{1}{m} \sum_{j=1}^{m} \ln \left( 1 - f(g(z_j)) \right) \right].
\]

**Algorithm alternates these two steps:**

1. Hold \( g \) fixed and optimize \( f \). Specifically, generate a sample \( (\hat{x}_j)_{j=1}^{m} = (g(z_j))_{j=1}^{m} \), and approximately optimize

\[
\sup_{f \in F} \left[ \frac{1}{n} \sum_{i=1}^{n} \ln \left( f(x_i) \right) + \frac{1}{m} \sum_{j=1}^{m} \ln \left( 1 - f(\hat{x}_j) \right) \right].
\]

2. Hold \( f \) fixed and optimize \( g \). Specifically, generate \( (z_j)_{j=1}^{m} \) and approximately optimize

\[
\inf_{g \in G} \left[ \frac{1}{n} \sum_{i=1}^{n} \ln \left( f(x_i) \right) + \frac{1}{m} \sum_{j=1}^{m} \ln \left( 1 - f(g(z_j)) \right) \right].
\]
Algorithm alternates these two steps:

1. Hold $g$ fixed and optimize $f$. Specifically, generate a sample $(\hat{x}_j)_{j=1}^m = (g(z_j))_{j=1}^m$, and approximately optimize

$$\sup_{f\in\mathcal{F}, f:X\to(0,1)} \left[ \frac{1}{n} \sum_{i=1}^n \ln (f(x_i)) + \frac{1}{m} \sum_{j=1}^m \ln (1 - f(\hat{x}_j)) \right].$$

2. Hold $f$ fixed and optimize $g$. Specifically, generate $(z_j)_{j=1}^m$ and approximately optimize

$$\inf_{g\in\mathcal{G}} \left[ \frac{1}{n} \sum_{i=1}^n \ln (f(x_i)) + \frac{1}{m} \sum_{j=1}^m \ln (1 - f(g(z_j))) \right].$$

Remarks.

► Common practice: do many $f$ ascents for each $g$ descent.
► Training has all sorts of instabilities and heuristics fixes; e.g., **mode collapse** ($g$ outputs a small subset of training elements).
► Original intuition was game-theoretic: generator and critic compete.
Wasserstein GAN (WGAN)

**Original GAN objective:**

\[
\inf_{g \in \mathcal{G}} \sup_{f \in \mathcal{F}} \left[ \frac{1}{n} \sum_{i=1}^{n} \ln (f(x_i)) + \frac{1}{m} \sum_{j=1}^{m} \ln (1 - f(g(z_j))) \right].
\]
**Wasserstein GAN (WGAN)**

**Original GAN objective:**

\[
\inf_{g \in \mathcal{G}} \sup_{f \in \mathcal{F}} \left[ \sum_{i=1}^{n} \frac{1}{n} \ln (f(x_i)) + \frac{1}{m} \sum_{j=1}^{m} \ln (1 - f(g(z_j))) \right].
\]

**Wasserstein GAN objective:**

\[
\inf_{g \in \mathcal{G}} \sup_{f \in \mathcal{F}, \|f\|_{\text{Lip}} \leq 1} \left[ \sum_{i=1}^{n} \frac{1}{n} f(x_i) - \sum_{j=1}^{m} \frac{1}{m} f(g(z_j)) \right],
\]

where “\(\|f\|_{\text{Lip}} \leq 1\)” means \(f\) 1-Lipschitz (\(\|f(x) - f(y)\| \leq \|x - y\|\)).
WGAN remarks

Wasserstein GAN objective:

\[
\inf_{g \in G} \sup_{f \in F} \left[ \frac{1}{n} \sum_{i=1}^{n} f(x_i) - \frac{1}{m} \sum_{j=1}^{m} f(g(z_j)) \right],
\]

where "\( \| f \|_{\text{Lip}} \leq 1 \)" means \( f \) 1-Lipschitz (\( \| f(x) - f(y) \| \leq \| x - y \| \)).
Wasserstein GAN objective:

\[
\inf_{g \in G} \sup_{f \in F} \left[ \frac{1}{n} \sum_{i=1}^{n} f(x_i) - \frac{1}{m} \sum_{j=1}^{m} f(g(z_j)) \right],
\]

where “\( \|f\|_{\text{Lip}} \leq 1 \)” means \( f \) 1-Lipschitz (\( \|f(x) - f(y)\| \leq \|x - y\| \)).

**Remarks.**

- In practice, \( G \) and \( F \) are deep networks architectures, \( \|f\|_{\text{Lip}} \) is only approximately enforced.
- This objective is a “Wasserstein distance” or “earth mover distance”; it can be interpreted as how much mass we have to shift to convert one distribution into another (in this case, \( g \# \mu \) and the original).
- The above formulate for Wasserstein distance is the “dual form” given via “Kantorovich-Rubinstein duality”.
Majority vote

Green region is error of majority vote!
Suppose \( y_i \in \{-1, +1\} \).

\[
\text{MAJ}(y_1, \ldots, y_n) := \begin{cases} 
+1 & \text{when } \sum_i y_i \geq 0, \\
-1 & \text{when } \sum_i y_i < 0.
\end{cases}
\]

Error rate of majority classifier (with individual error probability \( p \)):

\[
\Pr[\text{Binom}(n, p) \geq n/2] = \sum_{i=n/2}^{n} \binom{n}{i} p^i (1-p)^{n-i} \leq \exp \left( -n \left( \frac{1}{2} - p \right)^2 \right).
\]
Green region is error of majority vote!
Suppose $y_i \in \{-1, +1\}$.

$$\text{MAJ}(y_1, \ldots, y_n) := \begin{cases} +1 & \text{when } \sum_i y_i \geq 0, \\ -1 & \text{when } \sum_i y_i < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability $p$):

$$\Pr[\text{Binom}(n, p) \geq n/2] = \sum_{i=n/2}^{n} \binom{n}{i} p^i (1-p)^{n-i} \leq \exp \left(-n \left(1/2 - p \right)^2 \right).$$
Majority vote

**Green region** is error of majority vote!

Suppose $y_i \in \{-1, +1\}$.

$$\text{MAJ}(y_1, \ldots, y_n) := \begin{cases} +1 & \text{when } \sum y_i \geq 0, \\ -1 & \text{when } \sum y_i < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability $p$):

$$\Pr[\text{Binom}(n, p) \geq n/2] = \sum_{i=n/2}^{n} \binom{n}{i} p^i (1-p)^{n-i} \leq \exp \left( -n(1/2 - p)^2 \right).$$
Majority vote

Green region is error of majority vote!
Suppose $y_i \in \{-1, +1\}$.

$$\text{MAJ}(y_1, \ldots, y_n) := \begin{cases} +1 & \text{when } \sum_i y_i \geq 0, \\ -1 & \text{when } \sum_i y_i < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability $p$):

$$\Pr[\text{Binom}(n, p) \geq n/2] = \sum_{i=n/2}^{n} \binom{n}{i} p^i (1-p)^{n-i} \leq \exp \left( -n \left( \frac{1}{2} - p \right)^2 \right).$$
Majority vote

Green region is error of majority vote!
Suppose $y_i \in \{-1, +1\}$.

$$\text{MAJ}(y_1, \ldots, y_n) := \begin{cases} +1 & \text{when } \sum y_i \geq 0, \\ -1 & \text{when } \sum y_i < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability $p$):

$$\Pr[\text{Binom}(n, p) \geq n/2] = \sum_{i=n/2}^{n} \binom{n}{i} p^i (1-p)^{n-i} \leq \exp \left(-n \left(\frac{1}{2} - p\right)^2\right).$$
**Green region** is error of majority vote!

Suppose $y_i \in \{-1, +1\}$.

$$\text{MAJ}(y_1, \ldots, y_n) := \begin{cases} +1 & \text{when } \sum_i y_i \geq 0, \\ -1 & \text{when } \sum_i y_i < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability $p$):

$$\Pr[\text{Binom}(n, p) \geq n/2] = \sum_{i=n/2}^{n} \binom{n}{i} p^i (1-p)^{n-i} \leq \exp \left( -n \left( \frac{1}{2} - p \right)^2 \right).$$
Bottom line

Green region is error of majority vote!
**Green region** is error of majority vote!

Error of majority vote classifier goes down **exponentially** in $n$.

$$\Pr[\text{Binom}(n, p) \geq n/2] = \sum_{i=n/2}^{n} \binom{n}{i} p^i (1-p)^{n-i} \leq \exp\left(-n(1/2 - p)^2\right).$$
From independent errors to an algorithm

How to use independent errors in an algorithm?

1. For $t = 1, 2, \ldots, T$:
   1.1 Obtain IID data $S_t := ((x_i^{(t)}, y_i^{(t)}))_{i=1}^n$,
   1.2 Train classifier $f_t$ on $S_t$.

2. Output $x \mapsto \text{MAJ} \left( f_1(x), \ldots, f_T(x) \right)$. 

▶ Good news: errors are independent! (Our exponential error estimate from before is valid.)

▶ Bad news: classifiers trained on $1/T$ fraction of data (why not just train ResNet on all of it...).
From independent errors to an algorithm

How to use independent errors in an algorithm?

1. For $t = 1, 2, \ldots, T$:
   
   1.1 Obtain IID data $S_t := ((x_{i}^{(t)}, y_{i}^{(t)}))_{i=1}^{n}$,
   
   1.2 Train classifier $f_t$ on $S_t$.

2. Output $\mathbf{x} \mapsto \text{MAJ} (f_1(\mathbf{x}), \ldots, f_T(\mathbf{x}))$.

▶ **Good news:** errors are independent!
   (Our exponential error estimate from before is valid.)

▶ **Bad news:** classifiers trained on $1/T$ fraction of data
   (why not just train ResNet on all of it...).
From independent errors to an algorithm

How to use independent errors in an algorithm?

1. For $t = 1, 2, \ldots, T$:
   
   1.1 Obtain IID data $S_t := \{(x_i^{(t)}, y_i^{(t)})\}_{i=1}^n$,
   
   1.2 Train classifier $f_t$ on $S_t$.

2. Output $x \mapsto \text{MAJ}(f_1(x), \ldots, f_T(x))$. 
How to use independent errors in an algorithm?

1. For $t = 1, 2, \ldots, T$:
   
   1.1 Obtain IID data $S_t := ((x_i^{(t)}, y_i^{(t)}))_{i=1}^n$,
   
   1.2 Train classifier $f_t$ on $S_t$.

2. Output $x \mapsto \text{MAJ}(f_1(x), \ldots, f_T(x))$.

- **Good news**: errors are independent!
  (Our exponential error estimate from before is valid.)

- **Bad news**: classifiers trained on $1/T$ fraction of data
  (why not just train ResNet on all of it...).
Bagging = Bootstrap aggregating (Leo Breiman, 1994).

1. Obtain IID data $S := ((x_i, y_i))_{i=1}^n$.

2. For $t = 1, 2, \ldots, T$:
   2.1 Resample $n$ points uniformly at random with replacement from $S$, obtaining “Bootstrap sample” $S_t$.
   2.2 Train classifier $f_t$ on $S_t$.

3. Output $x \mapsto \text{MAJ}(f_1(x), \ldots, f_T(x))$. 
Bagging = Bootstrap aggregating (Leo Breiman, 1994).

1. Obtain IID data $S := \{(x_i, y_i)\}_{i=1}^n$.
2. For $t = 1, 2, \ldots, T$:
   2.1 Resample $n$ points uniformly at random with replacement from $S$, obtaining "Bootstrap sample" $S_t$.
   2.2 Train classifier $f_t$ on $S_t$.
3. Output $x \mapsto \text{MAJ}(f_1(x), \ldots, f_T(x))$.

▶ **Good news:** using most of the data for each $f_t$!
▶ **Bad news:** errors no longer independent…?
Sampling with replacement?

Question:
Take $n$ samples uniformly at random with replacement from a population of size $n$. What is the probability that a given individual is not picked?
Sampling with replacement?

**Question:**

*Take* $n$ *samples uniformly at random with replacement from* a population of size $n$. *What is the probability that a given individual is not picked?*

**Answer:**  

$$
\left(1 - \frac{1}{n}\right)^n ; \text{ for large } n: \lim_{n \to \infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e} \approx 0.3679.
$$

Implications for bagging:

▶ Each bootstrap sample contains about 63% of the data set.
▶ Remaining 37% can be used to estimate error rate of classifier trained on the bootstrap sample.
▶ If we have three classifiers, some of their error estimates must share examples! Independence is violated!
Sampling with replacement?

Question:
Take \( n \) samples uniformly at random with replacement from a population of size \( n \). What is the probability that a given individual is not picked?

Answer: \( \left( 1 - \frac{1}{n} \right)^n \); for large \( n \): \( \lim_{n \to \infty} \left( 1 - \frac{1}{n} \right)^n = \frac{1}{e} \approx 0.3679 \).

Implications for bagging:

- Each bootstrap sample contains about 63\% of the data set.
- Remaining 37\% can be used to estimate error rate of classifier trained on the bootstrap sample.
- If we have three classifiers, some of their error estimates must share examples! Independence is violated!
Boosting overview

- We no longer assume classifiers have independent errors.
- We no longer output a simple majority: we *reweight* the classifiers via optimization.
- There is a rich theory with many interpretations.
Simplified boosting scheme

1. Start with data \( ((x_i, y_i))_{i=1}^n \) and classifiers \( (h_1, \ldots, h_T) \).

2. Find weights \( w \in \mathbb{R}^T \) which approximately minimize

\[
\frac{1}{n} \sum_{i=1}^n \ell \left( y_i \sum_{j=1}^T w_j h_j(x_i) \right) = \frac{1}{n} \sum_{i=1}^n \ell (y_i w^T z_i),
\]

where \( z_i = (h_1(x_i), \ldots, h_T(x_i)) \in \mathbb{R}^T \).

(We use classifiers to give us features.)

3. Predict with \( x \mapsto \sum_{j=1}^T w_j h_j(x) \).

Remarks.

▶ If \( \ell \) is convex, this is standard linear prediction: convex in \( w \).

▶ In the classical setting: \( \ell(r) = \exp(-r) \), optimizer = coordinate descent, \( T = \infty \).

▶ Most commonly, \( (h_1, \ldots, h_T) \) are decision stumps.

▶ Popular software implementation: xgboost.
Simplified boosting scheme

1. Start with data \( ((x_i, y_i)_{i=1}^n \) and classifiers \( (h_1, \ldots, h_T) \).
2. Find weights \( w \in \mathbb{R}^T \) which approximately minimize

\[
\frac{1}{n} \sum_{i=1}^n \ell \left( y_i \sum_{j=1}^T w_j h_j(x_i) \right) = \frac{1}{n} \sum_{i=1}^n \ell (y_i w^T z_i),
\]

where \( z_i = (h_1(x_i), \ldots, h_T(x_i)) \in \mathbb{R}^T \).

(We use classifiers to give us features.)

3. Predict with \( x \mapsto \sum_{j=1}^T w_j h_j(x) \).

Remarks.

- If \( \ell \) is convex, this is standard linear prediction: convex in \( w \).
- In the classical setting:
  \( \ell(r) = \exp(-r) \), optimizer = coordinate descent, \( T = \infty \).
- Most commonly, \( (h_1, \ldots, h_T) \) are decision stumps.
- Popular software implementation: xgboost.
Classifying irises by sepal and petal measurements

- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
- $x_1 =$ ratio of sepal length to width
- $x_2 =$ ratio of petal length to width
Classifying irises by sepal and petal measurements

- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
- $x_1 =$ ratio of sepal length to width
- $x_2 =$ ratio of petal length to width

$\hat{y} = 2$
Classifying irises by sepal and petal measurements

- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
- $x_1 =$ ratio of sepal length to width
- $x_2 =$ ratio of petal length to width

$x_1 > 1.7$
Decision stumps?

Classifying irises by sepal and petal measurements

- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
- $x_1 =$ ratio of sepal length to width
- $x_2 =$ ratio of petal length to width
Decision stumps?

Classifying irises by sepal and petal measurements

- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
- $x_1 =$ ratio of sepal length to width
- $x_2 =$ ratio of petal length to width

\[ x_1 > 1.7 \]

$\hat{y} = 1$ \quad $\hat{y} = 3$

...and stop there!
Minimizing \( \frac{1}{n} \sum_{i=1}^{n} \ell \left( y_i \sum_{j=1}^{T} w_j h_j(x_i) \right) \) over \( w \in \mathbb{R}^T \), where \((h_1, \ldots, h_T)\) are decision stumps.
Boosting decision stumps

Minimizing \( \frac{1}{n} \sum_{i=1}^{n} \ell \left( y_{i} \sum_{j=1}^{T} w_{j} h_{j}(x_{i}) \right) \) over \( w \in \mathbb{R}^{T} \), where \((h_{1}, \ldots, h_{T})\) are decision stumps.

Boosted stumps. 
(\(\mathcal{O}(n)\) param.)

2-layer ReLU. 
(\(\mathcal{O}(n)\) param.)

3-layer ReLU. 
(\(\mathcal{O}(n)\) param.)
Minimizing $\frac{1}{n} \sum_{i=1}^{n} \ell \left( y_i \sum_{j=1}^{T} w_j h_j(x_i) \right)$ over $w \in \mathbb{R}^T$, where $(h_1, \ldots, h_T)$ are decision stumps.

Boosted stumps.  
($O(n)$ param.)

2-layer ReLU.  
($O(n)$ param.)

3-layer ReLU.  
($O(n)$ param.)
Boosting decision stumps

Minimizing \( \frac{1}{n} \sum_{i=1}^{n} \ell \left( y_i \sum_{j=1}^{T} w_j h_j(x_i) \right) \) over \( w \in \mathbb{R}^T \), where \((h_1, \ldots, h_T)\) are decision stumps.

Boosted stumps. 
(\(O(n)\) param.)

2-layer ReLU. 
(\(O(n)\) param.)

3-layer ReLU. 
(\(O(n)\) param.)
Boosting **decision stumps**

Minimizing $\frac{1}{n} \sum_{i=1}^{n} \ell \left( y_i \sum_{j=1}^{T} w_j h_j(x_i) \right)$ over $w \in \mathbb{R}^T$, where $(h_1, \ldots, h_T)$ are decision stumps.

- **Boosted stumps.**
  $(O(n) \text{ param.})$

- **2-layer ReLU.**
  $(O(n) \text{ param.})$

- **3-layer ReLU.**
  $(O(n) \text{ param.})$
Selected homework concepts
Please go through all mathy homework problems (except for the extra credit in hw6).