Computational Intelligence Lab

Cristian Perez Jensen

July 16, 2024

Note that these are not the official lecture notes of the course, but only notes written by a student of the course. As such, there might be mistakes. The source code can be found at `github.com/cristianpjensen/eth-cs-notes`. If you find a mistake, please create an issue or open a pull request.

Contents

List of symbols

$\doteq$	Equality by definition
$\approx$	Approximate equality
$\propto$	Proportional to
$\mathbb{N}$	Set of natural numbers
$\mathbb{R}$	Set of real numbers
$i : j$	Set of natural numbers between i and j. *I.e.*, $\{i, i+1, \dots, j\}$
$f : A \to B$	Function f that maps elements of set A to elements of set B
$\mathbb{1}\{\text{predicate}\}$	Indicator function (1 if predicate is true, otherwise 0)

$\boldsymbol{v} \in \mathbb{R}^n$	n-dimensional vector
$\boldsymbol{M} \in \mathbb{R}^{m \times n}$	$m \times n$ matrix
$\mathbf{T} \in \mathbb{R}^{d_1 \times \cdots \times d_n}$	Tensor
$\boldsymbol{M}^\top$	Transpose of matrix $\boldsymbol{M}$
$\boldsymbol{M}^{-1}$	Inverse of matrix $\boldsymbol{M}$
$\det(\boldsymbol{M})$	Determinant of $\boldsymbol{M}$

$\frac{\mathrm{d}}{\mathrm{d}x} f(x)$	Ordinary derivative of $f(x)$ w.r.t. x at point $x \in \mathbb{R}$
$\frac{\partial}{\partial x} f(\boldsymbol{x})$	Partial derivative of $f(\boldsymbol{x})$ w.r.t. x at point $\boldsymbol{x} \in \mathbb{R}^n$
$\boldsymbol{\nabla}_{\boldsymbol{x}} f(\boldsymbol{x}) \in \mathbb{R}^n$	Gradient of $f : \mathbb{R}^n \to \mathbb{R}$ at point $\boldsymbol{x} \in \mathbb{R}^n$
$\boldsymbol{\nabla}^2_{\boldsymbol{x}} f(\boldsymbol{x}) \in \mathbb{R}^{n \times n}$	Hessian of $f : \mathbb{R}^n \to \mathbb{R}$ at point $\boldsymbol{x} \in \mathbb{R}^n$

$\boldsymbol{\theta} \in \Theta$	Parametrization of a model, where Θ is a compact subset of $\mathbb{R}^K$
$\mathcal{X}$	Input space
$\mathcal{Y}$	Output space
$\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$	Labeled training data

1 Dimensionality reduction

The motivation behind dimensionality reduction is to find a low-dimensional representation of high-dimensional data.[1] This can be split into two goals: (1) compressing the data, while preserving as much as possible of the relevant information, and (2) interpreting the data in low dimensionality is easier than high dimensionality.

[1] Often, the original raw representation is high-dimensional and redundant, *e.g.*, images, audio, time series.

Dimensionality reduction is often performed by an autoencoder, which typically has a bottleneck and aims to predict its input; see Figure 1. In general, we have an encoder F and a decoder G,

$$F : \mathbb{R}^n \to \mathbb{R}^m, \quad G : \mathbb{R}^m \to \mathbb{R}^n.$$

$m \ll n$.

The reconstruction function is then the following function,

$$G \circ F : \mathbb{R}^n \to \mathbb{R}^n,$$

which is ideally the identity function.

1.1 Linear autoencoders

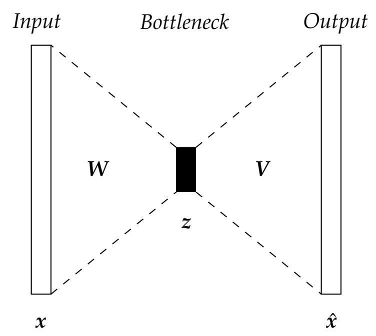

Figure 1. Diagram of a single layer linear autoencoder.

To build a nice theory, we will only consider a single layer linear autoencoder, which means that we have the following functions,

$$\begin{aligned} F &: \boldsymbol{x} \mapsto \boldsymbol{z} = \boldsymbol{W}\boldsymbol{x}, \quad \boldsymbol{W} \in \mathbb{R}^{m \times n}. \\ G &: \boldsymbol{z} \mapsto \hat{\boldsymbol{x}} = \boldsymbol{V}\boldsymbol{z}, \quad \boldsymbol{V} \in \mathbb{R}^{n \times m}. \end{aligned}$$

The objective to minimize of the linear encoder is the following,

$$\mathcal{R}(\boldsymbol{W}, \boldsymbol{V}) = \mathcal{R}(\boldsymbol{P} \doteq \boldsymbol{V}\boldsymbol{W}) \doteq \mathbb{E}\left[\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{P}\boldsymbol{x}\|^2\right].$$

Corollary. For centered data, *i.e.* $E[\boldsymbol{x}] = 0$, optimal affine maps degenerate to linear ones.

Proof. Proof by contradiction. Let $\boldsymbol{a} \neq \boldsymbol{0}$, then

$$\begin{aligned} \mathbb{E}\left[\|\boldsymbol{x} - (\boldsymbol{P}\boldsymbol{x} + \boldsymbol{a})\|^2\right] &= \mathbb{E}[\langle \boldsymbol{x} - \boldsymbol{P}\boldsymbol{x} - \boldsymbol{a}, \boldsymbol{x} - \boldsymbol{P}\boldsymbol{x} - \boldsymbol{a}\rangle] \\ &= \mathbb{E}[\langle \boldsymbol{x} - \boldsymbol{P}\boldsymbol{x}, \boldsymbol{x} - \boldsymbol{P}\boldsymbol{x}\rangle + 2\langle \boldsymbol{x} - \boldsymbol{P}\boldsymbol{x}, -\boldsymbol{a}\rangle + \langle -\boldsymbol{a}, -\boldsymbol{a}\rangle] \\ &= \mathbb{E}\left[\|\boldsymbol{x} - \boldsymbol{P}\boldsymbol{x}\|^2\right] - 2\langle \underbrace{\mathbb{E}[\boldsymbol{x}] - \boldsymbol{P}\mathbb{E}[\boldsymbol{x}]}_{=\boldsymbol{0}}, \boldsymbol{a}\rangle + \underbrace{\|\boldsymbol{a}\|^2}_{>0} \\ &> \mathbb{E}\left[\|\boldsymbol{x} - \boldsymbol{P}\boldsymbol{x}\|^2\right]. \end{aligned}$$

Thus, the risk is strictly worse if $\boldsymbol{a} \neq \boldsymbol{0}$. ■

Thus, we will assume that the data is centered, which makes the analysis easier, since we do not need to consider the affine case.

Note that while the optimal linear reconstruction map $\boldsymbol{P}$ is unique, its parametrization $\boldsymbol{W}, \boldsymbol{V}$ is not unique, since for any invertible matrix $\boldsymbol{A} \in \mathbb{R}^{m \times m}$, we can construct an optimal parametrization,

$$\boldsymbol{V}\boldsymbol{W} = \boldsymbol{V}\boldsymbol{I}\boldsymbol{W} = \boldsymbol{V}(\boldsymbol{A}\boldsymbol{A}^{-1})\boldsymbol{W} = (\boldsymbol{V}\boldsymbol{A})(\boldsymbol{A}^{-1}\boldsymbol{W}),$$

with $\boldsymbol{A}^{-1}\boldsymbol{W}, \boldsymbol{V}\boldsymbol{A}$. The weight matrices are non-identifiable.

Since $\boldsymbol{P}$ cannot be any $n \times n$ matrix, we want to know how the composition of $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times m}$ characterizes the matrix $\boldsymbol{P}$ and which constraints they impose. The answer to this is that the weight matrices impose a rank constraint on $\boldsymbol{P}$,

$$\operatorname{rank}(\boldsymbol{P}) = \min\{\operatorname{rank}(\boldsymbol{W}), \operatorname{rank}(\boldsymbol{W})\} \leq \min\{m, n\} = m.$$

Thus, when optimizing for $\boldsymbol{P}$, we are constrained to matrices with rank less or equal to m.

1.2 Projection

The rank constraint and linearity of $\boldsymbol{P}$ means that the image (column space) of $\boldsymbol{P}$ is a linear subspace $\mathcal{U} \subseteq \mathbb{R}^n$ of dimension at most m. We will break the solution to our problem into two parts: (1) finding the optimal subspace $\mathcal{U}$, and (2) finding the optimal mapping to that subspace.[2]

[2] We do not search for the weight matrices $\boldsymbol{W}, \boldsymbol{V}$, since they are not unique, but $\boldsymbol{P}$ is unique.

Finding the optimal mapping to a subspace. We will first focus on (2); given subspace $\mathcal{U}$, we need to determine the optimal linear map $\boldsymbol{P}^\star$, such that

$$\boldsymbol{P}^\star = \operatorname*{argmin}_{\boldsymbol{P}} \|\boldsymbol{x} - \boldsymbol{P}\boldsymbol{x}\|^2, \quad \operatorname{col}(\boldsymbol{P}) = \mathcal{U}.$$

Definition 1 (Orthogonal projection)**.** A linear transformation $P : \mathcal{V} \to \mathcal{V}$ is called an orthogonal projection onto $\mathcal{U}$ if $\forall \boldsymbol{x} \in \mathcal{V}$:

1. *Projection*: $P(\boldsymbol{x}) \in \mathcal{U}$;
2. *Orthogonality*: $\operatorname{null}(P) \perp \operatorname{col}(P)$, which is equivalent to the following holding, $\langle P(\boldsymbol{x}), \boldsymbol{y} \rangle = \langle \boldsymbol{x}, P(\boldsymbol{y}) \rangle$ (self-adjointness);
3. *Idempotency*: $P(P(\boldsymbol{x})) = P(\boldsymbol{x})$.

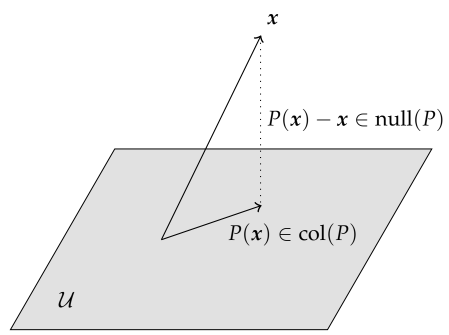

Figure 2. Orthogonal projection of $\boldsymbol{x}$ onto subspace plane $\mathcal{U}$.

Definition 2. The orthogonal projection to a linear subspace $\mathcal{U} \subseteq \mathbb{R}^n$ is defined as

$$\Pi_{\mathcal{U}} : \mathbb{R}^n \to \mathcal{U}, \quad \Pi_{\mathcal{U}}(\boldsymbol{x}) = \operatorname*{argmin}_{\boldsymbol{x}' \in \mathcal{U}} \|\boldsymbol{x} - \boldsymbol{x}'\|.$$

Proof. We need to show that the definition of $\Pi_{\mathcal{U}}$ indeed is an orthogonal projection by showing that it adheres to the properties of Definition 1.

1. *Projection*: This is true by definition of the values that the argmin are allowed to take on;
2. *Idempotency*: For all $\boldsymbol{u} \in \mathcal{U}$, $\Pi_{\mathcal{U}}(\boldsymbol{u}) = \operatorname{argmin}_{\boldsymbol{x}' \in \mathcal{U}} \|\boldsymbol{u} - \boldsymbol{x}'\| = \boldsymbol{u}$. Thus, $\Pi_{\mathcal{U}} = \Pi_{\mathcal{U}} \circ \Pi_{\mathcal{U}}$;
3. *Orthogonality*: We need to show that $\Pi_{\mathcal{U}}(\boldsymbol{x}) - \boldsymbol{x} \in \mathcal{U}^{\perp}$. Decompose it into $\boldsymbol{u} \in \mathcal{U}$ and $\boldsymbol{u}^{\perp} \in \mathcal{U}^{\perp}$ by $\Pi_{\mathcal{U}}(\boldsymbol{x}) - \boldsymbol{x} = \boldsymbol{u} + \boldsymbol{u}^{\perp}$. Then, we only need to show that $\boldsymbol{u} = \mathbf{0}$.

   Proof by contradiction. Let $\boldsymbol{u} \neq \mathbf{0}$, then

$$\begin{aligned}
\|\Pi_{\mathcal{U}}(\boldsymbol{x}) - \boldsymbol{x}\|^2 &= \underbrace{\|\boldsymbol{u}\|^2}_{>0} + \|\boldsymbol{u}^{\perp}\|^2 + 2\underbrace{\langle \boldsymbol{u}, \boldsymbol{u}^{\perp} \rangle}_{=0} \\
&> \|\boldsymbol{u}^{\perp}\|^2 \\
&= \|\underbrace{\Pi_{\mathcal{U}}(\boldsymbol{x}) - \boldsymbol{u}}_{\in \mathcal{U}} - \boldsymbol{x}\|^2,
\end{aligned}$$

   which contradicts with

$$\|\Pi_{\mathcal{U}}(\boldsymbol{x}) - \boldsymbol{x}\| = \min_{\boldsymbol{x}' \in \mathcal{U}} \|\boldsymbol{x}' - \boldsymbol{x}\| \leq \|\tilde{\boldsymbol{u}} - \boldsymbol{x}\|, \quad \forall \tilde{\boldsymbol{u}} \in \mathcal{U}.$$

   Hence, $\Pi_{\mathcal{U}}(\boldsymbol{x}) - \boldsymbol{x} \in \mathcal{U}^{\perp}$ and is unique;
4. *Linearity*: We need to show homogeneity,

$$\begin{aligned}
\Pi_{\mathcal{U}}(\alpha \boldsymbol{x}) &= \operatorname*{argmin}_{\boldsymbol{x}'' \in \mathcal{U}} \|\alpha \boldsymbol{x} - \boldsymbol{x}''\| \\
&= \operatorname*{argmin}_{\alpha \boldsymbol{x}' \in \mathcal{U}} \|\alpha \boldsymbol{x} - \alpha \boldsymbol{x}'\| && \boldsymbol{x}' = \tfrac{1}{\alpha} \boldsymbol{x}''. \\
&= \alpha \operatorname*{argmin}_{\boldsymbol{x}' \in \mathcal{U}} |\alpha| \|\boldsymbol{x} - \boldsymbol{x}'\| && \operatorname{argmin}_{\lambda x} f(x) = \lambda \operatorname{argmin}_x f(x). \\
&= \alpha \operatorname*{argmin}_{\boldsymbol{x}' \in \mathcal{U}} \|\boldsymbol{x} - \boldsymbol{x}'\| \\
&= \alpha \Pi_{\mathcal{U}}(\boldsymbol{x}).
\end{aligned}$$

   And, we need to show additivity,

$$\begin{aligned}
\Pi_{\mathcal{U}}(\boldsymbol{x} + \boldsymbol{y}) &= \operatorname*{argmin}_{\boldsymbol{z} \in \mathcal{U}} \|\boldsymbol{z} - (\boldsymbol{x} + \boldsymbol{y})\| \\
&= \operatorname*{argmin}_{\Pi_{\mathcal{U}}(\boldsymbol{x}) + \boldsymbol{y}' \in \mathcal{U}} \|\Pi_{\mathcal{U}}(\boldsymbol{x}) + \boldsymbol{y}' - \boldsymbol{x} - \boldsymbol{y}\| && \boldsymbol{z} \doteq \Pi_{\mathcal{U}}(\boldsymbol{x}) + \boldsymbol{y}' \text{ for some } \boldsymbol{y}' \in \mathcal{U}. \\
&= \Pi_{\mathcal{U}}(\boldsymbol{x}) + \operatorname*{argmin}_{\boldsymbol{y}' \in \mathcal{U}} \|(\boldsymbol{y}' - \boldsymbol{y}) + (\Pi_{\mathcal{U}}(\boldsymbol{x}) - \boldsymbol{x})\|^2 && \operatorname{argmin}_{x+\lambda} f(x) = \lambda + \operatorname{argmin}_x f(x). \\
&= \Pi_{\mathcal{U}}(\boldsymbol{x}) + \operatorname*{argmin}_{\boldsymbol{y}' \in \mathcal{U}} \|\boldsymbol{y}' - \boldsymbol{y}\|^2 + \|\Pi_{\mathcal{U}}(\boldsymbol{x}) - \boldsymbol{x}\|^2 + 2\langle \Pi_{\mathcal{U}}(\boldsymbol{x}) - \boldsymbol{x}, \boldsymbol{y}' - \boldsymbol{y} \rangle \\
&= \Pi_{\mathcal{U}}(\boldsymbol{x}) + \operatorname*{argmin}_{\boldsymbol{y}' \in \mathcal{U}} \|\boldsymbol{y}' - \boldsymbol{y}\|^2 + 2\underbrace{\langle \Pi_{\mathcal{U}}(\boldsymbol{x}) - \boldsymbol{x}, \boldsymbol{y}' \rangle}_{\langle \mathcal{U}^{\perp}, \mathcal{U} \rangle = 0} && \langle \Pi_{\mathcal{U}}(\boldsymbol{x}) - \boldsymbol{x}, \boldsymbol{y}' - \boldsymbol{y} \rangle = \langle \Pi_{\mathcal{U}}(\boldsymbol{x}) - \boldsymbol{x}, \boldsymbol{y}' \rangle - \boldsymbol{y}. \\
&= \Pi_{\mathcal{U}}(\boldsymbol{x}) + \Pi_{\mathcal{U}}(\boldsymbol{y}).
\end{aligned}$$

■

So, we know that $\Pi_{\mathcal{U}}$ is an orthogonal projection. Now, we want to find the matrix $\boldsymbol{P}$ representing that linear transformation.

Given an orthonormal basis $\boldsymbol{U}$ of $\mathcal{U}$, we can compute the optimal projection matrix,

$$\boldsymbol{P} = \boldsymbol{U}\boldsymbol{U}^\top.$$

Note that in this case $\boldsymbol{W}$ and $\boldsymbol{V}$ share parameters, and $\boldsymbol{U}\boldsymbol{U}^\top$ is the optimal weight matrix if we enforce parameter sharing via $\boldsymbol{V} = \boldsymbol{W}^\top$.

Proof. The image of the projection matrix is $\mathcal{U}$,

$$\boldsymbol{P}\boldsymbol{x} = \left(\sum_{i=1}^{m} \boldsymbol{u}_i \boldsymbol{u}_i^\top\right)\boldsymbol{x} = \sum_{i=1}^{m} \boldsymbol{u}_i \boldsymbol{u}_i^\top \boldsymbol{x} = \sum_{i=1}^{m} \langle \boldsymbol{u}_i, \boldsymbol{x}\rangle \boldsymbol{u}_i.$$

Self-adjointness,

$$\boldsymbol{P}^\top = (\boldsymbol{U}\boldsymbol{U}^\top)^\top = \boldsymbol{U}\boldsymbol{U}^\top = \boldsymbol{P}.$$

Idempotency,

$$\boldsymbol{P}\boldsymbol{P} = \boldsymbol{U}\underbrace{\boldsymbol{U}^\top\boldsymbol{U}}_{\boldsymbol{I}_m}\boldsymbol{U}^\top = \boldsymbol{U}\boldsymbol{U}^\top = \boldsymbol{P}.$$

Orthogonality, TODO ■

In general, we do not have an orthonormal basis for $\mathcal{U}$. In a non-orthonormal basis $\boldsymbol{V}$ for $\mathcal{U}$, we can recover the projection matrix,

$$\boldsymbol{P} = \boldsymbol{V}\boldsymbol{V}^+, \quad \boldsymbol{V}^+ \doteq \left(\boldsymbol{V}^\top\boldsymbol{V}\right)^{-1}\boldsymbol{V}^\top.$$

Note that $\boldsymbol{V}^+$ is the left Moore-Penrose pseudo-inverse of $\boldsymbol{V}$.

Proof. $\boldsymbol{P}$ is the projection matrix of $\mathcal{U}$,

$$\boldsymbol{P}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{V}^+\boldsymbol{V} = \boldsymbol{V}\left(\boldsymbol{V}^\top\boldsymbol{V}\right)^{-1}\boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{V}.$$

Together with the rank constraint, this yields $\boldsymbol{P}\boldsymbol{u}^\perp = \boldsymbol{0}$ for all $\boldsymbol{u}^\perp \in \mathcal{U}^\perp$.

Self-adjointness,

$$\boldsymbol{P}^\top = \left(\boldsymbol{V}\left(\boldsymbol{V}^\top\boldsymbol{V}\right)^{-1}\boldsymbol{V}^\top\right)^\top = \boldsymbol{V}\left(\boldsymbol{V}^\top\boldsymbol{V}\right)^{-1}\boldsymbol{V}^\top = \boldsymbol{P}.$$

Idempotency,

$$\boldsymbol{P}\boldsymbol{P} = \boldsymbol{V}\left(\boldsymbol{V}^\top\boldsymbol{V}\right)^{-1}\boldsymbol{V}^\top\boldsymbol{V}\left(\boldsymbol{V}^\top\boldsymbol{V}\right)^{-1}\boldsymbol{V}^\top = \boldsymbol{V}\boldsymbol{V}^+ = \boldsymbol{P}.$$

■

Finding the optimal subspace. Now we need to find out which subspace of dimension m or less is optimal to project onto. First, we need to rewrite the objective function to find a new interpretation,

$$\begin{aligned}
\mathcal{R}(\boldsymbol{P}) &= \frac{1}{2}\mathbb{E}\left[\|\boldsymbol{x} - \boldsymbol{P}\boldsymbol{x}\|^2\right] \\
&= \frac{1}{2}\mathbb{E}[\langle \boldsymbol{x}, \boldsymbol{x}\rangle + 2\langle \boldsymbol{x}, -\boldsymbol{P}\boldsymbol{x}\rangle + \langle -\boldsymbol{P}\boldsymbol{x}, -\boldsymbol{P}\boldsymbol{x}\rangle] \\
&= \frac{1}{2}\mathbb{E}\left[\|\boldsymbol{x}\|^2\right] + \frac{1}{2}\mathbb{E}\left[\|\boldsymbol{P}\boldsymbol{x}\|^2\right] - \mathbb{E}[\langle \boldsymbol{x}, \boldsymbol{P}\boldsymbol{x}\rangle] \\
&= \frac{1}{2}\mathbb{E}\left[\|\boldsymbol{x}\|^2\right] + \frac{1}{2}\mathbb{E}\left[\|\boldsymbol{P}\boldsymbol{x}\|^2\right] - \mathbb{E}[\langle \boldsymbol{x}, \boldsymbol{P}^2\boldsymbol{x}\rangle] \\
&= \frac{1}{2}\mathbb{E}\left[\|\boldsymbol{x}\|^2\right] + \frac{1}{2}\mathbb{E}\left[\|\boldsymbol{P}\boldsymbol{x}\|^2\right] - \mathbb{E}\left[\|\boldsymbol{P}\boldsymbol{x}\|^2\right] \\
&= \frac{1}{2}\mathbb{E}\left[\|\boldsymbol{x}\|^2\right] - \frac{1}{2}\mathbb{E}\left[\|\boldsymbol{P}\boldsymbol{x}\|^2\right].
\end{aligned}$$

Idempotency of projection matrices.

$\langle \boldsymbol{x}, \boldsymbol{P}^2\boldsymbol{x}\rangle = \langle \boldsymbol{P}\boldsymbol{x}, \boldsymbol{P}\boldsymbol{x}\rangle$.

Because our data is centered, we know the following,

$$\begin{aligned}
\mathrm{Var}[\boldsymbol{x}] &= \mathbb{E}\left[\|\boldsymbol{x}\|^2\right] - \|\underbrace{\mathbb{E}[\boldsymbol{x}]}_{=\boldsymbol{0}}\|^2 \\
&= \mathbb{E}\left[\|\boldsymbol{x}\|^2\right]. \\
\mathrm{Var}[\boldsymbol{P}\boldsymbol{x}] &= \mathbb{E}\left[\|\boldsymbol{P}\boldsymbol{x}\|^2\right] - \|\mathbb{E}[\boldsymbol{P}\boldsymbol{x}]\|^2 \\
&= \mathbb{E}\left[\|\boldsymbol{P}\boldsymbol{x}\|^2\right] - \|\boldsymbol{P}\underbrace{\mathbb{E}[\boldsymbol{x}]}_{=\boldsymbol{0}}\|^2 \\
&= \mathbb{E}\left[\|\boldsymbol{P}\boldsymbol{x}\|^2\right].
\end{aligned}$$

$$\mathcal{R}(\boldsymbol{P}) = \frac{1}{2}(\mathrm{Var}[\boldsymbol{x}] - \mathrm{Var}[\boldsymbol{P}\boldsymbol{x}]) \propto -\frac{1}{2}\mathrm{Var}[\boldsymbol{P}\boldsymbol{x}].$$

Hence, minimizing $\mathcal{R}(\boldsymbol{P})$ is equivalent to maximizing the variance $\mathrm{Var}[\boldsymbol{P}\boldsymbol{x}]$.

We can further simplify this expression to find a sufficient statistic for the objective function,

$$\begin{aligned}
-\frac{1}{2}\mathrm{Var}[\boldsymbol{P}\boldsymbol{x}] &= -\frac{1}{2}\mathbb{E}\left[\|\boldsymbol{P}\boldsymbol{x}\|^2\right] \\
&= -\frac{1}{2}\mathbb{E}[\langle \boldsymbol{x}, \boldsymbol{P}\boldsymbol{x}\rangle] \\
&= -\frac{1}{2}\mathbb{E}\left[\mathrm{tr}\left(\boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x}\right)\right] \\
&= -\frac{1}{2}\mathrm{tr}\left(\mathbb{E}\left[\boldsymbol{P}\boldsymbol{x}\boldsymbol{x}^\top\right]\right) \\
&= -\frac{1}{2}\mathrm{tr}\left(\boldsymbol{P}\mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^\top\right]\right).
\end{aligned}$$

$\|\boldsymbol{P}\boldsymbol{x}\|^2 = \langle \boldsymbol{P}\boldsymbol{x}, \boldsymbol{P}\boldsymbol{x}\rangle = \langle \boldsymbol{x}, \boldsymbol{P}^2\boldsymbol{x}\rangle = \langle \boldsymbol{x}, \boldsymbol{P}\boldsymbol{x}\rangle$.

Cyclic property of trace.

$\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top]$ is a sufficient statistic for $\mathcal{R}(\boldsymbol{P})$.

The optimal projection is fully determined by the covariance matrix $\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top]$, together with $\mathbb{E}[\boldsymbol{x}]$ for centering.

1.3 *Principal component analysis*

Theorem 3 (Spectral theorem)**.** Any symmetric and positive semidefinite matrix $\boldsymbol{\Sigma}$ can be non-negatively diagonalized with an orthogonal matrix,

$$\boldsymbol{\Sigma} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top, \quad \boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \dots, \lambda_n),$$

where $\lambda \geq \cdots \geq \lambda_n \geq 0$ and $\boldsymbol{Q}$ is orthogonal.

Remark. $\boldsymbol{Q}$ is composed of ordered eigenvectors of $\boldsymbol{\Sigma}$, and $\boldsymbol{\Lambda}$ is composed of ordered eigenvalues of $\boldsymbol{\Sigma}$.

Theorem 4 (PCA theorem)**.** The variance maximizing projection matrix $\boldsymbol{P}$ for a covariance matrix $\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top] = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top$ as in the spectral theorem is given by

$$\boldsymbol{P} = \boldsymbol{U}\boldsymbol{U}^\top, \quad \boldsymbol{U} = \boldsymbol{Q}\begin{bmatrix} \boldsymbol{I}_m \\ \boldsymbol{0} \end{bmatrix}.$$

Proof.

$$\begin{aligned} \mathrm{Var}[\boldsymbol{P}\boldsymbol{x}] &= \mathrm{tr}\Big(\boldsymbol{P}\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top]\Big) \\ &= \mathrm{tr}\Big(\boldsymbol{U}\boldsymbol{U}^\top\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top\Big) && \boldsymbol{P} = \boldsymbol{U}\boldsymbol{U}^\top,\ \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top] = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top. \\ &= \mathrm{tr}\Big(\Big(\boldsymbol{Q}^\top\boldsymbol{U}\Big)\Big(\boldsymbol{Q}^\top\boldsymbol{U}\Big)^\top\boldsymbol{\Lambda}\Big). && \text{Cyclic property.} \end{aligned}$$

This term is maximized by $\boldsymbol{Q}^\top\boldsymbol{U} = \begin{bmatrix} \boldsymbol{I}_m & \boldsymbol{0} \end{bmatrix}^\top$. ∎

1.4 *Learning algorithms*

Eigenvalue decomposition of the (symmetric) sample covariance matrix has $\mathcal{O}(n^3)$ complexity. Furthermore, the complexity of computing $\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top]$ is $\mathcal{O}(Nn^2)$.[3] This is quite costly, thus we need to search for algorithms that have lower runtime complexity.

[3] Typically, $N \gg n$.

Power method. The power method is a recursive algorithm for computing principal eigenvectors. It initializes a vector at random $\boldsymbol{v}^{(0)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Then, it iteratively improves this guess,

$$\boldsymbol{v}^{(t+1)} = \frac{\boldsymbol{A}\boldsymbol{v}^{(t)}}{\|\boldsymbol{A}\boldsymbol{v}^{(t)}\|}.$$

The computational complexity of this algorithm is $\mathcal{O}(Tn^2)$.

Lemma 5. Let $\boldsymbol{u}_1$ be the unique principal eigenvector of a diagonalizable matrix $\boldsymbol{A}$ with eigenvalues $\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. If $\langle v_0, \boldsymbol{u}_1 \rangle \neq 0$, then

$$\lim_{t\to\infty} \boldsymbol{v}^{(t)} = \boldsymbol{u}_1.$$

Proof. We can decompose vectors as a linear combination of eigenvectors, $\boldsymbol{v}^{(0)} = \sum_{i=1}^{n} \alpha_i \boldsymbol{u}_i$. Then,

$$\begin{aligned}
\boldsymbol{v}^{(k)} &\propto \boldsymbol{A}^k \boldsymbol{v}^{(0)} \\
&= \sum_{i=1}^{n} \alpha_i \lambda_i^k \boldsymbol{u}_i \\
&\propto \alpha_1 \boldsymbol{u}_1 + \sum_{i=2}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1}\right)^k \boldsymbol{u}_i.
\end{aligned}$$

$\boldsymbol{A}^k \boldsymbol{v}^{(0)} = \sum_{i=1}^{n} \alpha_i \boldsymbol{A}^k \boldsymbol{u}_i$.

Divide by λ_1.

$\lambda_i/\lambda_1 < 1$ for $i > 1$, thus the sum goes to 0 and $\boldsymbol{v}^{(k)} \to \boldsymbol{u}_1$. ■

We can use this algorithm to also compute the next principal eigenvectors by factoring out $\boldsymbol{u}_1$ and then doing the algorithm again to recover $\boldsymbol{u}_2$, and continue doing that until we have the m principal eigenvectors.

Thus, the total complexity of finding the m principal eigenvectors is $\mathcal{O}(Tmn^2)$. However, this does not get rid of the $\mathcal{O}(Nn^2)$ complexity for computing the sample covariance matrix.

Gradient descent. By treating the autoencoder as a neural network, we can use deep learning techniques, such as gradient descent. Gradient descent iteratively updates the weights by

$$\boldsymbol{P}^{(t+1)} = \boldsymbol{P}^{(t)} - \eta \boldsymbol{\nabla}_{\boldsymbol{P}} \mathcal{R}(\boldsymbol{P}).$$

The gradient is computed by $(\boldsymbol{P} - \boldsymbol{I})\boldsymbol{x}\boldsymbol{x}^\top$. The problem with this is that we cannot constrain $\boldsymbol{P}$ to be a projection. Thus, we actually need to update $\boldsymbol{V}$ and $\boldsymbol{W}$. Thus, by the chain rule for matrix derivatives,

$$\begin{aligned}
\boldsymbol{\nabla}_{\boldsymbol{W}} \mathcal{R}(\boldsymbol{W}, \boldsymbol{V}) &= (\boldsymbol{P} - \boldsymbol{I})\boldsymbol{x}\boldsymbol{x}^\top \boldsymbol{W}^\top \\
\boldsymbol{\nabla}_{\boldsymbol{V}} \mathcal{R}(\boldsymbol{W}, \boldsymbol{V}) &= \boldsymbol{V}^\top (\boldsymbol{P} - \boldsymbol{I})\boldsymbol{x}\boldsymbol{x}^\top.
\end{aligned}$$

The complexity for T iterations is then $\mathcal{O}(T(m+k)n^2)$, where k is the batch size.

1.5 Non-linear autoencoders

We can get much better performance by considering non-linearities, and we can easily use gradient descent to work with non-linear architectures.

2 Matrix completion

The goal of matrix completion is to fill in the missing entries of a sparse matrix $\boldsymbol{A}$. For example, Netflix might want to use matrix completion to predict which users will give which ratings to which movies, given the ratings given by all users of Netflix. However, not all users have rated all movies, thus the matrix is sparse and must thus be filled in. Netflix can use this prediction to decide which movies they should recommend to their users.

If we assume that all entries are independent, then no information is carried by one entry about another. Because of this, we cannot reconstruct the matrix. Thus, we need to make an assumption about the dependency within the matrix. A minimal assumption that we can make is that entries within the same row or the same column are not independent, *i.e.*,

$$a_{ij} \perp\!\!\!\perp \{a_{kl} \mid k \neq i \wedge l \neq j\} \mid \{a_{il} \mid l \neq j\} \cup \{a_{kj} \mid k \neq i\}.$$

This states that a_{ij} is independent from all entries not on the same row or column, given that we know all entries on the same row or column. However, in reality, we do not have all values on the row and column for all entries. Thus, effectively, we have an indirect coupling between all entries, since a_{kl} influences a_{kj} and a_{il} (if they are unknown), which influence a_{ij}.

Formally, we have a rating matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ and an observation matrix $\boldsymbol{\Omega} \in \{0,1\}^{n\times m}$, where $\omega_{ij} = 1$ means that a_{ij} is observed. The goal is to predict all values a_{kl}, where $\omega_{kl} = 0$.

A possible issue with the values is that some users might generally give higher ratings than others.[4] Thus, we want to account for this by variance normalization. We can do this per row (user) or per column (item), where the two means can be computed by

[4] While it makes more sense intuitively that we should normalize ratings per user, it has been shown to be more effective to normalize items.

$$\mu_i^{\text{row}} = \frac{\sum_{j=1}^m \omega_{ij} a_{ij}}{\max\{1, \sum_{j=1}^m \omega_{ij}\}}, \quad \mu_j^{\text{col}} = \frac{\sum_{i=1}^n \omega_{ij} a_{ij}}{\max\{1, \sum_{i=1}^n \omega_{ij}\}}.$$

We then normalize by

$$Z = \frac{X - \mu}{\sigma}.$$

To approximate $\boldsymbol{A} \in \mathbb{R}^{n\times m}$, we will make the assumption that $\boldsymbol{A}$ is a rank-k matrix and approximate it by the product of two matrices $\boldsymbol{U} \in \mathbb{R}^{n\times k}$ and $\boldsymbol{V} \in \mathbb{R}^{m\times k}$,

$$\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{V}^\top.$$

Then, we get the following loss function that we wish to minimize,

$$\boldsymbol{U}, \boldsymbol{V} \in \operatorname*{argmin}_{\boldsymbol{U},\boldsymbol{V}} \ell(\boldsymbol{U}, \boldsymbol{V}) \doteq \frac{1}{2}\left\|\Pi_{\boldsymbol{\Omega}}\left(\boldsymbol{A} - \boldsymbol{U}\boldsymbol{V}^\top\right)\right\|_F^2,$$

where $\Pi_{\Omega}(\boldsymbol{M}) = \boldsymbol{M} \odot \boldsymbol{\Omega}$ and $\|\cdot\|_F$ is the Frobenius norm,

$$\|\boldsymbol{M}\|_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} m_{ij}.$$

This can be viewed as $\boldsymbol{u}_i \in \mathbb{R}^k$ representing each user and $\boldsymbol{v}_j \in \mathbb{R}^k$ representing each movie. Their dot product is the predicted value of $\boldsymbol{a}_{ij}$. Furthermore, we have that $\boldsymbol{U}$ and $\boldsymbol{V}$ are fully determined by the observed values, but can be used to extrapolate to a full matrix, achieving our objective.

2.1 *Fully observed case*

In this section, we consider the fully observed case, where $\omega_{ij} = 1$ for all $i \in [n], j \in [m]$. Thus, we optimize w.r.t. the following loss function,

$$\ell(\boldsymbol{U}, \boldsymbol{V}) = \frac{1}{2} \left\| \boldsymbol{A} - \boldsymbol{U}\boldsymbol{V}^\top \right\|_F^2,$$

Scalar case. To observe the properties of the gradients, we will first consider the 1-dimensional case, where $a \approx uv$, with $u, v \in \mathbb{R}$. We have the following loss function,

$$\ell(u, v) = \frac{1}{2}(a - uv)^2.$$

We thus have the following gradients,

$$\delta = uv - a$$
$$\frac{\partial}{\partial u}\ell(u, v) = \delta v$$
$$\frac{\partial}{\partial v}\ell(u, v) = \delta u,$$

which induces the negative gradient field shown in Figure 3. Furthermore, we have the following Hessian,

$$\nabla^2 \ell(u, v) = \begin{bmatrix} v^2 & 2uv - a \\ 2uv - a & u^2 \end{bmatrix}.$$

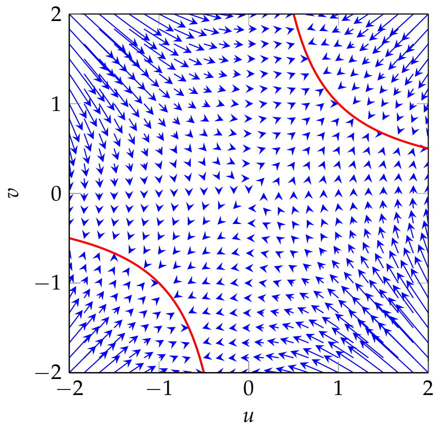

Figure 3. Negative gradient field for $a = 1$ in the scalar case with minima indicated by red. As can be seen, $[0, 0]$ is a 2-way saddle point and any vector $[-z, z]$ for $z \in \mathbb{R}$ moves toward it. We can start at any other point and use gradient-based optimization to converge to the minimum.

Lemma 6 (Second-order characterization of convexity)**.** If $f : \mathcal{X} \to \mathbb{R}$ is twice differentiable, then f is convex if and only if

$$\nabla^2 f(\boldsymbol{x}) \succeq \boldsymbol{0}, \quad \forall \boldsymbol{x} \in \operatorname{Int}(\mathcal{X}).$$

At the origin, we have the following Hessian,

$$\nabla^2 \ell(0, 0) = \begin{bmatrix} 0 & -a \\ -a & 0 \end{bmatrix}.$$

Thus, in the scalar case, the objective function is non-convex for $a \neq 0$. The same result can be generalized to any $n, m \geq 1$. This means that we do not have any general convergence guarantees using gradient-based methods. However, we can make guarantees if we study the gradient flow of the loss function with ordinary differential equations (ODE).

Consider a balanced initialization $u_0 = v_0$. We see that u and v will evolve identically, because their partial derivatives are equal in this case. We have the following update rule,

$$u_t = u_{t-1} - \eta \frac{\partial}{\partial u}\ell(u,v) = u_{t-1} - \eta v(uv - a),$$

where η is an arbitrarily small stepsize. We then have the following ODE,

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{u_t - u_{t-1}}{\eta} = -v(uv - a).$$

Consider $x = uv$, then we have the following ODE,

$$\frac{\mathrm{d}x}{\mathrm{d}t} = -2uv(uv - a) = -2x(x - a).$$

Solving this ODE yields the following solution for $x(t)$,

$$x(t) = a + \frac{ac - a^2}{ce^{2at} + a - c} \overset{t\to\infty}{=} a.$$

In conclusion, starting from a balanced initialization, gradient descent with a small enough η will converge uv to a.

Rank-1 model. Now consider the fully observed rank-1 model,

$$\ell(\boldsymbol{u},\boldsymbol{v}) = \frac{1}{2}\|\boldsymbol{A} - \boldsymbol{u}\boldsymbol{v}^\top\|_F^2, \quad \boldsymbol{A} \in \mathbb{R}^{n\times m}, \boldsymbol{u} \in \mathbb{R}^n, \boldsymbol{v} \in \mathbb{R}^m$$

Rewriting,

$$\begin{aligned}
&= \frac{1}{2}\mathrm{tr}\Big(\big(\boldsymbol{A} - \boldsymbol{u}\boldsymbol{v}^\top\big)^\top\big(\boldsymbol{A} - \boldsymbol{u}\boldsymbol{v}^\top\big)\Big) && \text{Identity: } \|\boldsymbol{M}\|_F^2 = \mathrm{tr}(\boldsymbol{M}^\top\boldsymbol{M}).\\
&= \frac{1}{2}\mathrm{tr}\Big(\boldsymbol{A}^\top\boldsymbol{A} - \boldsymbol{v}\boldsymbol{u}^\top\boldsymbol{A} - \boldsymbol{A}^\top\boldsymbol{u}\boldsymbol{v} + \boldsymbol{v}\boldsymbol{u}^\top\boldsymbol{u}\boldsymbol{v}^\top\Big)\\
&= \frac{1}{2}\Big(\mathrm{tr}\big(\boldsymbol{A}^\top\boldsymbol{A}\big) - \mathrm{tr}\big(\boldsymbol{v}\boldsymbol{u}^\top\boldsymbol{A}\big) - \mathrm{tr}\big(\boldsymbol{A}^\top\boldsymbol{u}\boldsymbol{v}\big) + \mathrm{tr}\big(\boldsymbol{v}\boldsymbol{u}^\top\boldsymbol{u}\boldsymbol{v}^\top\big)\Big) && \text{Linearity of trace.}\\
&= \frac{1}{2}\Big(\mathrm{tr}\big(\boldsymbol{A}^\top\boldsymbol{A}\big) - \mathrm{tr}\big(\boldsymbol{u}^\top\boldsymbol{A}\boldsymbol{v}\big) - \mathrm{tr}\big(\boldsymbol{v}^\top\boldsymbol{A}^\top\boldsymbol{u}\big) + \mathrm{tr}\big(\boldsymbol{v}^\top\boldsymbol{v}\boldsymbol{u}^\top\boldsymbol{u}\big)\Big) && \text{Cyclic property of trace.}\\
&= \frac{1}{2}\mathrm{tr}\big(\boldsymbol{A}^\top\boldsymbol{A}\big) + \frac{1}{2}\|\boldsymbol{u}\|^2\|\boldsymbol{v}\|^2 - \boldsymbol{u}^\top\boldsymbol{A}\boldsymbol{v}.
\end{aligned}$$

The first term is a constant, thus we have the following effective loss function,

$$\ell(\boldsymbol{u},\boldsymbol{v}) = \frac{1}{2}\|\boldsymbol{u}\|^2\|\boldsymbol{v}\|^2 - \boldsymbol{u}^\top\boldsymbol{A}\boldsymbol{v}.$$

We notice that the direction of $\boldsymbol{u}$ and $\boldsymbol{v}$ is fully decided by the second term. Thus, if we first solve for that, we can then easily solve for the norm of the two vectors by also considering the first term.

Specifically, let $\boldsymbol{u} = c_1\tilde{\boldsymbol{u}}$ and $\boldsymbol{v} = c_2\tilde{\boldsymbol{v}}$, such that $\|\tilde{\boldsymbol{u}}\| = \|\tilde{\boldsymbol{v}}\| = 1$, and $c = c_1 \cdot c_2$. Then, we have the following loss function with an additional parameter c,

$$\ell(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, c) = \frac{1}{2}c^2 - c\tilde{\boldsymbol{u}}^\top \boldsymbol{A}\tilde{\boldsymbol{v}}.$$

Thus, we first maximize $\tilde{\boldsymbol{u}}^\top \boldsymbol{A}\tilde{\boldsymbol{v}}$ under the constraint that $\boldsymbol{u}$ and $\boldsymbol{v}$ are unit vectors, and then decide which c minimizes the loss function. Thus, first we have the following problem

$$\boldsymbol{u}, \boldsymbol{v} \in \underset{\|\boldsymbol{u}\|=\|\boldsymbol{v}\|=1}{\operatorname{argmax}} \ \boldsymbol{u}^\top \boldsymbol{A}\boldsymbol{v}.$$

We can solve this problem with its constraints by using Lagrangian multipliers,

$$\mathcal{L} = \boldsymbol{u}^\top \boldsymbol{A}\boldsymbol{v} - \alpha(\|\boldsymbol{u}\|^2 - 1) - \beta(\|\boldsymbol{v}\|^2 - 1).$$

Using the first-order optimality condition, we get the following solutions,

$$\begin{aligned}
\nabla_{\boldsymbol{u}}\mathcal{L} &= \boldsymbol{A}\boldsymbol{v} - 2\alpha\boldsymbol{u} \stackrel{!}{=} 0 \implies \boldsymbol{u} = \frac{\boldsymbol{A}\boldsymbol{v}}{\|\boldsymbol{A}\boldsymbol{v}\|} \\
\nabla_{\boldsymbol{v}}\mathcal{L} &= \boldsymbol{A}^\top\boldsymbol{u} - 2\beta\boldsymbol{v} \stackrel{!}{=} 0 \implies \boldsymbol{v} = \frac{\boldsymbol{A}^\top\boldsymbol{u}}{\|\boldsymbol{A}^\top\boldsymbol{u}\|}.
\end{aligned}$$

Thus, the solutions must satisfy

$$\boldsymbol{u} \propto \left(\boldsymbol{A}\boldsymbol{A}^\top\right)\boldsymbol{u}, \quad \boldsymbol{v} \propto \left(\boldsymbol{A}^\top\boldsymbol{A}\right)\boldsymbol{v}.$$

As a result, we know that $\boldsymbol{u}$ should be proportional to an eigenvector of $\boldsymbol{A}\boldsymbol{A}^\top$ and $\boldsymbol{v}$ should be proportional to an eigenvector of $\boldsymbol{A}^\top\boldsymbol{A}$. Using this fact, we can rewrite the objective as

$$\begin{aligned}
\boldsymbol{u}^\top\boldsymbol{A}\boldsymbol{v} &= \frac{\boldsymbol{u}^\top\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{u}}{\|\boldsymbol{A}^\top\boldsymbol{u}\|} \\
&= \frac{\boldsymbol{u}^\top\lambda\boldsymbol{u}}{\sqrt{\boldsymbol{u}^\top\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{u}}} \\
&= \frac{\boldsymbol{u}^\top\lambda\boldsymbol{u}}{\sqrt{\boldsymbol{u}^\top\lambda\boldsymbol{u}}} \\
&= \frac{\lambda\|\boldsymbol{u}\|^2}{\sqrt{\lambda}\|\boldsymbol{u}\|} \\
&= \sqrt{\lambda}.
\end{aligned}$$

λ is the eigenvalue of $\boldsymbol{A}\boldsymbol{A}^\top$ corresponding to $\boldsymbol{u}$.

$\boldsymbol{u}$ is a unit vector.

In order to maximize this term, we must thus select $\boldsymbol{u}$ and $\boldsymbol{v}$ to be the principal eigenvectors of their respective matrices. Furthermore, $\ell(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, c)$ with this selection of $\tilde{\boldsymbol{u}}$ and $\tilde{\boldsymbol{v}}$ is minimized by selecting $c = \sqrt{\lambda_1}$, where λ_1 is the principal eigenvalue of $\boldsymbol{A}\boldsymbol{A}^\top$ and $\boldsymbol{A}^\top\boldsymbol{A}$.

In conclusion, the loss for a rank-1 model is minimized by selecting

$$\boldsymbol{A} = \sqrt{\lambda_1}\boldsymbol{u}\boldsymbol{v}^\top,$$

where $\boldsymbol{u}$ and $\boldsymbol{v}$ are the principal eigenvectors of $\boldsymbol{A}\boldsymbol{A}^\top$ and $\boldsymbol{A}^\top\boldsymbol{A}$, and λ_1 is the principal eigenvalue. We can compute these vectors by the power iteration algorithm.

Rank-k model. We can generalize the previous result by making use of the singular value decomposition.

Theorem 7 (SVD theorem)**.** For each matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$, there exists a diagonal matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times m}$ with ordered entries $\sigma_i \geq \sigma_{i+1} \geq 0, \forall i \in \min\{n,m\}$ and orthogonal matrices $\boldsymbol{U} \in \mathbb{R}^{n\times n}, \boldsymbol{V} \in \mathbb{R}^{m\times m}$ such that $\boldsymbol{A}$ can be expressed as

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top.$$

Lemma 8. Let the SVD of $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ be given by $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, then

$$\|\boldsymbol{A}\|_F^2 = \sum_{i=1}^{\min\{n,m\}} \sigma_i^2.$$

Proof. This can be shown by the properties of the trace operator,

$$\begin{aligned}\|\boldsymbol{A}\|_F^2 &= \mathrm{tr}\left(\boldsymbol{A}^\top\boldsymbol{A}\right) = \mathrm{tr}\left(\boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\right) = \mathrm{tr}\left(\boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}\boldsymbol{V}^\top\right)\\ &= \mathrm{tr}\left(\boldsymbol{V}^\top\boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}\right) = \mathrm{tr}\left(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}\right) = \sum_{i=1}^{\min\{n,m\}} \sigma_i^2.\end{aligned}$$

■

Lemma 9. Let the SVD of $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ be given by $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, then

$$\|\boldsymbol{A}\|_2 \doteq \sup_{\|\boldsymbol{x}\|=1} \|\boldsymbol{A}\boldsymbol{x}\| = \sigma_1.$$

Spectral norm.

Proof. We use the fact that orthogonal matrices preserve the Euclidean norm,

$$\begin{aligned}\sup_{\|\boldsymbol{x}\|=1} \|\boldsymbol{A}\boldsymbol{x}\| &= \sup_{\|\boldsymbol{x}\|=1} \|\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{x}\| = \sup_{\|\boldsymbol{x}\|=1} \|\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{x}\|\\ &= \sup_{\|\boldsymbol{z}\|=1} \|\boldsymbol{\Sigma}\boldsymbol{z}\| = \|\boldsymbol{\Sigma}\|_2 = \sigma_1.\end{aligned}$$

■

The Eckart-Young theorem is widely used for linear matrix approximation. It states that if we prune the singular values below σ_k in the SVD representation, we get an optimal rank-k approximation of a matrix.

Theorem 10 (Eckart-Young theorem)**.** Let the SVD of $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ be given by $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$. Then, for all $1 \leq k \leq \min\{n,m\}$, we have

$$\boldsymbol{A}_k \doteq \boldsymbol{U}\mathrm{diag}([\sigma_1,\ldots,\sigma_k])\boldsymbol{V}^\top \in \operatorname*{argmin}_{\mathrm{rank}(\boldsymbol{B})\leq k} \|\boldsymbol{A}-\boldsymbol{B}\|_F.$$

From this, we can easily find a rank-k approximation by

$$\boldsymbol{A} \approx \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}^\top.$$

Furthermore, we can directly compute the squared error of low rank approximations as

$$\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2 = \sum_{i=k+1}^{\operatorname{rank}(\boldsymbol{A})} \sigma_i^2.$$

The SVD is computable in $\mathcal{O}\big(\min\{nm^2, mn^2\}\big)$, which is fast for a non-convex problem. However, we cannot use this technique in the case of incomplete observations.

2.2 *Incompletely observed case*

In practice, the matrix $\boldsymbol{A}$ is often incompletely observed, meaning that we can only compute the loss function w.r.t. the observed entries. This task is called "Matrix completion", and we cannot use SVD in this case. Even worse, this problem is NP-hard, thus we have to resort to approximation algorithms.

Alternating least squares. As we saw, we parametrize the approximation of $\boldsymbol{A}$ as a factorization of two matrices $\boldsymbol{U} \in \mathbb{R}^{n \times k}$ and $\boldsymbol{V} \in \mathbb{R}^{m \times k}$,

$$\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{V}^\top.$$

Since we cannot solve the problem exactly, we want to add regularization to the loss function to increase numerical stability of the solution (this will also be very important later),

$$\ell(\boldsymbol{U}, \boldsymbol{V}) = \frac{1}{2}\left\|\Pi_{\boldsymbol{\Omega}}\left(\boldsymbol{A} - \boldsymbol{U}\boldsymbol{V}^\top\right)\right\|_F^2 + \frac{\lambda}{2}\left(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2\right), \quad \lambda > 0.$$

If we fully expand the norms, we will find that this objective is a 4-th degree polynomial in the parameters (entries of $\boldsymbol{U}$ and $\boldsymbol{V}$) with the following monomials,

$$\begin{aligned} \omega_{ij} u_{ir} v_{jr} u_{is} v_{js}, &\quad 1 \le r, s \le k \\ \omega_{ij} u_{ir} v_{jr}, &\quad 1 \le r \le k \\ u_{ir}^2, &\quad 1 \le r \le k \\ v_{jr}^2, &\quad 1 \le r \le k. \end{aligned}$$

Importantly, every 4-th order term involves exactly one row index i of $\boldsymbol{U}$ and one row index j of $\boldsymbol{V}$. In other words, the objective w.r.t. $\boldsymbol{u}_i$ depends further only on $\boldsymbol{V}$ and not any other row of $\boldsymbol{U}$.[5] Using this information, we can separate out part of the objective function depending on a row $\boldsymbol{v}_j$,

[5] The parameter dependencies form a bipartite graph between the rows of $\boldsymbol{U}$ and $\boldsymbol{V}$.

where we treat $\boldsymbol{U}$ as fixed,

$$\begin{aligned}
\ell_{\boldsymbol{U}}(\boldsymbol{v}_j) &= \frac{1}{2}\left\|\Pi_{\boldsymbol{\Omega}}\left(\boldsymbol{A}-\boldsymbol{U}\boldsymbol{V}^\top\right)\right\|_F^2 + \frac{\lambda}{2}\left(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2\right) \\
&\propto \frac{1}{2}\sum_{i=1}^{n}\sum_{j'=1}^{m}\omega_{ij'}\left|a_{ij'} - \left(\boldsymbol{U}\boldsymbol{V}^\top\right)_{ij'}\right|^2 + \frac{\lambda}{2}\sum_{j'=1}^{m}\sum_{r=1}^{k}|v_{j'r}|^2 \\
&\propto \frac{1}{2}\sum_{i=1}^{n}\omega_{ij}\left(a_{ij} - \boldsymbol{u}_i^\top\boldsymbol{v}_j\right)^2 + \frac{\lambda}{2}\sum_{r=1}^{k}v_{jr}^2 \\
&= \frac{1}{2}\sum_{i=1}^{n}\omega_{ij}a_{ij}^2 + \omega_{ij}\boldsymbol{u}_i^\top\boldsymbol{v}_j\boldsymbol{u}_i^\top\boldsymbol{v}_j - 2\omega_{ij}a_{ij}\boldsymbol{u}_i^\top\boldsymbol{v}_j + \frac{\lambda}{2}\boldsymbol{v}_j^\top\boldsymbol{v}_j \\
&\propto \frac{1}{2}\sum_{i=1}^{n}\omega_{ij}\boldsymbol{v}_j^\top\boldsymbol{u}_i\boldsymbol{u}_i^\top\boldsymbol{v}_j + \frac{\lambda}{2}\boldsymbol{v}_j^\top\boldsymbol{v}_j - \sum_{i=1}^{n}\omega_{ij}a_{ij}\boldsymbol{u}_i^\top\boldsymbol{v}_j \\
&= \frac{1}{2}\left(\boldsymbol{v}_j^\top\left(\sum_{i=1}^{n}\omega_{ij}\boldsymbol{u}_i\boldsymbol{u}_i^\top\right)\boldsymbol{v}_j + \lambda\boldsymbol{v}_j^\top\boldsymbol{v}_j\right) - \left(\sum_{i=1}^{n}\omega_{ij}a_{ij}\boldsymbol{u}_i^\top\right)\boldsymbol{v}_j \\
&= \frac{1}{2}\boldsymbol{v}_j^\top\left(\sum_{i=1}^{n}\omega_{ij}\boldsymbol{u}_i\boldsymbol{u}_i^\top + \lambda\boldsymbol{I}_k\right)\boldsymbol{v}_j - \left(\sum_{i=1}^{n}\omega_{ij}a_{ij}\boldsymbol{u}_i^\top\right)\boldsymbol{v}_j.
\end{aligned}$$

Expand norms and remove $\|\boldsymbol{U}\|_F^2$, because it does not depend on $\boldsymbol{v}_j$.

Remove all terms that do not contain entries of $\boldsymbol{v}_j$.

$\boldsymbol{u}_i^\top\boldsymbol{v}_j = \boldsymbol{v}_j^\top\boldsymbol{u}_i$.

Analogously, we can construct the following objective w.r.t. $\boldsymbol{u}_i$,

$$\ell_{\boldsymbol{V}}(\boldsymbol{u}_i) = \frac{1}{2}\boldsymbol{u}_i^\top\left(\sum_{j=1}^{m}\omega_{ij}\boldsymbol{v}_j\boldsymbol{v}_j^\top + \lambda\boldsymbol{I}_k\right)\boldsymbol{u}_i - \left(\sum_{j=1}^{m}\omega_{ij}a_{ij}\boldsymbol{v}_j^\top\right)\boldsymbol{u}_i.$$

Since the matrix in the first term is rank-k,[6] we can invert it. We have the following first-order optimal solution to the above loss function,

$$\begin{aligned}
\boldsymbol{v}_j^\star &= \left(\sum_{i=1}^{n}\omega_{ij}\boldsymbol{u}_i\boldsymbol{u}_i^\top + \lambda\boldsymbol{I}_k\right)^{-1}\left(\sum_{i=1}^{n}\omega_{ij}a_{ij}\boldsymbol{u}_i\right) \\
\boldsymbol{u}_i^\star &= \left(\sum_{j=1}^{m}\omega_{ij}\boldsymbol{v}_j\boldsymbol{v}_j^\top + \lambda\boldsymbol{I}_k\right)^{-1}\left(\sum_{j=1}^{m}\omega_{ij}a_{ij}\boldsymbol{v}_j\right).
\end{aligned}$$

This involves taking the inverse of a $k \times k$ matrix, thus computing $\boldsymbol{u}_i^\star$ and $\boldsymbol{v}_i^\star$ has $\mathcal{O}(k^3)$ complexity, which is generally quite fast.

Alternating least squares (ALS) makes use of these subproblems by alternating between optimizing $\boldsymbol{U}$ given $\boldsymbol{V}$ and $\boldsymbol{V}$ given $\boldsymbol{U}$,

$$\begin{aligned}
\boldsymbol{V}^{(t+1)} &= \operatorname*{argmin}_{\boldsymbol{V}} \ell_{\boldsymbol{U}^{(t)}}(\boldsymbol{V}) \\
\boldsymbol{U}^{(t+1)} &= \operatorname*{argmin}_{\boldsymbol{U}} \ell_{\boldsymbol{V}^{(t+1)}}(\boldsymbol{U}).
\end{aligned}$$

An iteration of ALS has complexity $\mathcal{O}\left((n+m)\cdot k^3\right)$. However, since each row of both matrices can be optimized in parallel, this can be done much faster in practice. Furthermore, since we are optimizing the objective fully over one half of the parameter space, the objective monotonically decreases. Hence, it converges to a first-order optimal fixed point, which may be a saddle point. Moreover, we have the advantage that we can easily augment the model by adding additional dimensions and minimizing w.r.t. the existing dimensions, which may be useful when new users enter the system or new items become available.

[6] This is because of the identity matrix, which is a result of the regularization.

Projection methods. We will now consider optimizing the objective by gradient descent, where the gradient is

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{B}}\frac{1}{2}\|\Pi_{\boldsymbol{\Omega}}(\boldsymbol{A}-\boldsymbol{B})\|_F^2 &= \Pi_{\boldsymbol{\Omega}}(\boldsymbol{A}-\boldsymbol{B}) \odot \frac{\partial}{\partial \boldsymbol{B}}\Pi_{\boldsymbol{\Omega}}(\boldsymbol{A}-\boldsymbol{B}) \\ &= \Pi_{\boldsymbol{\Omega}}(\boldsymbol{A}-\boldsymbol{B}) \odot \Pi_{\boldsymbol{\Omega}}(-\boldsymbol{1}_{n\times m}) \\ &= -\Pi_{\boldsymbol{\Omega}}(\boldsymbol{A}-\boldsymbol{B}).\end{aligned}$$

$\frac{\partial}{\partial \boldsymbol{X}}\|\boldsymbol{X}\|_F^2 = 2\boldsymbol{X}$.

However, gradient descent does not constrain its iterates to be rank-k matrices. Projected gradient descent solves this by projecting to the constrained space in between every gradient step. In general, it is hard to project to a space of rank-k matrices, but SVD makes it possible by making use of the Eckart-Young theorem. We will denote $[\boldsymbol{M}]_k$ as the projection to the space of rank-k matrices, which is computed by pruning all singular values below σ_k, as shown by the Eckart-Young theorem. Hence, the update rule is the following,

$$\boldsymbol{A}^{(t+1)} = \left[\boldsymbol{A}^{(t)} + \eta\Pi_{\boldsymbol{\Omega}}\left(\boldsymbol{A}-\boldsymbol{A}^{(t)}\right)\right]_k.$$

This converges to a first-order optimal solution if η is small enough as a general result of projected gradient descent.

The problem with this approach is that the space of rank-k matrices is non-convex. Thus, the next idea is to find the tightest convex relaxation of this space. *I.e.*, we want to find a convex space that contains all rank-k matrices, but not too many more. For this, we use the nuclear norm,

$$\|\boldsymbol{M}\|_* \doteq \sum_{i=1}^{\operatorname{rank}(\boldsymbol{M})} \sigma_i,$$

where σ_i is the i-th singular value of $\boldsymbol{M}$.

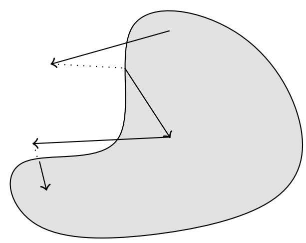

Figure 4. Illustration of projected gradient descent, where the dotted lines indicate projection steps.

Let $\boldsymbol{\sigma}(\boldsymbol{M})$ be the vector of singular values of $\boldsymbol{M}$, then, as shown by Lemma 8, the Frobenius norm can be computed by

$$\|\boldsymbol{M}\|_F = \|\boldsymbol{\sigma}(\boldsymbol{M})\|_2.$$

Similarly, the nuclear norm can be computed by

$$\|\boldsymbol{M}\|_* = \|\boldsymbol{\sigma}(\boldsymbol{M})\|_1.$$

Definition 11 (Convex envelope)**.** The convex envelope of a function $f : \mathcal{X} \to \mathbb{R}$ is the largest convex function g such that

$$g(\boldsymbol{x}) \leq f(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathcal{X}.$$

Theorem 12. The convex envelope of $\operatorname{rank}(\cdot)$ on $\{\boldsymbol{M} \mid \|\boldsymbol{M}\|_2 \leq 1\}$ is the nuclear norm $\|\cdot\|_*$.

We will use this to approximate the objective by a convex function. We have the following objective,

$$\ell(\boldsymbol{B}) = \frac{1}{2}\|\Pi_{\boldsymbol{\Omega}}(\boldsymbol{A}-\boldsymbol{B})\|_F^2, \quad \text{rank}(\boldsymbol{B}) \leq k, \|\boldsymbol{B}\|_2 \leq c.$$

The second constraint comes from the condition of the space of Theorem 12.

This can be rewritten using Lagrange multipliers as a non-convex objective,

$$\begin{aligned}\ell(\boldsymbol{B}) &= \frac{1}{2}\|\Pi_{\boldsymbol{\Omega}}(\boldsymbol{A}-\boldsymbol{B})\|_F^2 + r\cdot\text{rank}(\boldsymbol{B}) + \mu\|\boldsymbol{B}\|_2 \\ &\approx \frac{1}{2}\|\Pi_{\boldsymbol{\Omega}}(\boldsymbol{A}-\boldsymbol{B})\|_F^2 + \tau\|\boldsymbol{B}\|_* + \gamma\|\boldsymbol{B}\|_F.\end{aligned}$$

The nuclear norm is the convex envelope of the rank function and the Frobenius norm upper bounds the spectral norm.

This approximation of the objective is convex, so, using gradient descent and correct η, we can solve it.

Theorem 13. Let $\boldsymbol{A}$ with SVD $\boldsymbol{A} = \boldsymbol{U}\text{diag}(\boldsymbol{\sigma})\boldsymbol{V}^\top$ be given, then

$$\begin{aligned}\text{shrink}_\tau(\boldsymbol{A}) &\doteq \underset{\boldsymbol{B}}{\text{argmin}}\,\frac{1}{2}\|\boldsymbol{A}-\boldsymbol{B}\|_F^2 + \tau\|\boldsymbol{B}\|_* \\ &= \boldsymbol{U}\text{diag}(\boldsymbol{\sigma}-\tau)_+\boldsymbol{V}^\top,\end{aligned}$$

where $\boldsymbol{M}_+$ clips all entries of $\boldsymbol{M}$ at zero.

Note that the rank of $\text{shrink}_\tau(\boldsymbol{A})$ decreases monotonically with τ, because more singular values are zeroed out as τ increases.

Using the shrink operator, we can define a new update rule, starting from $\boldsymbol{A}^{(0)} = \boldsymbol{0}$,

$$\boldsymbol{A}^{(t+1)} = \boldsymbol{A}^{(t)} + \eta_t\Pi_{\boldsymbol{\Omega}}\Big(\boldsymbol{A} - \text{shrink}_\tau\Big(\boldsymbol{A}^{(t)}\Big)\Big).$$

With a suitable schedule $\eta_t > 0$, this algorithm will converge to

$$\begin{aligned}\text{shrink}_\tau\Big(\boldsymbol{A}^{(t)}\Big) \xrightarrow{t\to\infty} \boldsymbol{A}^\star = &\underset{\boldsymbol{B}}{\text{argmin}}\left\{\tau\|\boldsymbol{B}\|_* + \frac{1}{2\tau}\|\boldsymbol{B}\|_F^2\right\} \\ &\text{s.t.}\quad \Pi_{\boldsymbol{\Omega}}(\boldsymbol{A}-\boldsymbol{B}) = \boldsymbol{0}.\end{aligned}$$

The result of this algorithm will be exactly reproducing the observed entries, which the non-relaxed problem cannot guarantee, as there may be no rank-k algorithm with a projected residual that is zero. However, the convex relaxation approach does not define a low-rank sequence of matrices. Rather, the shrinkage operator is used to implicitly encourage low-rank approximations.

In conclusion, the singular value projection approach maintains a rank-k matrix, while the convex relaxation approach maintains a sparse iterate sequence. Both are beneficial relative to a dense representation of $\boldsymbol{A}$.

Exact recovery. So far we have only been interested in how to find the "best" completed matrix, given an incomplete matrix $\boldsymbol{A}$. However, now we are interested in the conditions under which we can exactly recover the matrix, given that we know that the underlying matrix $\boldsymbol{A}$ is of rank k.

Assume $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, then the degrees of freedom are the k singular values and k left and right singular vectors. The i-th singular vector has $n - i$ degrees of freedom due to the constraints of unit length and pairwise orthogonality, thus the SVD has the following degrees of freedom,

$$k + 2\sum_{i=1}^{k} n - i = k + 2\left(nk - \sum_{i=1}^{k} i\right) = 2nk - k^2.$$

Thus, a necessary condition to exactly recover $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is that we have at least that number of observations. This condition is not sufficient, which we will prove by an example. Let $\boldsymbol{A}$ have the SVD $\boldsymbol{A} = \boldsymbol{U\Sigma V}^\top$, then

$$\boldsymbol{A} = \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^\top + \sum_{i=2}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top.$$

Let $\boldsymbol{u}_1 = \boldsymbol{e}_i$ and $\boldsymbol{v}_1 = \boldsymbol{e}_j$, then in order to recover σ_1, we need to have sampled a_{ij}. Thus, it does not only matter how many entries we have sampled, but also where we have sampled them. Intuitively, we need the information to be sufficiently distributed.

The following gives us conditions for which we can exactly reconstruct the underlying matrix with high probability, under the assumption that the entries have been uniformly sampled.

Definition 14 (Incoherence)**.** Let $\boldsymbol{A}$ be a rank-k matrix with SVD $\boldsymbol{A} = \boldsymbol{U\Sigma V}^\top$. Furthermore, define the following matrices,

$$\boldsymbol{P} \doteq \sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{u}_i^\top, \quad \boldsymbol{Q} \doteq \sum_{i=1}^{k} \boldsymbol{v}_i \boldsymbol{v}_i^\top, \quad \boldsymbol{E} \doteq \sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{v}_i^\top.$$

Then, $\boldsymbol{A}$ is incoherent with parameter μ if and only if the following conditions are satisfied,

$$\begin{aligned} |p_{ij}|, |q_{ij}| &\leq \frac{\mu\sqrt{k}}{n}, \quad i \neq j \\ \left|p_{ii} - \frac{k}{n}\right|, \left|q_{ii} - \frac{k}{n}\right| &\leq \frac{\mu\sqrt{k}}{n} \\ |e_{ij}| &\leq \frac{\mu\sqrt{k}}{n}. \end{aligned}$$

Theorem 15. Let $A \in \mathbb{R}^{n \times n}$ be a rank-k matrix that is incoherent with $\mu \geq 1$ and for which S samples have been observed at random. Then, there is a universal constant C such that if $S \geq C\mu^2 nk(\log n)^6$, then with probability at least $1 - n^{-3}$, A fulfills

$$A = \underset{B}{\operatorname{argmin}} \|B\|_*, \quad \text{s.t. } \Pi_\Omega(B) = \Pi_\Omega(A).$$

This says that if the matrix is recoverable from its sampled entries, then it can be recovered via nuclear norm minimization.

2.3 *Randomized methods for SVD*

TODO: Is this necessary?

3 Latent variable models

The philosophy behind latent variable models is that we have *observables* $\boldsymbol{X}$, which are augmented by *latent variables* $\boldsymbol{Z}$, which happens by specifying a *complete data model* $p(\boldsymbol{X}, \boldsymbol{Z})$, which implies a *marginal model*,

$$p(\boldsymbol{X} = \boldsymbol{x}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} p(\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{Z} = \boldsymbol{z}).$$

The marginal model can be specified by a conditional $p(\boldsymbol{X} \mid \boldsymbol{Z})$ and prior $p(\boldsymbol{Z})$,

$$p(\boldsymbol{X} = \boldsymbol{x}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} p(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{Z} = \boldsymbol{z}) p(\boldsymbol{Z} = \boldsymbol{z}).$$

3.1 Probabilistic clustering models

Assume we are given a dataset of s patterns $\{\boldsymbol{x}_t \mid t \in [s]\}$. The conceptually simplest family of latent variable models associates a k-class categorical random variable Z_t with each pattern. The latent information tags a pattern as a member of a group or class.

Specifically, we have

$$Z_t \sim \text{Categorical}(\pi_1, \ldots, \pi_k), \quad P(Z_t = z) = \pi_z,$$

where $\boldsymbol{\pi} \in \Delta^{k-1}$ is an unknown parameter that encodes the prior probabilities for each class.[7] To fully parametrize the latent variable model, we need a class conditional distribution for each class, $p(\boldsymbol{x} \mid z)$. By parametrizing the class-conditional distribution of z by $\boldsymbol{\theta}_z$, we have the following parametrized distribution over $\boldsymbol{X}$,

[7] Δ^{k-1} is the k-dimensional probability simplex, which is the space of vectors where all elements are non-negative and sum to 1.

$$p(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{z=1}^{k} \pi_z p(\boldsymbol{x}; \boldsymbol{\theta}_z).$$

We observe that mixture distributions are convex combinations of class-specific distributions. Moreover, this model is fully parametrized by

$$\boldsymbol{\theta} = [\boldsymbol{\pi}, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k].$$

For given parameters $\boldsymbol{\theta}$, we can use Bayes' rule to compute the latent posteriors,

$$p(z \mid \boldsymbol{x}; \boldsymbol{\theta}) = \frac{\pi_z p(\boldsymbol{x}; \boldsymbol{\theta}_z)}{\sum_{\zeta=1}^{k} \pi_\zeta p(\boldsymbol{x}; \boldsymbol{\theta}_\zeta)}.$$

We can interpret these probabilities as probabilistic cluster memberships.

Learning the parameters. A common approach to learn the best parameters is to maximize the likelihood of the data, which is called maximum likelihood estimation (MLE). In other words, we choose the model parameters that maximize the probability of the observed data. It works by maximizing the log-likelihood,

$$\ell(\boldsymbol{\theta}; \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_s\}) = \sum_{t=1}^{s} \log p(\boldsymbol{x}_t; \boldsymbol{\theta}) = \sum_{t=1}^{s} \log \sum_{z=1}^{k} \pi_z p(\boldsymbol{x}_t; \boldsymbol{\theta}_z).$$

This gives us the following optimization problem,

$$\boldsymbol{\theta}^{\mathrm{MLE}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_s\}),$$

which we optimize by the expectation-maximization (EM) algorithm. The EM algorithm is a general tool for learning in latent variable models, but we will introduce it in the context of mixture models.

Let $\boldsymbol{q}_t \in \Delta^{k-1}, \forall t \in [s]$. Using this we can get the evidence lower bound (ELBO),

$$\begin{aligned}
\ell(\boldsymbol{\theta}; \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_s\}) &= \sum_{t=1}^{s} \log \sum_{z=1}^{k} q_{tz} \frac{\pi_z}{q_{tz}} p(\boldsymbol{x}; \boldsymbol{\theta}_z) \\
&\geq \sum_{t=1}^{s} \sum_{z=1}^{k} q_{tz} \log\left(\frac{\pi_z}{q_{tz}} p(\boldsymbol{x}; \boldsymbol{\theta}_z)\right) \\
&= \sum_{t=1}^{s} \sum_{z=1}^{k} q_{tz} \log p(\boldsymbol{x}; \boldsymbol{\theta}_z) - \sum_{z=1}^{k} q_{tz} \log \frac{q_{tz}}{\pi_z} \\
&= \sum_{t=1}^{s} \sum_{z=1}^{k} q_{tz} \log p(\boldsymbol{x}; \boldsymbol{\theta}_z) - D_{\mathrm{KL}}(\boldsymbol{q}_t \| \boldsymbol{\pi}).
\end{aligned}$$

Jensen's inequality.

Since this lower bounds the MLE, we can use the ELBO as an objective to be maximized. Maximizing w.r.t. $\boldsymbol{q}$ increases the tightness of the bound of ELBO on the MLE, while maximizing w.r.t. $\boldsymbol{\theta}$ improves the model fit.

Firstly, we will solve for the optimal choice of $\boldsymbol{q}_t$,

$$\sum_{t=1}^{s} \sum_{z=1}^{k} q_{tz} \log p(\boldsymbol{x}; \boldsymbol{\theta}_z) - D_{\mathrm{KL}}(\boldsymbol{q}_t \| \boldsymbol{\pi}), \quad \sum_{z=1}^{k} q_{tz} = 1, \forall t \in [s].$$

We can solve for every $\boldsymbol{q}_t$ independently, because the ELBO is separable w.r.t. $\boldsymbol{q}_t$. Furthermore, we enforce the normalization constraint on $\boldsymbol{q}$ by introducing a Lagrange multiplier λ. This results in the following objective to maximize per $\boldsymbol{q}_t$,

$$\boldsymbol{q}_t^\star \in \operatorname*{argmax}_{\boldsymbol{q}_t} \ell(\boldsymbol{q}_t) \doteq \sum_{z=1}^{k} q_z \left(\log p(\boldsymbol{x}_t; \boldsymbol{\theta}_z) - \log \frac{q_z}{\pi_z}\right) - \lambda\left(\sum_{z=1}^{k} q_z - 1\right).$$

The derivative w.r.t. q_z of this function is

$$\begin{aligned}
\frac{\partial \ell(\boldsymbol{q}_t)}{\partial q_{tz}} &= \frac{\partial}{\partial q_{tz}} \sum_{z'=1}^{k} q_{tz'}(\log p(\boldsymbol{x}_t; \boldsymbol{\theta}_{z'}) + \log \pi_{z'} - \log q_{tz'}) \\
&= \log p(\boldsymbol{x}_t; \boldsymbol{\theta}_z) + \log \pi_z - \frac{\partial}{\partial q_{tz}} q_{tz} \log q_{tz} - \lambda \\
&= \log p(\boldsymbol{x}_t; \boldsymbol{\theta}_z) + \log \pi_z - \log q_{tz} - \frac{q_{tz}}{q_{tz}} - \lambda \\
&= \log p(\boldsymbol{x}_t; \boldsymbol{\theta}_z) + \log \pi_z - \log q_{tz} - \lambda - 1.
\end{aligned}$$

Thus, we have the following first-order optimality condition,

$$\log p(\boldsymbol{x}_t; \boldsymbol{\theta}_z) + \log \pi_z - \log q_{tz} \overset{!}{=} \lambda + 1.$$

Exponentiating both sides yields

$$q_{tz} \stackrel{!}{=} \frac{\pi_z p(\boldsymbol{x}_t; \boldsymbol{\theta}_z)}{e^{\lambda+1}}.$$

Enforcing the constraint of $\boldsymbol{q}_t \in \Delta^{k-1}$, we get

$$q_{tz} \stackrel{!}{=} \frac{\pi_z p(\boldsymbol{x}_t; \boldsymbol{\theta}_z)}{\sum_{\zeta=1}^{k} \pi_\zeta p(\boldsymbol{x}_t; \boldsymbol{\theta}_\zeta)}.$$

As we saw before, this is the posterior of the latent class variable $p(z \mid \boldsymbol{x}_t; \boldsymbol{\theta})$. Note that the optima choice of the variational parameters depend on the parameters $\boldsymbol{\theta}$. Thus, it is only a partial step, called the expectation (E) step.

Now we also need a step to maximize the model parameters $\boldsymbol{\theta}$. We can easily solve for $\boldsymbol{\pi}$ by

$$\pi_z^\star = \frac{1}{s} \sum_{t=1}^{s} q_{tz}.$$

Moreover, the solution for $\boldsymbol{\theta}_z$ depends on the choice of the model distribution, but we can generally get to separable problems,

$$\boldsymbol{\theta}_z^\star \in \operatorname*{argmax}_{\boldsymbol{\theta}_z} \sum_{t=1}^{s} q_{tz} \log p(\boldsymbol{x}_t; \boldsymbol{\theta}_z).$$

This means that the parameters for different classes z are decoupled given the variational parameters $\boldsymbol{q}$. Thus, for each component, we only have to solve a weighted MLE problem, which is often possible to do analytically. This partial step is called the maximization (M) step.

Gaussian mixture model. We will consider a common special case of the above framework, where we specify the component models $p(\boldsymbol{X} \mid Z)$ by Gaussians with unit variance,

$$p(\boldsymbol{x}; \boldsymbol{\pi}, \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}) = \sum_{z=1}^{k} \pi_z p(\boldsymbol{x}; \boldsymbol{\mu}_z),$$

where

$$p(\boldsymbol{x}; \boldsymbol{\mu}_z) = (2\pi)^{-d/2} \exp\left(-\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{\mu}_z\|^2\right).$$

The EM algorithm then consists of the following alternating equations,

$$q_{tz} \doteq p(z \mid \boldsymbol{x}_t, \boldsymbol{\theta}) = \frac{\pi_z \exp\left(-\frac{1}{2}\|\boldsymbol{z}_t - \boldsymbol{\mu}_z\|^2\right)}{\sum_{\zeta=1}^{k} \pi_\zeta \exp\left(-\frac{1}{2}\|\boldsymbol{z}_t - \boldsymbol{\mu}_\zeta\|^2\right)} \qquad \text{E-step.}$$

$$\boldsymbol{\mu}_z = \frac{\sum_{t=1}^{s} q_{tz} \boldsymbol{x}_t}{\sum_{t=1}^{s} q_{tz}}, \quad \pi_z = \frac{1}{s} \sum_{t=1}^{s} q_{tz}. \qquad \text{M-step.}$$

Intuitively, the E-step (soft) clusters the data points and the M-step computes the weighted centroids of each component.

3.2 Topic models

We will now consider a class of latent variable models known as topic models. These are used to analyze document collections and to discover the topical content of document. Informally, topical content is the information that the words of a document carry about what the document is about.

Let Σ be the word vocabulary with $|\Sigma| = m$ words. A document $\boldsymbol{X}_i$ of length s_i is part of a collection of n documents and is a field of random variables X_{it}, whose realizations are words $x_{it} \in \Sigma, \forall t \in [s]$. We will assume that topical content is invariant to word order. Mathematically, this is known as exchangeability, which says that the distribution of a sequence of random variables does not change under any permutation of their order. As a consequence, the sufficient statistics of a document are the frequencies of occurrence of words in the documents. Hence, we can reduce the data to a bag-of-words representation of occurrence counts,

$$N_{ij} = |\{x_{it} = w_j \mid t \in [s_i]\}|.$$

In words, N_{ij} denotes how often word w_j occurred in document $\boldsymbol{x}_i$. We can thus summarize the full document corpus in an occurrence matrix,

$$\boldsymbol{N} = (N_{ij}) \in \mathbb{N}^{n \times m}.$$

Moreover, we have to conceptualize what we mean by "topic". Generally, topics refer to things that people are interested to talk or write about. Hence, we can represent topics by latent variables $Z \in [k]$, akin to what we saw in mixture models. We will then associate these topics with word occurrences, which induce topics of entire documents.[8] Thus, the complete data will be a sequence of word-topic pairs for each document,

[8] If we were to associate the topic variables with documents, we would effectively be performing document clustering. However, in this case, we want the topics of documents to be induced by the words that they contain.

$$(X_{it}, Z_{it}) \in \Sigma \times [k].$$

Note that topics are not mutually exclusive, since documents can concern multiple.

We will assume that the documents in the corpus were created according to the following generative process.[9] For each word, we sample a topic from $p(z \mid d)$, and then sample the word from $p(w \mid z)$. Thus, in order to define the latent variable data model, we need a distribution over latent variables, $p(z \mid d)$, and a topic-conditional distribution over words, $p(w \mid z)$. We can then define a document-conditional word distribution,

[9] This is important so we can infer/reverse engineer a model from it.

$$p(w_j \mid d_i) = \sum_{z=1}^{k} p(w_j \mid z) p(z \mid d_i).$$

From this, we define the log-likelihood objective as

$$\ell(\boldsymbol{\theta}; \boldsymbol{N}) = \log p(\boldsymbol{N}; \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{m} N_{ij} \log p(w_j \mid d_i).$$

Learning. We can equivalently write the log-likelihood objective in terms of the raw data,

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{t=1}^{s_i} \log p(x_{it}; \boldsymbol{\theta}), \quad p(x_{it}; \boldsymbol{\theta}) = \sum_{z=1}^{k} p(x_{it} \mid z) p(z \mid d_i).$$

Similarly to the mixture models, we define an ELBO,

$$\ell(\boldsymbol{\theta}) \geq \hat{\ell}(\boldsymbol{\theta}; \boldsymbol{q}) \doteq \sum_{i=1}^{n} \sum_{t=1}^{s_i} \sum_{z=1}^{k} q_{itz} (\log p(x_{it} \mid z) + \log p(z \mid d_i) - \log q_{itz}).$$

Moreover, following the same steps, we can derive the EM equations,

$$q_{itz} = \frac{p(x_{it} \mid z) p(z \mid d_i)}{\sum_{\zeta=1}^{k} p(x_{it} \mid \zeta) p(\zeta \mid d_i)}$$

E-step.

$$p(w_j \mid z) = \frac{\sum_{i=1}^{n} \sum_{t=1}^{s_i} q_{itz} \cdot \mathbb{1}\{x_{it} = w_j\}}{\sum_{i=1}^{n} \sum_{t=1}^{s_i} q_{itz}}, \quad p(z \mid d_i) = \frac{1}{s_i} \sum_{t=1}^{s_i} q_{itz}.$$

M-step.

Intuitively, the E-step computes the posterior probabilities, where $p(z \mid d_i)$ acts as a prior. This yields a probabilistic clustering of word occurrences. The M-step computes the MLE for a $\boldsymbol{q}$-weighted multinomial sample. This algorithm will converge, but not necessarily to the global maximizer.

As a result, we can use the class-conditional word distribution, $p(w \mid z)$, to find similar words that are connected by a common topic.

Latent Dirichlet allocation. The problem with the above topic model is that it assumes a fixed set of documents and selects the parameters to maximize the predictability of words within these. The natural next step is to extend the above in a way that accounts for modeling unseen documents.

The generative process of this model is that we first choose a distribution over topics, $p(z \mid \boldsymbol{\alpha})$. Then, for each word, sample a topic from $p(z \mid \boldsymbol{\alpha})$ and sample the word from $p(w \mid z)$.

The latent Dirichlet allocation (LDA) model takes a "Bayesian step up" from latent variable modeling. Its main idea is that it defines a distribution over mixture vectors,

$$\boldsymbol{v} \in \Delta^{k-1},$$

as a prior over topics. This parameter is distributed according to a Dirichlet distribution, and has hyperparameter $\boldsymbol{\alpha}$,

$$p(\boldsymbol{v}; \boldsymbol{\alpha}) \propto \prod_{z=1}^{k} v_z^{\alpha_z - 1}, \quad \alpha_z > 0, \forall z \in [k].$$

The Dirichlet distribution is chosen as the prior because it is a conjugate prior of the categorical distribution. Typically, we set $\alpha_k = \alpha$ and optimize α on held-out validation data.

Let $\boldsymbol{U} = [p(w_j \mid z)] \in [0,1]^{m \times k}$, then we have the following distribution,

$$p(\boldsymbol{X} = [x_1, \ldots, x_s] \mid \boldsymbol{U}) = \int \prod_{t=1}^{s} p(x_t \mid \boldsymbol{U}, \boldsymbol{v}) p(\boldsymbol{v}; \boldsymbol{\alpha}) \mathrm{d}\boldsymbol{v}$$
$$p(w_j \mid \boldsymbol{U}, \boldsymbol{v}) = \sum_{z=1}^{k} u_{jz} v_z.$$

As can be seen, the probabilities are not conditioned on the document. As a result, when a new document is introduced, we can use the same distributions, whose parameters were learned from an existing document collection. In this sense, the LDA model is more robust than the topic model that was initially introduced.

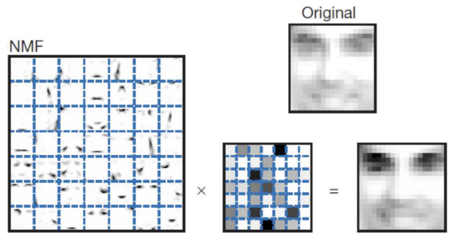

Figure 5. Factors identified by non-negative matrix factorization in a face reconstruction task.

Probabilistic matrix decomposition. The topic model introduced above is intimately related to matrix decomposition, where we see that the topic variable $z \in [k]$ plays the role of a rank constraint. More specifically, we define the following matrices,

$$\boldsymbol{U} \doteq [p(w_j \mid z)] \in [0,1]^{m \times k}, \quad \boldsymbol{V} \doteq [p(z \mid d_i)] \in [0,1]^{k \times n}.$$

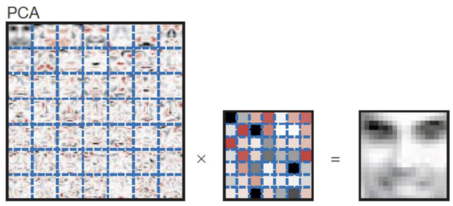

Figure 6. Factors identified by principal component analysis in a face reconstruction task.

We then form $\hat{\boldsymbol{N}} = \boldsymbol{U}\boldsymbol{V}$, which is a rank-$k$ matrix. We interpret the entries of this new matrix $\hat{\boldsymbol{N}}$ as

$$\hat{N}_{ji} = \sum_{z=1}^{k} p(w_j \mid z) p(z \mid d_i) = p(w_j \mid d_i).$$

Note that $\hat{\boldsymbol{N}}$ is a relative version of $\boldsymbol{N}$, where $N_{ij} \approx s_i \hat{N}_{ij}$. Thus, we have a matrix decomposition with additional constraints,

$$u_{jz} \geq 0, \forall j \in [m], z \in [k], \quad v_{zi} \geq 0, \forall i \in [n], z \in [k].$$

This is known as non-negative matrix factorization (NMF).[10] The objective follows from the maximum likelihood principle,

$$\ell(\boldsymbol{U}, \boldsymbol{V}; \boldsymbol{N}) = \sum_{i=1}^{n} \sum_{j=1}^{m} N_{ij} \log \hat{N}_{ij}, \quad \hat{\boldsymbol{N}} = \boldsymbol{U}\boldsymbol{V}.$$

Further, we have normalization constraints on the row and column level, namely

$$\sum_{j=1}^{m} u_{jz} = 1, \forall z \in [k], \quad \sum_{z=1}^{k} v_{zi} = 1, \forall i \in [n].$$

Thus, we have a special case of non-negative matrix decomposition with a log-likelihood objective.

[10] For NMF, we can use the ALS algorithm, where we add projection steps to enforce non-negativity,

$$u_{jz} = \max\{0, u_{jz}\}, \quad v_{zi} = \max\{0, v_{zi}\}.$$

Figures 5 and 6 show identified factors of NMF and PCA on a face reconstruction task. As can be seen, NMF tends to identify part-based representations and its features are sparse, because there is no way of removing added features, due to the non-negativity constraint. In other words, NMF models must be careful in what they add.

3.3 Embeddings

In natural language, the atomic units of meaning are symbols, such as words. Generally, these symbols do not carry their meaning with them. Rather their meaning come from their use within its language. The idea of embeddings is to learn representations that capture semantics by embedding symbols in vector space. Specifically, we want to learn a mapping from words to vectors such that the vectors represent word semantics.

The latent variable approach to this problem is to learn latent representations that are predictive of observations. Thus, we need to design a task such that the latent representations of the words must have some form of semantics to be able to perform the task well. In the skip-gram model, we use the task of predicting whether a word is in the context of another.[11] Effectively, we are treating the embedding of each word as a latent variable that predicts the co-occurring of context words. After the model has converged, we do not care about the performing the task itself well, but rather the parameters (the embeddings) of the model that lead to the task being performed well.

[11] Rupert Firth: "You shall know a word by the company it keeps."

This task has the following likelihood function that we wish to maximize,

$$\ell(\boldsymbol{\theta}; \boldsymbol{x}) = \sum_{t=1}^{T} \sum_{\delta \in \mathcal{I}} \log p(x_{t+\delta} \mid x_t; \boldsymbol{\theta}),$$

where $\boldsymbol{\theta}$ contains the embeddings as parameters of this model and $\mathcal{I}$ is a set of displacements, *e.g.*, $\mathcal{I} = \{-R, \dots, -1, 1, \dots, R\}$. The probabilities are computed by normalized inner products,

$$p(v \mid w; \boldsymbol{\theta}) = \frac{\exp\langle \boldsymbol{z}_w, \boldsymbol{z}_v \rangle}{\sum_{u \in \Sigma} \exp\langle \boldsymbol{z}_w, \boldsymbol{z}_u \rangle}.$$

We can further refine this by introducing biases $b_v \in \mathbb{R}$ to explicitly control the marginal probability and using different embeddings for conditioned and predicted words,[12] leading to

[12] This is necessary, because words are often not within their own context window, but will have a high inner product, because their vectors are equal.

$$p(v \mid w; \boldsymbol{\theta}) = \frac{\exp(\langle \boldsymbol{\zeta}_w, \boldsymbol{z}_v \rangle + b_v)}{\sum_{u \in \Sigma} \exp(\langle \boldsymbol{\zeta}_w, \boldsymbol{z}_u \rangle + b_u)}$$

with parameters

$$\boldsymbol{\theta}_w = (\boldsymbol{z}_w, \boldsymbol{\zeta}_w, b_w) \in \mathbb{R}^{2m+1}.$$

Sufficient statistics for this model is a co-occurrence matrix, where

$$N_{vw} = |\{t \mid x_t = w, x_{t+\delta} = v, \delta \in \mathcal{I}\}|.$$

The log-likelihood can then be computed by

$$\ell(\boldsymbol{\theta}; \boldsymbol{x}) = \sum_{v \in \Sigma} \sum_{w \in \Sigma} N_{vw} \left(\langle \boldsymbol{\zeta}_w, \boldsymbol{z}_v \rangle + b_v - \log \sum_{u \in \Sigma} \exp(\langle \boldsymbol{\zeta}_w, \boldsymbol{z}_u \rangle + b_u) \right).$$

The problem with this approach is that computing the normalization constant is expensive. We can discard normalization by reformulating the prediction problem as a classification problem. For this, we also need positive samples,

$$\mathcal{S}^+ = [(x_t, x_{t+\delta}) \mid t \in [T], \delta \in \mathcal{I}].$$

The [] notation indicates a multiset.

And, additionally we need negative samples, which we randomly sample,

$$\mathcal{S}^- = [(x_t, v_{tj}) \mid t \in [T], v_{tj} \overset{\text{iid}}{\sim} q, j \in [r]],$$

where q is a probability over words and r is the sampling factor (how many more negative samples we have than positive samples). Generally, q is chosen to satisfy

$$q(w) \propto p(w)^\alpha,$$

where typically $\alpha = 3/4$. The intuition behind this is that what matters most in learning semantic representations is not the very frequent words, which carry little meaning, and also not the infrequent words, but the in-between range. This choice of α makes them more likely to be sampled, as shown in Figure 7.

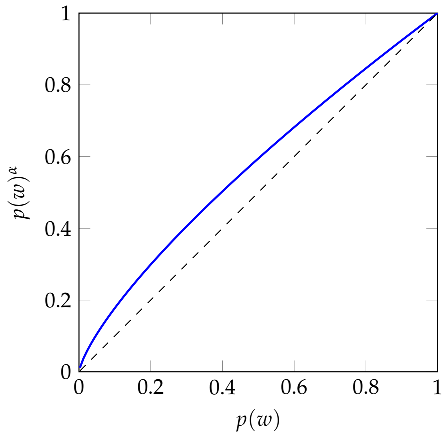

Figure 7. Plot of $p(w)^\alpha$ for $\alpha = 3/4$.

We can then define a logistic log-likelihood function that we wish to minimize,

$$\ell(\boldsymbol{\theta}, \boldsymbol{x}) = \sum_{(w,v)\in\mathcal{S}^+} \log \sigma(v, w; \boldsymbol{\theta}) + \sum_{(w,u)\in\mathcal{S}^-} \log(1 - \sigma(u, w; \boldsymbol{\theta})).$$

Pointwise mutual information. Let $p(v, w)$ denote the true distribution of co-occurring words and by $q(v, w) = p(w)p(v)$ the distribution used for negative sampling. The optimal Bayesian classifier would then be

$$\mathbb{P}((v, w) = \text{true}) \frac{\pi p(v, w)}{\pi p(v, w) + (1 - \pi) q(v, w)},$$

where π is the prior probability of a true pair. Considering the pre-image (logit) of the logistic function, we get

$$\begin{aligned}
h^\star_{vw} &= \sigma^{-1}\left(\frac{\pi p(v, w)}{\pi p(v, w) + (1 - \pi) q(v, w)}\right) \\
&= \log\left(\frac{\pi p(v, w)}{\pi p(v, w) + (1 - \pi) q(v, w)} \cdot \frac{\pi p(v, w) + (1 - \pi) q(v, w)}{(1 - \pi) q(v, w)}\right) \\
&= \log\left(\frac{\pi p(v, w)}{(1 - \pi) q(v, w)}\right) \\
&= \log \frac{p(v, w)}{q(v, w)} + \log \frac{\pi}{1 - \pi}.
\end{aligned}$$

$\sigma^{-1}(p) = \log \frac{p}{1-p}, \quad p \in (0, 1).$

Hence, in the case of balanced classes $\pi = 1/2$ (equivalent to $r = 1$) and $\alpha = 1$, we get

$$h^\star_{vw} = \log \frac{p(v, w)}{p(w)p(v)},$$

which is the pointwise mutual information.

GloVe. Global word vectors (GloVe) considers word embedding models in terms of matrix factorization. It maximizes a different objective,

$$\ell(\boldsymbol{\theta}, \boldsymbol{N}) = \sum_{\substack{v,w\in\Sigma \\ N_{vw}>0}} f(N_{vw}) \big(\log N_{vw} - \log \hat{N}_{vw}\big)^2, \quad \hat{N}_{vw} = p(v, w; \boldsymbol{\theta}),$$

which is a weighted square loss on the log-scale. In practice, the following weighting function is used,

$$f(N) = \min\left\{1, \left(\frac{N}{N_{\max}}\right)^{\alpha}\right\},$$

where often $\alpha = 3/4$.

The idea of this objective is that the we can work with an unnormalized conditional probability distribution and simply choose

$$\log p(v, w) = \langle \boldsymbol{\zeta}_w, \boldsymbol{z}_v \rangle.$$

The reason for this is that the squared objective is two-sided, whereas a likelihood objective will always increase if we increase probabilities and the balancing effect comes purely from the normalization.

GloVe can be interpreted as a low-rank matrix factorization with

$$\boldsymbol{U} \doteq [\boldsymbol{\zeta}_1, \ldots, \boldsymbol{\zeta}_n]^\top, \quad \boldsymbol{V} \doteq [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n]^\top.$$

Then, we have the following matrix of unnormalized probabilities,

$$\log \hat{\boldsymbol{N}} = \boldsymbol{U}\boldsymbol{V}^\top.$$

The GloVe objective is a weighted Frobenius norm of the approximation residual between the observed log-count matrix and a low-rank factorization of embedding matrices. As a special case, consider

$$f(N) = \min\{1, N\},$$

which results in a matrix completion problem,

$$\begin{aligned} \boldsymbol{U}, \boldsymbol{V} &\in \operatorname*{argmin}_{\boldsymbol{U}, \boldsymbol{V}} \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbb{1}\{N_{ij} > 0\} \left(\log N_{ij} - \left(\boldsymbol{U}\boldsymbol{V}^\top\right)_{ij}\right) \\ &= \left\| \Pi_{\mathbb{1}\{\boldsymbol{N} > 0\}} \left(\log \boldsymbol{N} - \boldsymbol{U}\boldsymbol{V}^\top\right) \right\|_F^2. \end{aligned}$$

This is a matrix completion problem with $\boldsymbol{A} = \log \boldsymbol{N}$ and $\omega_{ij} = \mathbb{1}\{N_{ij} > 0\}$.

We can optimize $\boldsymbol{U}$ and $\boldsymbol{V}$ by stochastic gradient descent,

$$\begin{aligned} \boldsymbol{\zeta}_w &\leftarrow \boldsymbol{\zeta}_w + 2\eta f(N_{vw})(\log N_{vw} - \langle \boldsymbol{\zeta}_w, \boldsymbol{z}_v \rangle)\boldsymbol{z}_v \\ \boldsymbol{z}_v &\leftarrow \boldsymbol{z}_v + 2\eta f(N_{vw})(\log N_{vw} - \langle \boldsymbol{\zeta}_w, \boldsymbol{z}_v \rangle)\boldsymbol{\zeta}_w, \end{aligned}$$

where we sample (v, w) at random.

4 Deep neural networks

Generally, deep learning models consist of a function $H : \mathbb{R}^n \to \mathbb{R}^p$ that extracts p-dimensional features from n-dimensional data and a linear map g that makes the final prediction. The final learned function is then formalized by $\psi = g \circ H$. Machine learning philosophies differ in the way that they extract features from the data. There are three main philosophies,

- Feature engineering: H is hand-crafted to extract intuitive features that have good predictive power of the label. The problem with this approach is that it requires domain expertise;
- Feature expansion: H maps to a high-dimensional feature space using kernels and implicit models;
- Compositionality: The feature extraction function H is learned through the composition of L simple building blocks,

$$H = H_L \circ H_{L-1} \circ \cdots \circ H_2 \circ H_1, \quad H_l : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}.$$

In compositional models, the partial maps $H_{1:l} \doteq H_l \circ \cdots \circ H_1$ produce intermediate representations. These satisfy the Markov property and, as such, need to preserve task-relevant information. Once information is lost, it cannot be recovered. The idea of the layers is to make relevant information more accessible and explicit with increasing depth, such that g can easily make a prediction.[13]

[13] For example, the early layers of CNN typically learn low-level features, whereas the later layers will learn higher level features that make use of the low-level features.

The key question is how to define the basic building blocks such that their composition is more powerful than any one block can be. Consider two linear layers,

$$F(\boldsymbol{x}; \boldsymbol{W}_2) = \boldsymbol{W}_2\boldsymbol{x}, \quad G(\boldsymbol{x}; \boldsymbol{W}_1) = \boldsymbol{W}_1\boldsymbol{x}.$$

Their composition is again a linear layer,

$$(F \circ G)(\boldsymbol{x}) = \boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x} = \boldsymbol{W}\boldsymbol{x}, \quad \boldsymbol{W} = \boldsymbol{W}_2\boldsymbol{W}_1.$$

So, linear layers are not appropriate building blocks.

The key idea is to combine a linear map with a non-linearity,

$$H(\boldsymbol{x}; \boldsymbol{W}) = \Phi(\boldsymbol{W}\boldsymbol{x}),$$

where Φ is a non-linear element-wise activation function,

$$\Phi(\boldsymbol{z}) = [\phi(z_1), \ldots, \phi(z_m)], \quad \phi : \mathbb{R} \to \mathbb{R}.$$

Theoretically, a neural network with one hidden layer and a non-polynomial activation function is a universal function approximator. This means that any function can be represented using a single hidden layer and a non-linear activation function.[14] However, in practice, this single hidden layer

[14] The following are commonly used activation functions,

- Sigmoid, $\sigma : \mathbb{R} \to (0, 1)$,
$$\sigma(z) \doteq \frac{1}{1 + e^{-z}};$$
- Hyperbolic tangent, $\tanh : \mathbb{R} \to (-1, 1)$,
$$\tanh(z) \doteq \frac{e^z - e^{-z}}{e^z + e^{-z}} = 2\sigma(2z) - 1;$$
- Rectified linear unit, $\mathrm{ReLU} : \mathbb{R} \to \mathbb{R}_{\geq 0}$,
$$\mathrm{ReLU}(z) \doteq \max\{0, z\}.$$

may need to be infinitely large. A single hidden layer multi-layer perceptron (MLP) is formalized by the following function,

$$\psi(\boldsymbol{x};\boldsymbol{\beta},\boldsymbol{W}) = \boldsymbol{\beta}^\top \sigma(\boldsymbol{W}\boldsymbol{x}) = \sum_{j=1}^m \beta_j \sigma(\boldsymbol{w}_j^\top \boldsymbol{x}) = \sum_{j=1}^m \frac{\beta_j}{1+\exp\left(-\boldsymbol{w}_j^\top \boldsymbol{x}\right)}.$$

In this case, $g(\boldsymbol{y};\boldsymbol{\beta}) = \boldsymbol{\beta}^\top \boldsymbol{y}$ and $H(\boldsymbol{x};\boldsymbol{W}) = \boldsymbol{W}\boldsymbol{x}$.

In order to tune this model, we need to be able to compute its gradients, which tell us how to locally optimize a loss function ℓ. In this case, we choose a squared loss, $\ell(\hat{y},y) = \frac{1}{2}(\hat{y}-y)^2$. The gradients are computed by

$$\begin{aligned}
\frac{\partial}{\partial \beta_j}\frac{1}{2}(\psi(\boldsymbol{x})-y)^2 &= (\psi(\boldsymbol{x})-y)\frac{\partial}{\partial \beta_j}\psi(\boldsymbol{x}) \\
&= \frac{\psi(\boldsymbol{x})-y}{1+\exp\left(-\boldsymbol{w}_j^\top \boldsymbol{x}\right)} \\
\frac{\partial}{\partial w_{ji}}\frac{1}{2}(\psi(\boldsymbol{x})-y)^2 &= (\psi(\boldsymbol{x})-y)\frac{\partial}{\partial w_{ji}}\psi(\boldsymbol{x}) \\
&= (\psi(\boldsymbol{x})-y)\beta_j \frac{\partial}{\partial w_{ji}}\sigma\left(\boldsymbol{w}_j^\top \boldsymbol{x}\right) \\
&= (\psi(\boldsymbol{x})-y)\beta_j \sigma\left(\boldsymbol{w}_j^\top \boldsymbol{x}\right)\sigma\left(-\boldsymbol{w}_j^\top \boldsymbol{x}\right)\frac{\partial}{\partial w_{ji}}\boldsymbol{w}_j^\top \boldsymbol{x} \\
&= \frac{\psi(\boldsymbol{x})-y}{1+\exp\left(-\boldsymbol{w}_j^\top \boldsymbol{x}\right)}\frac{\beta_j x_i}{1+\exp\left(\boldsymbol{w}_j^\top \boldsymbol{x}\right)}.
\end{aligned}$$

$\sigma'(z) = \sigma(z)(1-\sigma(z)) = \sigma(z)\sigma(-z)$.

Learning then typically involves by stochastic gradient descent, which iteratively selects a random mini-batch $\mathcal{S}_t \subseteq \mathcal{S} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$. The update rule is then

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \sum_{(\boldsymbol{x},y)\in\mathcal{S}_t} \frac{\partial \ell(\psi(\boldsymbol{x};\boldsymbol{\theta}), y)}{\partial \boldsymbol{\theta}}.$$

4.1 Backpropagation

Calculating the gradient by hand for every model is very tedious and time consuming. Backpropagation is an algorithm that can compute the gradient of any function, which consists of building blocks with known gradients, in linear time. This algorithm makes use of dynamic programming, which means that it breaks the problem down into smaller subproblems and re-uses solutions to previously seen subproblems. In this case, the solution to the subproblems are the gradients, which can be re-used in later gradients by the chain rule and sum rule. In short, backpropagation exploits compositionality to efficiently compute the gradient.

Let $H_k : \mathbb{R}^n \to \mathbb{R}^m$ be an intermediate layer of a compositional model. It's Jacobi map is defined as

$$[\boldsymbol{J}_k]_{ij} \doteq \frac{\partial h_{ki}}{\partial z_j},$$

h_{ki} is the i-th output of layer k and z_j is its j-th input.

which is an implicit function of the input $z \in \mathbb{R}^n$ to H_k. Furthermore, define the error signal as

$$\delta_k \doteq \left[\frac{\partial \ell}{\partial H_k}\right]^\top.$$

Using the chain rule, we can find a recurrence relation between error signals,

$$\delta_k \doteq \left[\frac{\partial \ell}{\partial H_k}\right]^\top = \left[\frac{\partial \ell}{\partial H_{k+1}}\frac{\partial H_{k+1}}{\partial H_k}\right]^\top \doteq \boldsymbol{J}_{k+1}^\top \delta_{k+1}.$$

Lastly, in order to compute the gradient w.r.t. the parameters, we use the chain rule again,

$$\begin{aligned}
\frac{\partial \ell}{\partial [\boldsymbol{W}_k]_{ij}} &= \frac{\partial \ell}{\partial H_k}\frac{\partial H_k}{\partial [\boldsymbol{W}_k]_{ij}} \\
&= \frac{\partial \ell}{\partial H_k}\left[\frac{\partial \phi(\boldsymbol{y})}{\partial \boldsymbol{y}}\bigg|_{\boldsymbol{y}=\boldsymbol{W}_k \boldsymbol{z}_{k-1}}\right]\frac{\partial}{\partial [\boldsymbol{W}_k]_{ij}}\boldsymbol{W}_k \boldsymbol{z}_{k-1} \\
&= \delta_k^\top \operatorname{diag}\big(\phi'(\boldsymbol{W}_k \boldsymbol{z}_{k-1})\big)\operatorname{vec}_i([\boldsymbol{z}_{k-1}]_j) \\
&= [\delta_k]_i \cdot \phi'\Big([\boldsymbol{W}_k]_i^\top \boldsymbol{z}_{k-1}\Big) \cdot [\boldsymbol{z}_{k-1}]_j.
\end{aligned}$$

Intuitively, the local parameter gradient $\frac{\partial \ell}{\partial \boldsymbol{W}_k}$ is the product of an upstream vector $\boldsymbol{z}_{k-1}$, a downstream error signal δ_k, and the local sensitivity of the unit $\phi'(\boldsymbol{W}_k \boldsymbol{z}_{k-1})$. Note that we first need to perform a forward pass in order to compute these gradients in the backward pass.

Moreover, we have the following Jacobi maps for different activation functions,

- Linear activation, $\boldsymbol{J}_k = \boldsymbol{W}$;
- ReLU layer, $\boldsymbol{J}_k = \operatorname{diag}(\mathbb{1}\{\boldsymbol{W}\boldsymbol{z} > 0\})\boldsymbol{W}$;
- Sigmoid layer, $\boldsymbol{J}_k = \operatorname{diag}(\sigma'(\boldsymbol{W}\boldsymbol{z}))\boldsymbol{W}$.

4.2 *Gradient methods*

In gradient descent, we iteratively update the parameters by

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \boldsymbol{\nabla}\ell(\boldsymbol{\theta}_k), \quad \eta > 0.$$

A fundamental question is whether gradient descent will converge to an optimal solution. The key intuition is that gradient descent can only work if gradient does not change too much relative to the step size. The gradient information must remain informative in a neighborhood around a point.

Definition 16 (Smoothness)**.** A differentiable function $\ell : \mathbb{R}^d \to \mathbb{R}$ is L-smooth for some $L > 0$ if

$$\|\boldsymbol{\nabla}\ell(\boldsymbol{\theta}) - \boldsymbol{\nabla}\ell(\boldsymbol{\theta}')\| \le L\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|, \quad \forall \boldsymbol{\theta}, \boldsymbol{\theta}'.$$

This is equivalent to the gradient being L-Lipschitz continuous.

Smoothness implies a bound on the Hessian,

$$\|\nabla^2 \ell(\boldsymbol{\theta})\| \leq L, \quad \forall \boldsymbol{\theta}.$$

As a consequence, this means that smoothness bounds the largest eigenvalue of the Hessian. A large L means that the gradient can change quickly, making it more unstable, thus we need to lower the stepsize. This is intuitively why the stepsize $\eta = 1/L$ works well.

Definition 17 (ϵ-critical point)**.** Let ℓ be differentiable at $\boldsymbol{\theta}$, then $\boldsymbol{\theta}$ is an ϵ-critical point if

$$\|\nabla \ell(\boldsymbol{\theta})\| \leq \epsilon.$$

Lemma 18. Gradient descent on an L-smooth, differentiable function $\ell : \mathbb{R}^d \to \mathbb{R}$ with step size $\eta = 1/L$ finds an ϵ-critical point in at most

$$k = \frac{2L(\ell(\boldsymbol{\theta}_0) - \ell^\star)}{\epsilon^2}$$

steps.

Thus, smoothness is sufficient to find local minima. The question is what properties of ℓ will ensure convergence to a global minimum.

Definition 19 (Polyak-Lojasiewicz condition)**.** A differentiable function $\ell : \mathbb{R}^d \to \mathbb{R}$ satisfies the Polyak-Lojasiewicz (PL) condition with parameter $\mu > 0$ if

$$\frac{1}{2}\|\nabla \ell(\boldsymbol{\theta})\|^2 \geq \mu(\ell(\boldsymbol{\theta}) - \ell^\star), \quad \forall \boldsymbol{\theta}.$$

Lemma 20. Let ℓ be differentiable, L-smooth, and μ-PL. Then, gradient descent with stepsize $\eta = 1/L$ converges at a geometric rate,

$$\ell\left(\boldsymbol{\theta}^{(k)}\right) - \ell^\star \leq \left(1 - \frac{\mu}{L}\right)^k (\ell(\boldsymbol{\theta}_0) - \ell^\star).$$

Definition 21 (Strong convexity)**.** A differentiable function $\ell : \mathbb{R}^d \to \mathbb{R}$ is μ-strongly convex for some $\mu > 0$ if

$$\ell(\boldsymbol{\theta}') \geq \ell(\boldsymbol{\theta}) + \langle \nabla \ell(\boldsymbol{\theta}), \boldsymbol{\theta}' - \boldsymbol{\theta} \rangle + \frac{\mu}{2}\|\boldsymbol{\theta}' - \boldsymbol{\theta}\|^2, \quad \forall \boldsymbol{\theta}, \boldsymbol{\theta}'.$$

Intuitively, strong convexity bounds the smallest eigenvalue of a locally quadratic approximation of ℓ.

Lemma 22 (Strong convexity $\Rightarrow$ PL)**.** Let ℓ be μ-strongly convex, then it fulfills the PL condition with the same μ.

Thus, strong convexity ensures convergence to a global optimum.

Momentum. The heavy ball method is an optimization algorithm with momentum,

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \boldsymbol{\nabla} \ell(\boldsymbol{\theta}_k) + \beta(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}), \quad \beta \in (0, 1).$$

Intuitively, we are acting as if the iterates have mass. As a consequence, we will pass small gradient areas faster, and thus overcomes converging to local minima.

Adaptivity. AdaGrad uses the gradient history to adapt the stepsize per dimension,

$$[\boldsymbol{\theta}_{k+1}]_i = [\boldsymbol{\theta}_k]_i - \eta_i^k \boldsymbol{\nabla}_i \ell(\boldsymbol{\theta}_k), \quad \eta_i^k \doteq \frac{\eta}{\sqrt{\gamma_i^k + \delta}},$$

where

$$\gamma_i^k = \gamma_i^{k-1} + (\boldsymbol{\nabla}_i \ell(\boldsymbol{\theta}_k))^2.$$

Acceleration. Nesterov's accelerated gradient descent has the following update rule,

$$\begin{aligned} \boldsymbol{\theta}'_{k+1} &= \boldsymbol{\theta}_k + \beta(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}) \\ \boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta}'_{k+1} - \eta \boldsymbol{\nabla} \ell(\boldsymbol{\theta}'_{k+1}). \end{aligned}$$

While it is not intuitive why this works, it provides a faster convergence rate than vanilla gradient descent.

Adam. Adam combines momentum and adaptivity to increase convergence speed. It defines the following variables,

$$\begin{aligned} \boldsymbol{g}_k &= \beta \boldsymbol{g}_{k-1} + (1 - \beta) \boldsymbol{\nabla} \ell(\boldsymbol{\theta}_k), \quad \beta \in [0, 1] \\ \boldsymbol{h}_k &= \alpha \boldsymbol{h}_{k-1} + (1 - \alpha) \boldsymbol{\nabla} \ell(\boldsymbol{\theta}_k)^{\odot 2}, \quad \alpha \in [0, 1]. \end{aligned}$$

The update rule is then

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \boldsymbol{\eta}_k \odot \boldsymbol{g}_k, \quad \boldsymbol{\eta}_k = \frac{\eta}{\sqrt{\boldsymbol{h}_k} + \delta}.$$

4.3 *Convolutional neural networks*

Images and audio have an extremely high dimensionality, which means that a single linear layer would have an extremely high parameter count. However, we can exploit the locality, scale, and shift invariance of these types of data to define a new operator; the convolution.

Definition 23 (Convolution)**.** Given two functions f, h, their convolution is defined as

$$(f * h)(u) \doteq \int_{-\infty}^{\infty} h(u - t) f(t) \mathrm{d}t = \int_{-\infty}^{\infty} f(u - t) h(t) \mathrm{d}t.$$

The convolution operator is shift invariant, meaning that if we shift and then apply the operator, we get the same result as if we were to first apply the operator and then shift. Formally,

$$(f * h)(u + \Delta) = (f * h)(u) + \Delta.$$

The converse is also true: any linear shift-invariant transformation can be written as a convolution.

In practice, we have discrete data, which means that we have to define a discrete convolution operator,

$$(f * h)[u] \doteq \sum_{t=-\infty}^{\infty} f[t]h[u - t].$$

This easily extends to two dimensions,

$$(f * h)[x, y] = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} f[u, v]h[x - u, y - v].$$

Typically, f and h are defined over a finite domain.

The cross-correlation operator is equal to a convolution with a flipped kernel,

$$(f \star h)[u] \doteq \sum_{t=-\infty}^{\infty} h[t]f[u + t].$$

Convolutional neural networks (CNN) learn the kernel of convolutional layers and stacks them in a compositional way to extract features from images. In this way, these layers exploit the shift invariance, locality, and scale of the data. Furthermore, it increases the statistical efficiency w.r.t. MLPs, because of the shared parameters.

As MLPs, CNNs alternate between (linear) convolutional layers and non-linear element-wise functions to increase model capacity. Moreover, it employs max-pooling layers to decrease the dimensionality of the input. It does so by taking only the maximum in every $k \times k$ patch. This has the effect that the input is subsampled k times. After many max-pooling layers, the data will no longer be location dependent, which allows us to throw away spatial information by flattening the data. From there, linear layers can be used to make the final prediction.