Data visualization
Data visualization

• Prior to choosing a particular ML model, it can sometimes be helpful to visualize your dataset.

• One of the most commonly used visualization techniques is called principal component analysis (PCA).
Data visualization

• Consider the MNIST dataset of hand-written digits.

• Visualizing each *individual* example is easy, e.g.:

  ![](image1)

• But this doesn’t tell us a whole lot about the dataset as a *whole*, or how *separable* the classes are.
Data visualization

• Each MNIST image is 28x28 pixels ⇒ 784 dimensions.

• To show all examples in the original input space, we would need a 784-dimensional visualization.
  
  • But humans struggle with perception beyond 3-D.

• How can we represent a collection of many high-dimensional images in just 2-D or 3-D?
Data visualization

• We somehow have to condense the interesting information of a 784-dim vector into just 2-3 dimensions!

• Core question:

  • How do we pick the dimensions — or more generally, *directions* — of data to present?
Data visualization

• Let’s give a naive try…

• For each image $\mathbf{x}^{(i)}$ (where $i=1, \ldots, n$):
• Let’s give a naive try…

• For each image $\mathbf{x}^{(i)}$ (where $i=1, \ldots, n$):
  
  • Retrieve the values of just the first two pixels, i.e., $(\mathbf{x}^{(i)}_1, \mathbf{x}^{(i)}_2)$.
Data visualization

- Let’s give a naive try…

- For each image \( \mathbf{x}^{(i)} \) (where \( i=1, \ldots, n \)):
  
  - Retrieve the values of just the first two pixels, i.e., \( (\mathbf{x}^{(i)}_1, \mathbf{x}^{(i)}_2) \).
  
  - Plot the point \( (\mathbf{x}^{(i)}_1, \mathbf{x}^{(i)}_2) \) in 2-D space.
MNIST 2-D visualization: first two pixel values

• Let’s apply this procedure to 2500 images from the MNIST test set…
MNIST 2-D visualization: first two pixel values

- Let’s apply this procedure to 2500 images from the MNIST test set…

What happened?
The problem is that, for all \( n \) images in our dataset, the value of the first two pixels is 0!

There was very little (actually, 0) \textit{variance} across each of these two dimensions.
Variance

- Intuitively, the **variance** of a vector of $n$ numbers is how “spread out” they are:
Variance

- Note that the variance is independent of the mean!
Variance

- From basic statistics:
  - **Mean** of an $n$-dimensional vector: sum and divide by $n$:
    
    $$
    \mathbb{E}(p) = \frac{1}{n} \sum_{i=1}^{n} p_i
    $$

Variance

- From basic statistics:
  - **Mean** of an \( n \)-dimensional vector: sum and divide by \( n \):
    \[
    \mathbb{E}(p) = \frac{1}{n} \sum_{i=1}^{n} p_i
    \]
  - **Variance** of an \( n \)-dimensional vector: the mean squared distance from the mean:
    \[
    \mathbb{V}(p) = \frac{1}{n} \sum_{i=1}^{n} (p_i - \mathbb{E}(p))^2
    \]
Variance

• Alternatively, we can compute the variance in two steps:

  • First subtract the mean from each element of \( p \):

\[
\tilde{p} = p - \mathbb{E}(p) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}
\]
Variance

- Alternatively, we can compute the variance in two steps:
  
  - First subtract the mean from each element of $p$:
    
    $$ \tilde{p} = p - \mathbb{E}(p) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} $$

  - Since the mean of $\tilde{p}$ is 0, we can compute its variance as:
    
    $$ \mathbb{V}(\tilde{p}) = \frac{1}{n} \sum_{i=1}^{n} (\tilde{p}_i - \mathbb{E}(\tilde{p}))^2 $$
Variance

• Alternatively, we can compute the variance in two steps:

  • First subtract the mean from each element of $\mathbf{p}$:

  $$
  \tilde{\mathbf{p}} = \mathbf{p} - \mathbb{E}(\mathbf{p}) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}
  $$

  • Since the mean of $\tilde{\mathbf{p}}$ is 0, we can compute its variance as:

  $$
  \mathbb{V}(\tilde{\mathbf{p}}) = \frac{1}{n} \sum_{i=1}^{n} (\tilde{p}_i - \mathbb{E}(\tilde{\mathbf{p}}))^2
  = \frac{1}{n} \sum_{i=1}^{n} (\tilde{p}_i - 0)^2
  $$
Variance

• Alternatively, we can compute the variance in two steps:

  • First subtract the mean from each element of \( \mathbf{p} \):

    \[
    \tilde{\mathbf{p}} = \mathbf{p} - \mathbb{E}(\mathbf{p}) \begin{bmatrix} 1 \\ \\ \vdots \\ 1 \end{bmatrix}
    \]

  • Since the mean of \( \tilde{\mathbf{p}} \) is 0, we can compute its variance as:

    \[
    \mathbb{V}(\tilde{\mathbf{p}}) = \frac{1}{n} \sum_{i=1}^{n} (\tilde{p}_i - \mathbb{E}(\tilde{\mathbf{p}}))^2
    \]

    \[
    = \frac{1}{n} \sum_{i=1}^{n} (\tilde{p}_i - 0)^2
    \]

    \[
    = \frac{1}{n} \sum_{i=1}^{n} (\tilde{p}_i)^2
    \]
Variance

• Alternatively, we can compute the variance in two steps:

  • First subtract the mean from each element of $\mathbf{p}$:

    $$\tilde{\mathbf{p}} = \mathbf{p} - \mathbb{E}(\mathbf{p}) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

  • Since the mean of $\tilde{\mathbf{p}}$ is 0, we can compute its variance as:

    $$\nabla(\tilde{\mathbf{p}}) = \frac{1}{n} \sum_{i=1}^{n} (\tilde{p}_i - \mathbb{E}(\tilde{\mathbf{p}}))^2$$

    $$= \frac{1}{n} \sum_{i=1}^{n} (\tilde{p}_i - 0)^2$$

    $$= \frac{1}{n} \sum_{i=1}^{n} (\tilde{p}_i)^2$$

    $$= \frac{1}{n} \tilde{\mathbf{p}}^\top \tilde{\mathbf{p}}$$
MNIST 2-D visualization: two most highly-varying dimensions

• Let’s search for the two pixel dimensions along which the images vary the most.

• Selecting a particular pixel from each image $\mathbf{x}$ is equivalent to projecting each $\mathbf{x}$ onto a unit vector.

$$\mathbf{x}^T \mathbf{d}$$

where $\mathbf{d}$ is a vector of $n-1$ zeros and 1 one (whose location corresponds to the particular pixel location):

$$\mathbf{d} = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$
Projection

- Recall that a projection of a vector $\mathbf{v}$ onto a direction (unit vector) $\mathbf{d}$ is given by $p = \mathbf{v}^T \mathbf{d}$:
Projection

• Recall that a (scalar) projection of a vector \( \mathbf{v} \) onto a direction (unit vector) \( \mathbf{d} \) is given by \( p = \mathbf{v}^T \mathbf{d} \):

\[
\begin{align*}
\mathbf{v} \\
\downarrow \quad \downarrow \\
\mathbf{d} \quad \quad p
\end{align*}
\]

• The scalar projection \( p \) measures the distance of \( \mathbf{v} \) along \( \mathbf{d} \).
MNIST 2-D visualization: two most highly-varying dimensions

- For each possible pixel dimension $d$, we can obtain the $n$-vector of all scalar projections (over all $n$ images) as:

  $$ p = X^\top d $$

- We can then calculate the variance of the scalar projections by calculating $\text{Var}(p)$.

- By searching over all 784 possible dimensions, we can find the two dimensions along which variance is maximized.
When we apply this to MNIST, we get:

- Pixel dimension of 1st-highest variance: \((r,c) = (13,14)\)
- Pixel dimension of 2nd-highest variance: \((r,c) = (14,14)\)
MNIST 2-D visualization: two most highly-varying dimensions

- Certainly much better!
- However, many of the values overlap each other due to saturation — in many images, these pixels’ values are maximized/minimized.
Beyond axis-aligned directions

- But why constrain ourselves to only axis-aligned unit vectors, i.e., vectors with $m-1$ zeros and 1 one?
- Consider the following dataset in which each image contains just 2 pixels.
- Along which direction $d$ is variance maximized?
Beyond axis-aligned directions

• But why constrain ourselves to only axis-aligned unit vectors, i.e., vectors with \(m-1\) zeros and 1 one?

• Consider the following dataset in which each image contains just 2 pixels.

• Along which direction \(\mathbf{d}\) is variance maximized?

\[
\mathbf{d} = \left[ \frac{1}{\sqrt{2}} \right] \left[ \frac{1}{\sqrt{2}} \right]
\]
Maximizing variance along any direction $d$

- In general, given a dataset $X$ of training examples, we want to find the direction $d$ that maximizes $\text{Var}(X^Td)$.

- For simplicity, let’s assume that the mean of $X$, along each pixel dimension $j$, is 0, i.e.:

  $$\frac{1}{n} \sum_{i=1}^{n} x_j^{(i)} = 0$$

- (If this is not the case, then just subtract off the mean vector from each example $x^{(i)}$.)
Maximizing variance along any direction \( d \)

- Since \( X \) has zero mean (for each pixel dimension \( j \)), then \( X^T d \) also has zero mean (for any \( d \)):

\[
\mathbb{E} \left( X^T d \right) = \frac{1}{n} \sum_{i=1}^{n} x^{(i)}^T d
\]

by applying the definition of mean.
Maximizing variance along any direction $d$

- Since $X$ has zero mean (for each pixel dimension $j$), then $X^T d$ also has zero mean (for any $d$):

$$
\mathbb{E}(X^T d) = \frac{1}{n} \sum_{i=1}^{n} x^{(i)^T} d
$$

$$
= \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} x_j^{(i)} d_j \quad \text{by applying the definition of inner product.}
$$
Maximizing variance along any direction $d$

- Since $X$ has zero mean (for each pixel dimension $j$), then $X^T d$ also has zero mean (for any $d$):

$$\mathbb{E}(X^T d) = \frac{1}{n} \sum_{i=1}^{n} x^{(i)^T} d$$

$$= \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} x^{(i)} d_j$$

$$= \frac{1}{n} \sum_{j=1}^{m} \sum_{i=1}^{n} x^{(i)} d_j$$

since we can swap the order of the summations.
Maximizing variance along any direction \( \mathbf{d} \)

- Since \( \mathbf{X} \) has zero mean (for each pixel dimension \( j \)), then \( \mathbf{X}^T \mathbf{d} \) also has zero mean (for any \( \mathbf{d} \)):

\[
\mathbb{E}\left( \mathbf{X}^T \mathbf{d} \right) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}^{(i)}^T \mathbf{d} \\
= \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} x_{j}^{(i)} d_{j} \\
= \frac{1}{n} \sum_{j=1}^{m} \sum_{i=1}^{n} x_{j}^{(i)} d_{j} \\
= \frac{1}{n} \sum_{j=1}^{m} d_{j} \sum_{i=1}^{n} x_{j}^{(i)}
\]
Maximizing variance along any direction $d$

- Since $X$ has zero mean (for each pixel dimension $j$), then $X^T d$ also has zero mean (for any $d$):

$$
\mathbb{E} \left( X^T d \right) = \frac{1}{n} \sum_{i=1}^{n} x^{(i)}^T d
$$

$$
= \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} x^{(i)}_j d_j
$$

$$
= \frac{1}{n} \sum_{j=1}^{m} \sum_{i=1}^{n} x^{(i)}_j d_j
$$

$$
= \frac{1}{n} \sum_{j=1}^{m} d_j \sum_{i=1}^{n} x^{(i)}_j
$$

$$
= \frac{1}{n} \sum_{j=1}^{m} d_j \cdot 0
$$

since $X$ has zero mean
Maximizing variance along any direction $d$

- Since $X$ has zero mean (for each pixel dimension $j$), then $X^T d$ also has zero mean (for any $d$):

$$
\mathbb{E}(X^T d) = \frac{1}{n} \sum_{i=1}^{n} x^{(i)}^T d
$$

$$
= \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} x^{(i)}_j d_j
$$

$$
= \frac{1}{n} \sum_{j=1}^{m} \sum_{i=1}^{n} x^{(i)}_j d_j
$$

$$
= \frac{1}{n} \sum_{j=1}^{m} d_j \sum_{i=1}^{n} x^{(i)}_j
$$

$$
= \frac{1}{n} \sum_{j=1}^{m} d_j \sum_{i=1}^{n} x^{(i)}
$$

$$
= \frac{1}{n} \sum_{j=1}^{m} d_j 0
$$

$$
= 0
$$
Maximizing variance along any direction \( \mathbf{d} \)

- Therefore, the variance of \( \mathbf{X}^\top \mathbf{d} \) (for any \( \mathbf{d} \)) is just:

\[
\frac{1}{n} (\mathbf{X}^\top \mathbf{d})^\top (\mathbf{X}^\top \mathbf{d})
\]

- We thus want to find the \( \mathbf{d} \) that maximizes:

\[
(\mathbf{X}^\top \mathbf{d})^\top (\mathbf{X}^\top \mathbf{d})
\]

subject to the constraint that \( \mathbf{d} \) is a unit vector, i.e.:

\[
\mathbf{d}^\top \mathbf{d} = 1
\]
Maximizing variance along any direction $d$

- Since this is a constrained optimization problem, we can set up a Lagrangian function $L(d, \alpha)$:

$$
L(d, \alpha) = (X^\top d)^\top (X^\top d) - \alpha(d^\top d - 1)
$$

Objective Constraint
Maximizing variance along any direction $d$

- Since this is a constrained optimization problem, we can set up a Lagrangian function $L(d, \alpha)$:

$$L(d, \alpha) = (X^\top d)^\top (X^\top d) - \alpha(d^\top d - 1)$$

$$= d^\top XX^\top d - \alpha(d^\top d - 1)$$
Maximizing variance along any direction $d$

- Since this is a constrained optimization problem, we can set up a Lagrangian function $L(d, \alpha)$:

$$L(d, \alpha) = (X^\top d)^\top (X^\top d) - \alpha(d^\top d - 1)$$

$$= d^\top XX^\top d - \alpha(d^\top d - 1)$$

$$\frac{\partial L}{\partial d} = 2XX^\top d - 2\alpha d = 0$$
Maximizing variance along any direction $d$

- Since this is a constrained optimization problem, we can set up a Lagrangian function $L(d, \alpha)$:

$$L(d, \alpha) = (X^\top d)^\top (X^\top d) - \alpha(d^\top d - 1)$$

$$= d^\top XX^\top d - \alpha(d^\top d - 1)$$

$$\frac{\partial L}{\partial d} = 2XX^\top d - 2\alpha d = 0$$

$$\implies XX^\top d = \alpha d$$
Maximizing variance along any direction $d$

- Since this is a constrained optimization problem, we can set up a Lagrangian function $L(d, \alpha)$:

$$L(d, \alpha) = (X^\top d)^\top (X^\top d) - \alpha(d^\top d - 1)$$

$$= d^\top XX^\top d - \alpha(d^\top d - 1)$$

$$\frac{\partial L}{\partial d} = 2XX^\top d - 2\alpha d = 0$$

$$\implies XX^\top d = \alpha d$$

- In other words, $d$ is an eigenvector of $XX^\top$.

- Since we want to maximize the variance of the projections, we want the eigenvector with largest associated eigenvalue.
An eigenvector \( \mathbf{v} \) of a (square) matrix \( \mathbf{A} \) satisfies:

\[
\mathbf{A} \mathbf{v} = \alpha \mathbf{v}
\]

for some scalar eigenvalue \( \alpha \).

For an \( n \times n \) matrix \( \mathbf{A} \), there are \( n \) eigenvectors \( \mathbf{v} \) and associated eigenvalues \( \alpha \).

Eigenvectors/eigenvalues can be computed (in \( O(n^3) \) time) using many standard linear algebra libraries (e.g., \texttt{numpy}).
Positive semi-definite matrices

• Recall that the eigenvalues of every PSD matrix $A$ are always non-negative.

• Since $XX^T$ is PSD (as shown previously in class), its eigenvalues are non-negative.
PCA

• The direction \( \mathbf{d} \) along which the dataset \( \mathbf{X} \) varies the most is the **principal eigenvector** of \( \mathbf{XX}^T \), i.e., the eigenvector with largest associated eigenvalue.

• It can be shown that the second-most highly varying direction of \( \mathbf{X} \) is the eigenvector with second-largest associated value, etc.
PCA

• Algorithm:

1. From design matrix $X$, compute the mean vector $\bar{x}$:

$$
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x^{(i)}
$$

3. Subtract $\bar{x}$ from each example $x^{(i)}$, and then form matrix $\tilde{X}$ (same size as $X$), which should have a mean (over all $n$ examples) of 0 along each dimension $j$.

$$
\tilde{X} = \begin{bmatrix}
(x^{(1)} - \bar{x}) & \ldots & (x^{(n)} - \bar{x})
\end{bmatrix}
$$
PCA

• Algorithm:

1. From design matrix $X$, compute the mean vector $\bar{x}$:

   $$ \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x^{(i)} $$

3. Subtract $\bar{x}$ from each example $x^{(i)}$, and then form matrix $\tilde{X}$ (same size as $X$), which should have a mean (over all $n$ examples) of 0 along each dimension $j$.

4. Compute the eigenvectors & eigenvalues of $\tilde{X}\tilde{X}^T$.

5. The $k^{th}$ principal component (PC) of $X$ is the eigenvector $v$ of $\tilde{X}\tilde{X}^T$ with the $k^{th}$-largest eigenvalue.
PCA on MNIST

For all classes
PCA on MNIST

For each class with its own color
Unsupervised learning

• PCA is an example of an unsupervised machine learning algorithm.

• **Unsupervised** — we never looked at the training labels!
  - In some settings, the data might not even be labeled.
Unsupervised learning

- PCA is an example of an unsupervised machine learning algorithm.

- **Unsupervised** — we never looked at the training labels!
  - In some settings, the data might not even be labeled.

- Note that there are other visualization methods (e.g., Linear Discriminant Analysis (LDA)) that are supervised:
  - Project data onto directions that best linearly separate the data classes.