CS 453X: Class 16

Jacob Whitehill

Data visualization

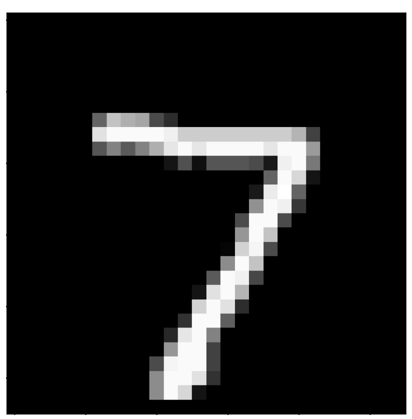
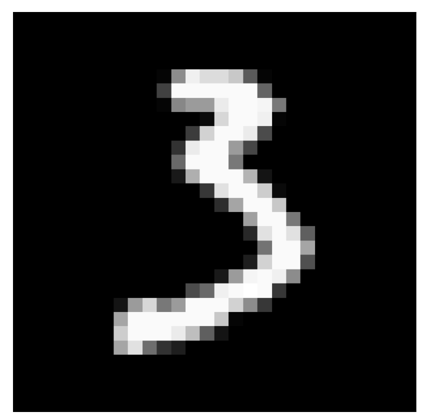
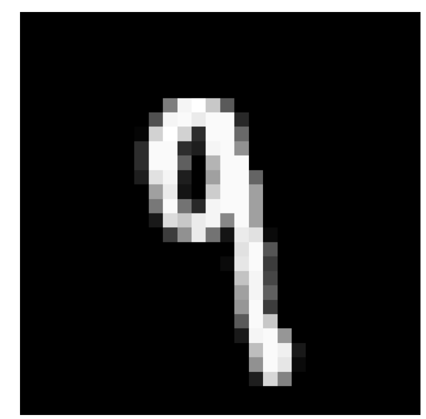
Data visualization

- 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.:

- 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, …, n):

Data visualization

- Let’s give a naive try…
- For each image $\mathbf{x}^{(i)}$ (where i=1, …, n):
  - Retrieve the values of just the first two pixels, i.e., ($\mathbf{x}^{(i)}{}_1$, $\mathbf{x}_2{}^{(i)}$).

(0,0)	**(1,0)**	(2,0)	…	(27,0)
(0,1)	…			
(0,2)				
…				
(0,27)				(27,27)

Data visualization

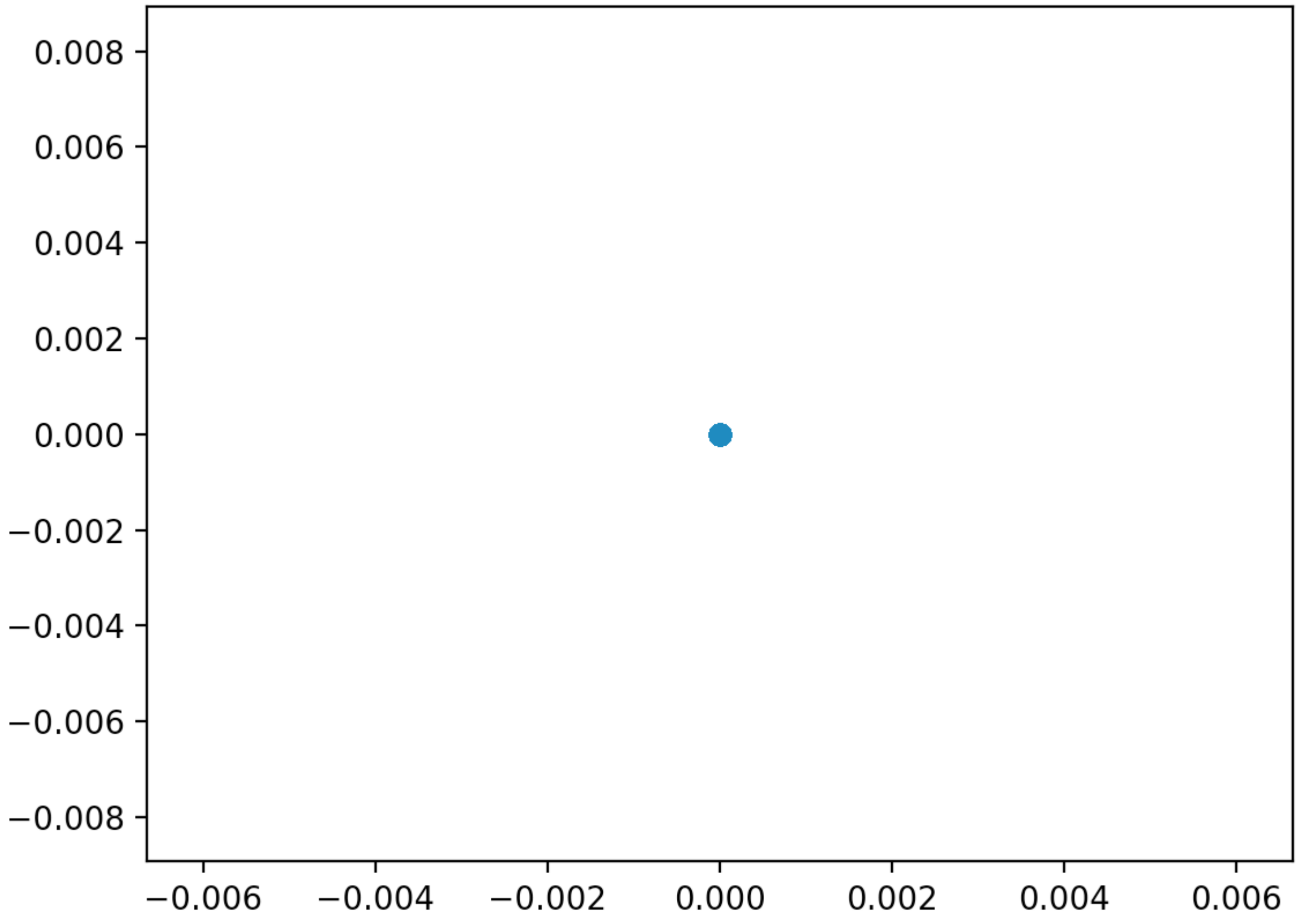
- Let's give a naive try…
- For each image $\mathbf{x}^{(i)}$ (where i=1, …, n):
    - Retrieve the values of just the first two pixels, i.e., $(\mathbf{x}^{(i)}{}_1, \mathbf{x}_2{}^{(i)})$.
    - Plot the point $(\mathbf{x}^{(i)}{}_1, \mathbf{x}_2{}^{(i)})$ in 2-D space.

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?

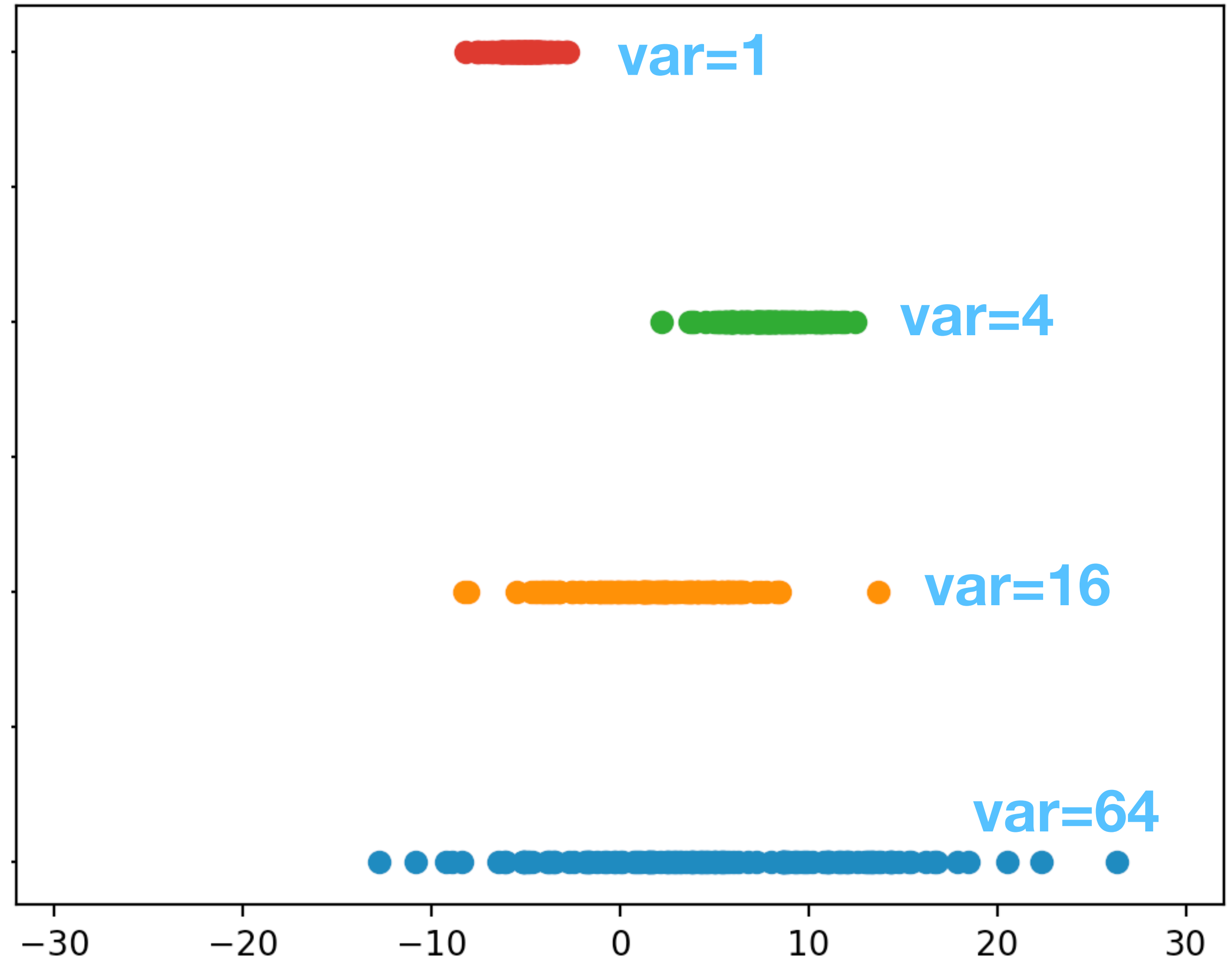
MNIST 2-D visualization: first two pixel values

- 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) **variance** across each of these two dimensions.

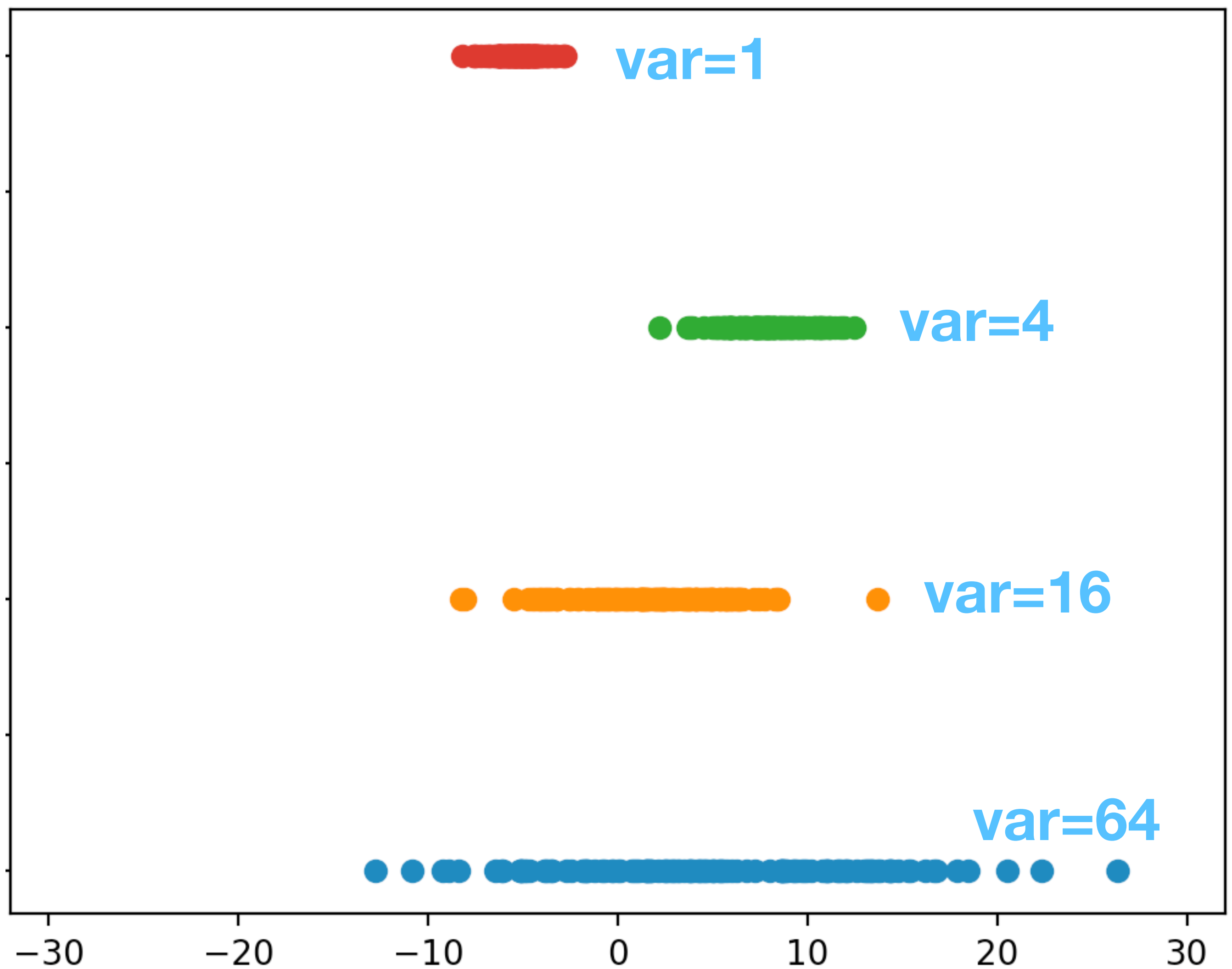
Variance

- 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*:

$$\mathbb{E}(\mathbf{p}) \quad = \quad \frac{1}{n}\sum_{i=1}^{n}\mathbf{p}_i$$

Variance

- From basic statistics:
  - **Mean** of an *n*-dimensional vector: sum and divide by *n*:

$$\mathbb{E}(\mathbf{p}) \quad = \quad \frac{1}{n}\sum_{i=1}^{n}\mathbf{p}_i$$

  - **Variance** of an *n*-dimensional vector: the mean squared distance from the mean:

$$\mathbb{V}(\mathbf{p}) \quad = \quad \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{p}_i - \mathbb{E}(\mathbf{p})\right)^2$$

Variance

- Alternatively, we can compute the variance in two steps:
  - First subtract the mean from each element of **p**:

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

Variance

- Alternatively, we can compute the variance in two steps:
  - First subtract the mean from each element of **p**:

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

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

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

Variance

- Alternatively, we can compute the variance in two steps:
  - First subtract the mean from each element of **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:

$$\begin{aligned} \mathbb{V}(\tilde{\mathbf{p}}) &= \frac{1}{n} \sum_{i=1}^{n} \left(\tilde{\mathbf{p}}_i - \mathbb{E}(\tilde{\mathbf{p}})\right)^2 \\ &= \frac{1}{n} \sum_{i=1}^{n} \left(\tilde{\mathbf{p}}_i - 0\right)^2 \end{aligned}$$

Variance

- Alternatively, we can compute the variance in two steps:
  - First subtract the mean from each element of **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:

$$\begin{aligned} \mathbb{V}(\tilde{\mathbf{p}}) &= \frac{1}{n}\sum_{i=1}^{n}\left(\tilde{\mathbf{p}}_i - \mathbb{E}(\tilde{\mathbf{p}})\right)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}\left(\tilde{\mathbf{p}}_i - 0\right)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}\left(\tilde{\mathbf{p}}_i\right)^2 \end{aligned}$$

Variance

- Alternatively, we can compute the variance in two steps:
  - First subtract the mean from each element of **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:

$$\begin{aligned}
\mathbb{V}(\tilde{\mathbf{p}}) &= \frac{1}{n}\sum_{i=1}^{n}\left(\tilde{\mathbf{p}}_i - \mathbb{E}(\tilde{\mathbf{p}})\right)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(\tilde{\mathbf{p}}_i - 0\right)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(\tilde{\mathbf{p}}_i\right)^2 \\
&= \frac{1}{n}\tilde{\mathbf{p}}^\top\tilde{\mathbf{p}}
\end{aligned}$$

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 **x** is equivalent to **projecting** each **x** onto a **unit vector**.

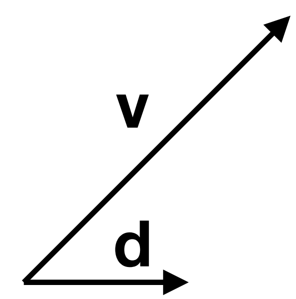
$$\mathbf{x}^\top \mathbf{d}$$

where **d** is a vector of *n*-1 zeros and 1 one (whose location corresponds to the particular pixel location):

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

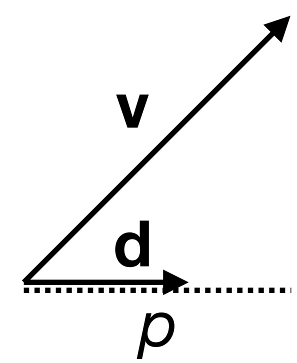
Projection

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



Projection

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



- The scalar projection p measures the distance of **v** along **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:

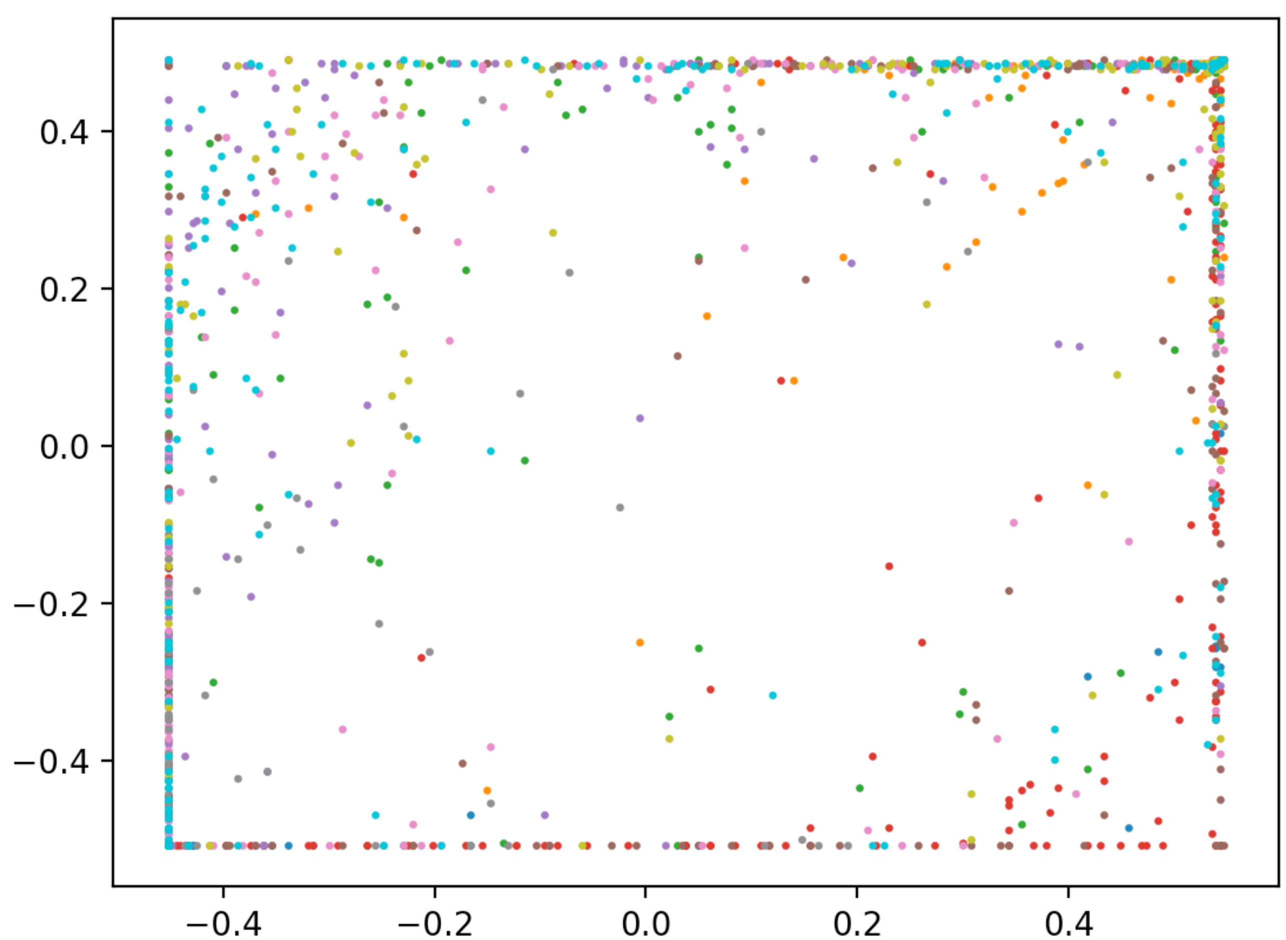
$$\mathbf{p} = \mathbf{X}^\top \mathbf{d}$$

- We can then calculate the variance of the scalar projections by calculating Var(**p**).

- By searching over all 784 possible dimensions, we can find the two dimensions along which variance is maximized.

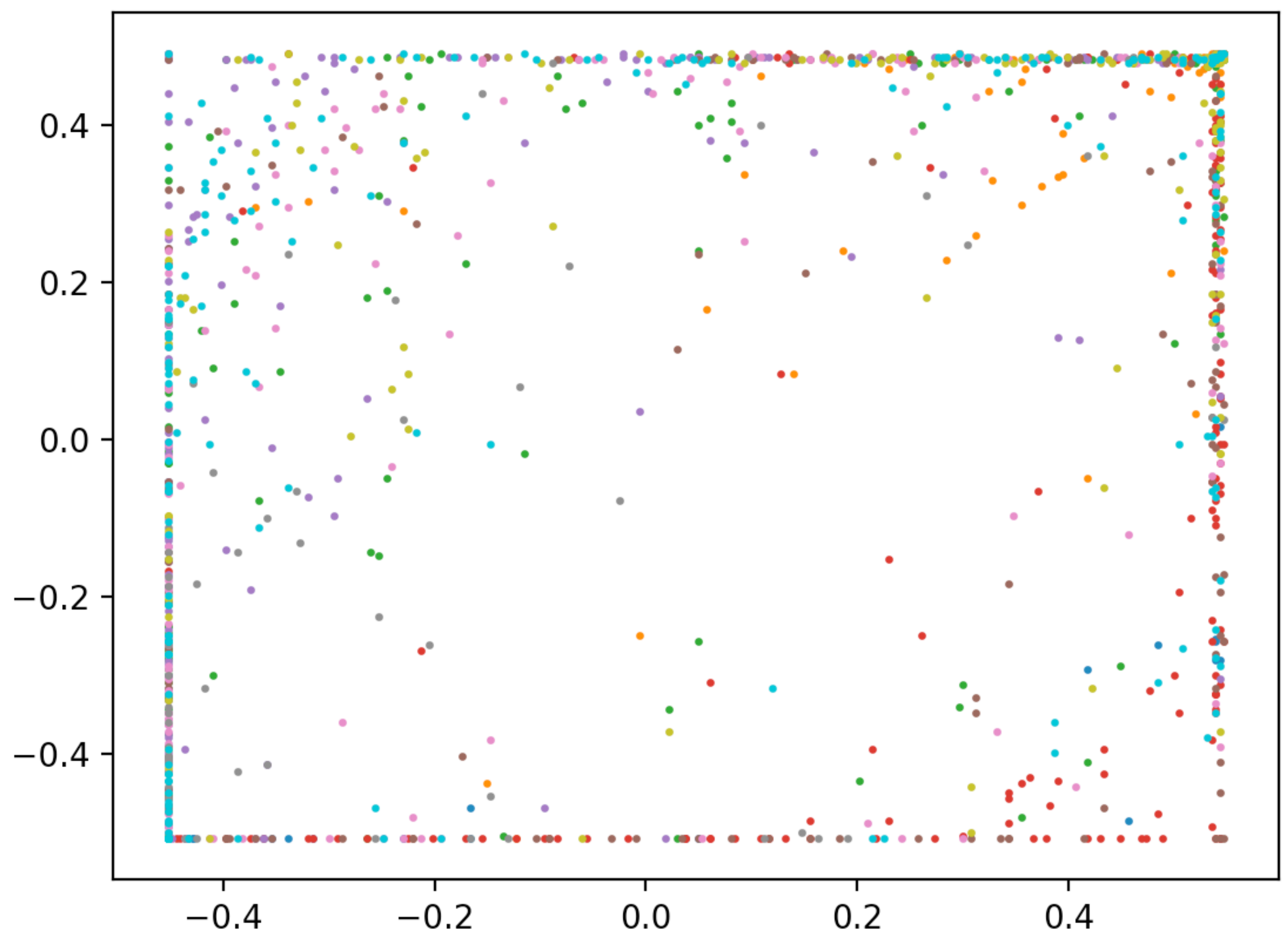
MNIST 2-D visualization: two most highly-varying dimensions

- 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)



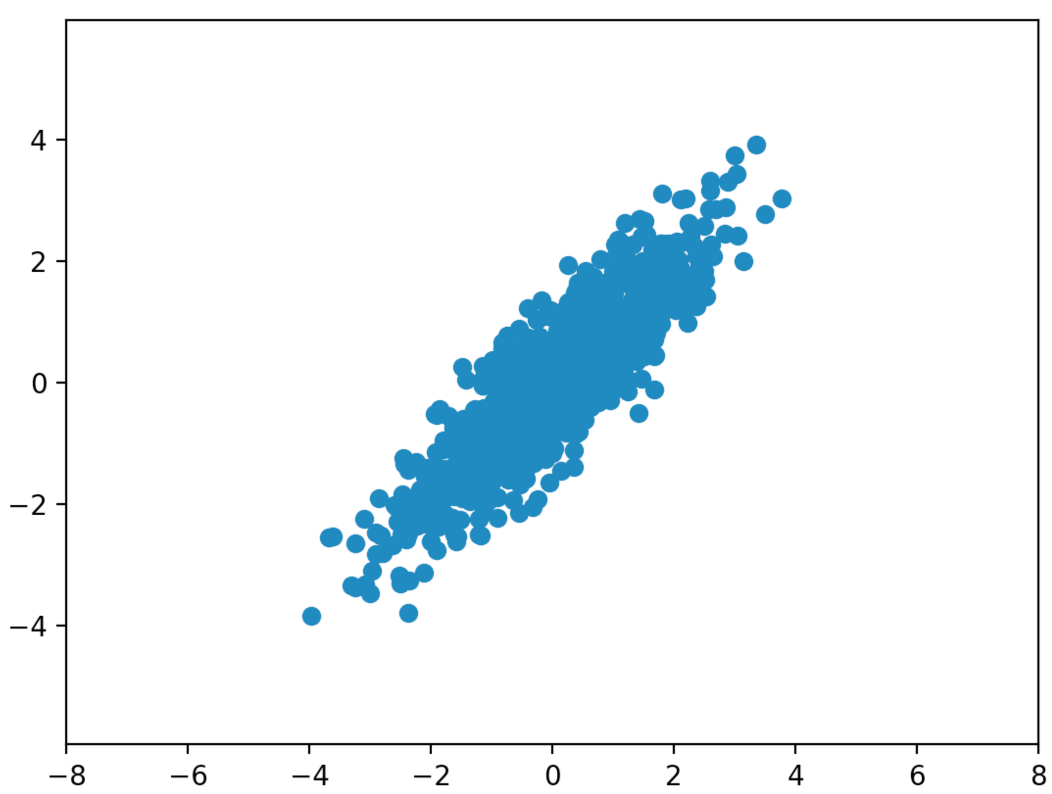
MNIST 2-D visualization: two most highly-varying dimensions

- 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?

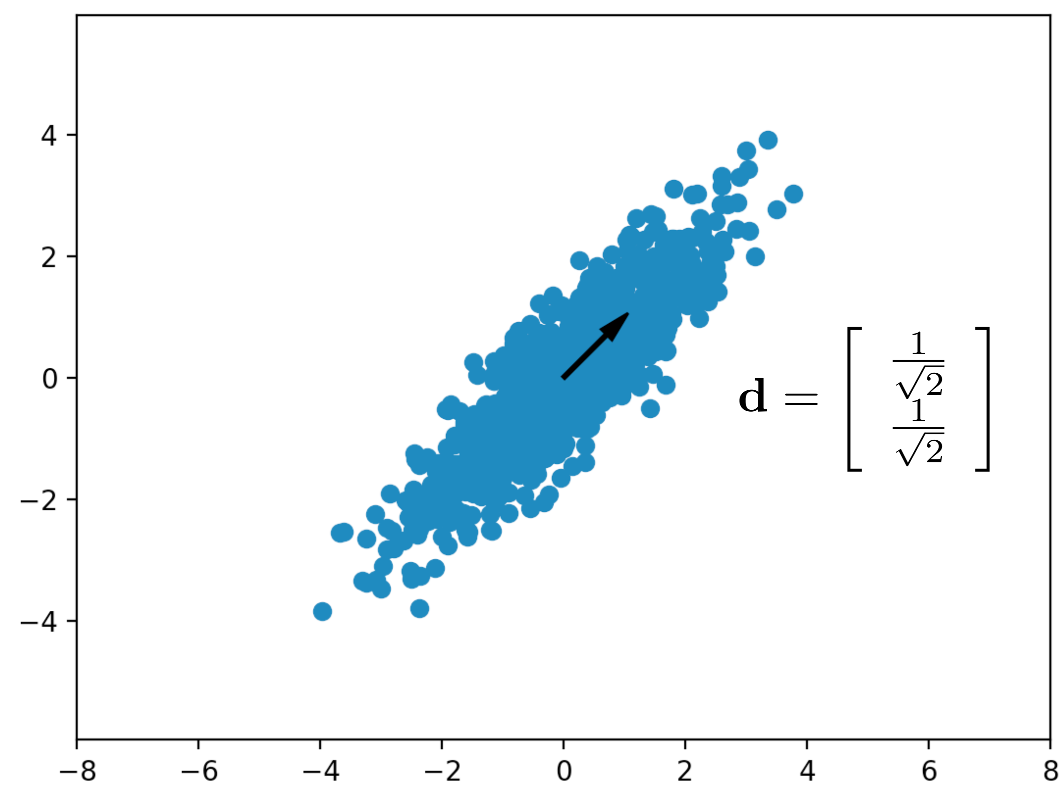


Beyond axis-aligned directions

- 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?



Maximizing variance along any direction d

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

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

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_j^{(i)} = 0$$

- (If this is not the case, then just subtract off the mean vector from each example $\mathbf{x}^{(i)}$.)

Maximizing variance along any direction d

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

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

by applying the definition of mean.

Maximizing variance along any direction d

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

$$
\begin{aligned}
\mathbb{E}\left(\mathbf{X}^\top \mathbf{d}\right) &= \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}^{(i)\top}\mathbf{d} \\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{x}_j^{(i)}\mathbf{d}_j
\end{aligned}
$$

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**$^\top$**d** also has zero mean (for any **d**):

$$
\begin{aligned}
\mathbb{E}\left(\mathbf{X}^\top \mathbf{d}\right) &= \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}^{(i)\top}\mathbf{d} \\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{x}_j^{(i)}\mathbf{d}_j \\
&= \frac{1}{n}\sum_{j=1}^{m}\sum_{i=1}^{n} \mathbf{x}_j^{(i)}\mathbf{d}_j
\end{aligned}
$$

since we can swap the order of the summations.

Maximizing variance along any direction d

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

$$
\begin{aligned}
\mathbb{E}\left(\mathbf{X}^\top \mathbf{d}\right) &= \frac{1}{n}\sum_{i=1}^{n} {\mathbf{x}^{(i)}}^\top \mathbf{d} \\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{x}_j^{(i)} \mathbf{d}_j \\
&= \frac{1}{n}\sum_{j=1}^{m}\sum_{i=1}^{n} \mathbf{x}_j^{(i)} \mathbf{d}_j \\
&= \frac{1}{n}\sum_{j=1}^{m} \mathbf{d}_j \sum_{i=1}^{n} \mathbf{x}_j^{(i)}
\end{aligned}
$$

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**):

$$
\begin{aligned}
\mathbb{E}\left(\mathbf{X}^\top \mathbf{d}\right) &= \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}^{(i)\top}\mathbf{d} \\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{x}_j^{(i)}\mathbf{d}_j \\
&= \frac{1}{n}\sum_{j=1}^{m}\sum_{i=1}^{n} \mathbf{x}_j^{(i)}\mathbf{d}_j \\
&= \frac{1}{n}\sum_{j=1}^{m} \mathbf{d}_j \boxed{\sum_{i=1}^{n} \mathbf{x}_j^{(i)}} \\
&= \frac{1}{n}\sum_{j=1}^{m} \mathbf{d}_j \boxed{0}
\end{aligned}
$$

since X has zero mean

Maximizing variance along any direction d

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

$$
\begin{aligned}
\mathbb{E}\left(\mathbf{X}^\top\mathbf{d}\right) &= \frac{1}{n}\sum_{i=1}^{n}{\mathbf{x}^{(i)}}^\top\mathbf{d} \\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbf{x}_j^{(i)}\mathbf{d}_j \\
&= \frac{1}{n}\sum_{j=1}^{m}\sum_{i=1}^{n}\mathbf{x}_j^{(i)}\mathbf{d}_j \\
&= \frac{1}{n}\sum_{j=1}^{m}\mathbf{d}_j\sum_{i=1}^{n}\mathbf{x}_j^{(i)} \\
&= \frac{1}{n}\sum_{j=1}^{m}\mathbf{d}_j 0 \\
&= \mathbf{0}
\end{aligned}
$$

Maximizing variance along any direction d

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

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

- We thus want to find the **d** that maximizes:

$$(\mathbf{X}^\top\mathbf{d})^\top(\mathbf{X}^\top\mathbf{d})$$

subject to the constraint that **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 $L(\mathbf{d}, \alpha)$:

$$L(\mathbf{d}, \alpha) = \underbrace{(\mathbf{X}^\top \mathbf{d})^\top (\mathbf{X}^\top \mathbf{d})}_{\text{Objective}} - \alpha \underbrace{(\mathbf{d}^\top \mathbf{d} - 1)}_{\text{Constraint}}$$

Maximizing variance along any direction d

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

$$
\begin{aligned}
L(\mathbf{d}, \alpha) &= (\mathbf{X}^\top \mathbf{d})^\top (\mathbf{X}^\top \mathbf{d}) - \alpha(\mathbf{d}^\top \mathbf{d} - 1) \\
&= \mathbf{d}^\top \mathbf{X}\mathbf{X}^\top \mathbf{d} - \alpha(\mathbf{d}^\top \mathbf{d} - 1)
\end{aligned}
$$

Maximizing variance along any direction d

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

$$
\begin{aligned}
L(\mathbf{d}, \alpha) &= (\mathbf{X}^\top \mathbf{d})^\top (\mathbf{X}^\top \mathbf{d}) - \alpha(\mathbf{d}^\top \mathbf{d} - 1) \\
&= \mathbf{d}^\top \mathbf{X}\mathbf{X}^\top \mathbf{d} - \alpha(\mathbf{d}^\top \mathbf{d} - 1) \\
\frac{\partial L}{\partial \mathbf{d}} &= 2\mathbf{X}\mathbf{X}^\top \mathbf{d} - 2\alpha \mathbf{d} = 0
\end{aligned}
$$

Maximizing variance along any direction d

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

$$
\begin{aligned}
L(\mathbf{d}, \alpha) &= (\mathbf{X}^\top \mathbf{d})^\top (\mathbf{X}^\top \mathbf{d}) - \alpha(\mathbf{d}^\top \mathbf{d} - 1) \\
&= \mathbf{d}^\top \mathbf{X}\mathbf{X}^\top \mathbf{d} - \alpha(\mathbf{d}^\top \mathbf{d} - 1) \\
\frac{\partial L}{\partial \mathbf{d}} &= 2\mathbf{X}\mathbf{X}^\top \mathbf{d} - 2\alpha \mathbf{d} = 0 \\
\implies \mathbf{X}\mathbf{X}^\top \mathbf{d} &= \alpha \mathbf{d}
\end{aligned}
$$

Maximizing variance along any direction d

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

$$
\begin{aligned}
L(\mathbf{d}, \alpha) &= (\mathbf{X}^\top \mathbf{d})^\top (\mathbf{X}^\top \mathbf{d}) - \alpha(\mathbf{d}^\top \mathbf{d} - 1) \\
&= \mathbf{d}^\top \mathbf{X}\mathbf{X}^\top \mathbf{d} - \alpha(\mathbf{d}^\top \mathbf{d} - 1) \\
\frac{\partial L}{\partial \mathbf{d}} &= 2\mathbf{X}\mathbf{X}^\top \mathbf{d} - 2\alpha \mathbf{d} = 0 \\
\implies \mathbf{X}\mathbf{X}^\top \mathbf{d} &= \alpha \mathbf{d}
\end{aligned}
$$

- In other words, $\mathbf{d}$ is an eigenvector of $\mathbf{X}\mathbf{X}^\top$.

- Since we want to maximize the variance of the projections, we want the eigenvector with largest associated eigenvalue.

Eigenvector

- An **eigenvector v** of a (square) matrix **A** satisfies:

  $$\mathbf{A}\mathbf{v} = \alpha\mathbf{v}$$

  for some scalar **eigenvalue** α.

- For an n x n matrix **A**, there are n eigenvectors **v** and associated eigenvalues α.

- Eigenvectors/eigenvalues can be computed (in $O(n^3)$ time) using many standard linear algebra libraries (e.g., **`numpy`**).

Positive semi-definite matrices

- Recall that the eigenvalues of every PSD matrix $\mathbf{A}$ are always non-negative.
- Since $\mathbf{XX}^T$ is PSD (as shown previously in class), its eigenvalues are non-negative.

PCA

- The direction **d** along which the dataset **X** varies the most is the **principal eigenvector** of $\mathbf{X}\mathbf{X}^T$, i.e., the eigenvector with largest associated eigenvalue.
- It can be shown that the *second*-most highly varying direction of **X** is the eigenvector with second-largest associated value, etc.

PCA

- Algorithm:

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

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

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

$$\widetilde{\mathbf{X}} = \begin{bmatrix} | & & | \\ (\mathbf{x}^{(1)} - \bar{\mathbf{x}}) & \dots & (\mathbf{x}^{(n)} - \bar{\mathbf{x}}) \\ | & & | \end{bmatrix}$$

PCA

- Algorithm:

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

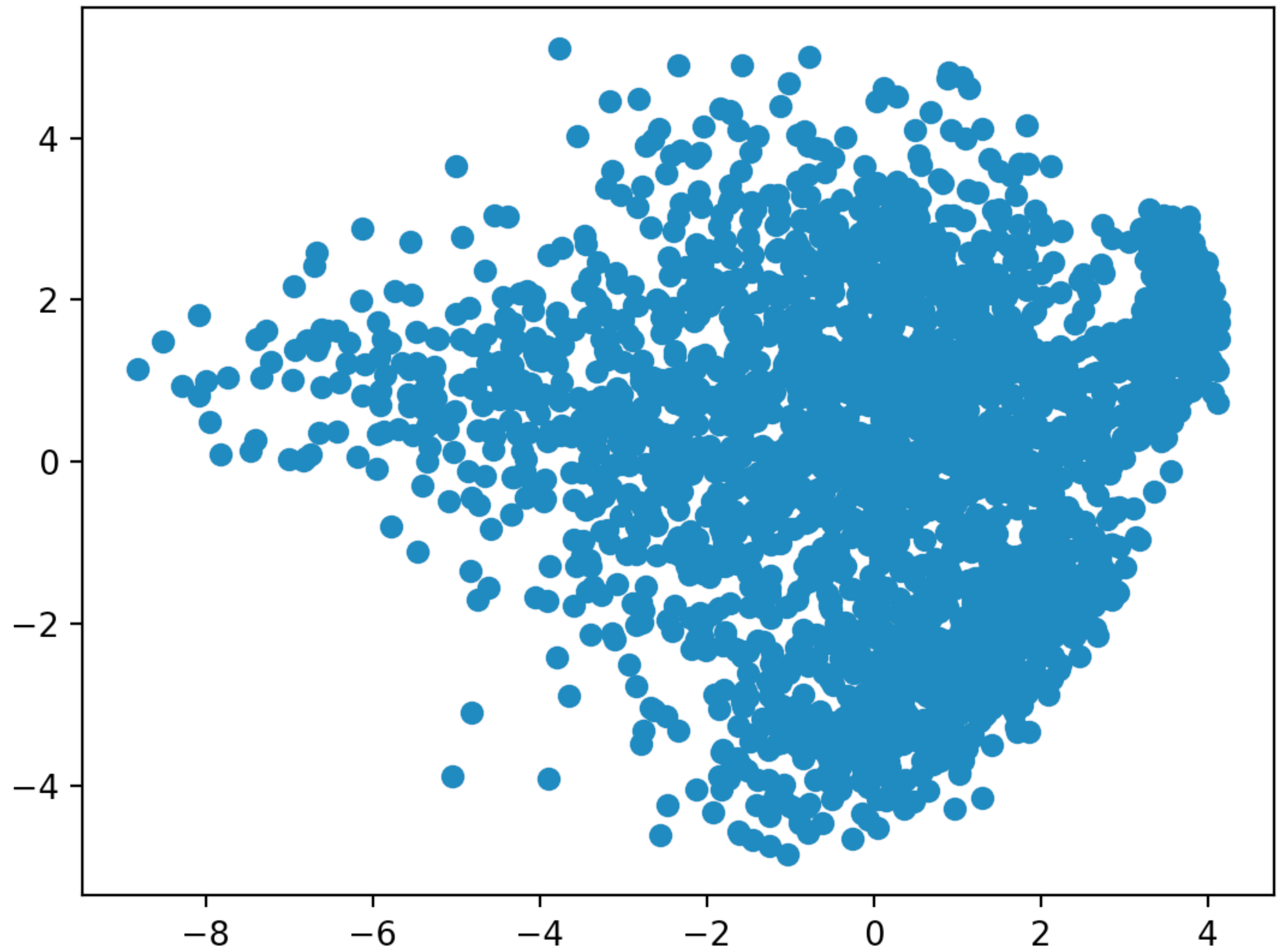
$$\overline{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}^{(i)}$$

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

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

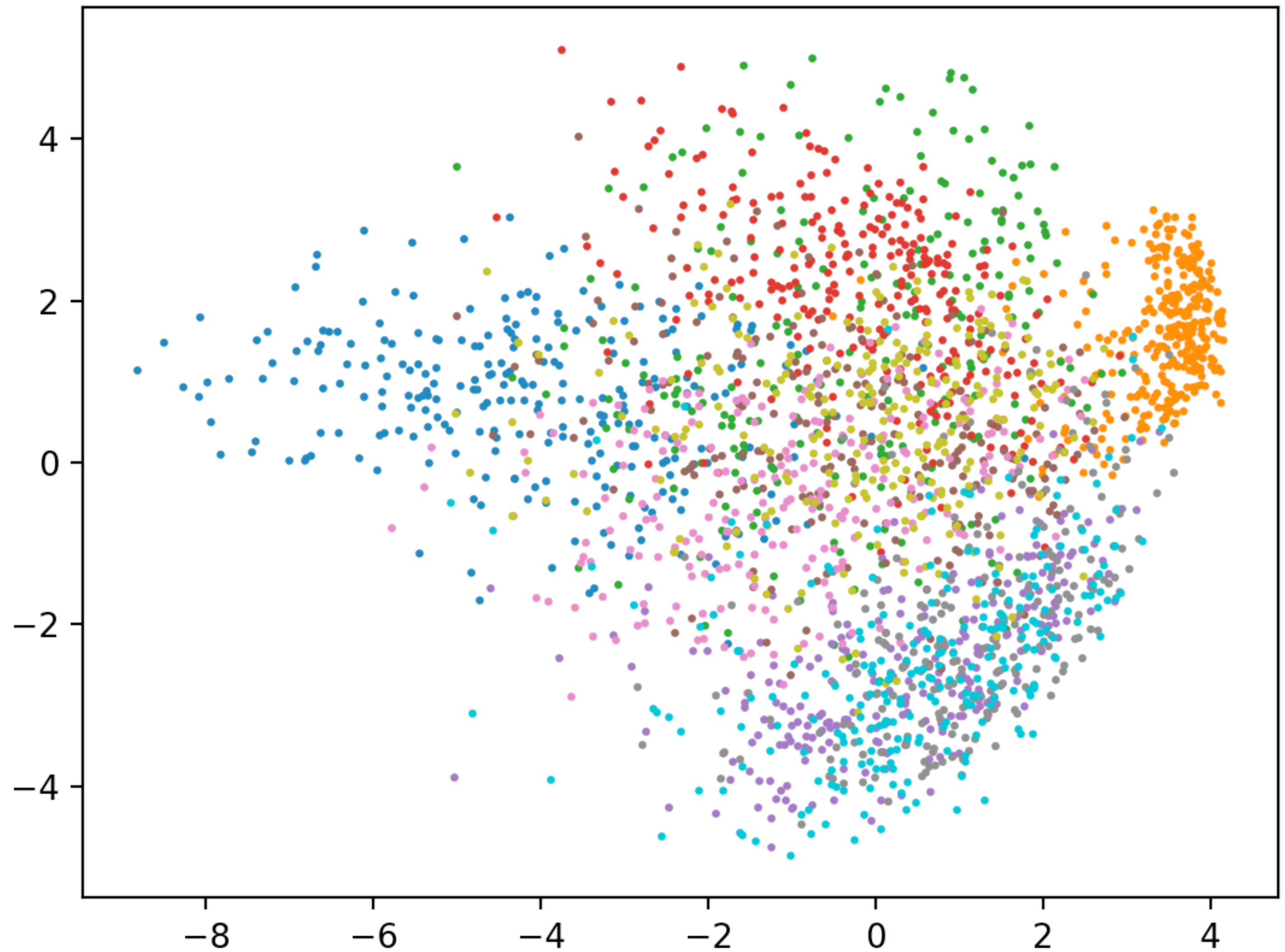
5. The k^{th} **principal component** (**PC**) of $\mathbf{X}$ is the eigenvector $\mathbf{v}$ of $\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\mathsf{T}$ with the k^{th}-largest eigenvalue.

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.