Dimensionality Reduction

- Some of the most interesting applications of machine learning involve data that lives in a high-dimensional space.
  - * Ex: An image that is 1000 pixels wide and 800 pixels tall lives in 800000-dimensional space.
- Moreover, the number of examples n in our dataset may be significantly lower than the dimension d.
  - → As a result, classifiers can end up overfitting the data.
- Ideally, we would like to find a lower-dimensional representation of the dataset that preserves the important relationships between examples.
  - → This is an unsupervised learning problem.
- We will focus on Principal Component Analysis, which is one of the simplest and most popular methods.

- It will be useful to recall a few important properties of the covariance matrix $\Sigma_{\underline{X}} = \mathbb{E}[(\underline{X} - \mathbb{E}[\underline{X}])(\underline{X} - \mathbb{E}[\underline{X}])^T]$.
  - → All of the eigenvalues are real and non-negative. Using this fact, we can sort them (along with the corresponding eigenvectors) into descending order $\lambda_1 \geq \cdots \geq \lambda_d \geq 0$.
  - → All of the eigenvectors are orthonormal to each other. Specifically, let $\underline{v}_1, \ldots, \underline{v}_d$ be the eigenvectors (sorted as above), then $\underline{v}_i^T \underline{v}_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$.
  - → Collecting the eigenvectors into a matrix $V = [\underline{v}_1 \cdots \underline{v}_d]$ and the eigenvalues into a diagonal matrix $\Lambda = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_d \end{bmatrix}$ we can write the eigendecomposition

$$\Sigma_{\underline{X}} = V \Lambda V^T \quad \text{where} \quad V^T V = V V^T = I$$

Orthogonal Matrix

  - → These properties hold for sample covariance matrices too.

- Given a random vector $\underline{X}$ with mean vector $\underline{\mu}_{\underline{X}} = \mathbb{E}[\underline{X}]$ and covariance matrix $\Sigma_{\underline{X}} = V\Lambda V^T$, we can carry out the following coordinate transformation:
  - → Center the distribution at the origin by subtracting the mean:

$$\underline{X}_{centered} = \underline{X} - \underline{\mu}_{\underline{X}}$$

  - → Rotate the coordinate system by multiplying by the orthogonal matrix of eigenvectors:

$$\underline{Z} = V^T \underline{X}_{centered} = V^T(\underline{X} - \underline{\mu}_{\underline{X}})$$

No redundancy between transformed features.

- The entries of $\underline{Z}$ are uncorrelated with each other, $\mathrm{Cov}[Z_i, Z_j] = 0 \quad i \neq j$ and their variances are equal to the eigenvalues $\mathrm{Var}[Z_i] = \lambda_i$. The covariance matrix is

$$\Sigma_{\underline{Z}} = V^T \Sigma_{\underline{X}} V = \underbrace{V^T V}_{I} \Lambda \underbrace{V^T V}_{I} = \Lambda$$

Covariance of Linear Transform

diagonal

- To reduce the dimension to k, we simply throw out all but the top k entries of $\underline{Z}$.

- Principal component analysis follows the same steps, except that we also need to estimate the mean vector and covariance matrix from the data $\underline{X}_1, \dots, \underline{X}_n$. ← Usually training data.

→ Collect the n d-dimensional examples into an $n \times d$ data matrix:

$$\mathbf{X} = \begin{bmatrix} \underline{X}_1^T \\ \vdots \\ \underline{X}_n^T \end{bmatrix}$$

all-ones vector

$$\underline{1}_n = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \Big\updownarrow n$$

→ Compute the sample mean vector: $\hat{\underline{\mu}}_{\underline{X}} = \frac{1}{n} \sum_{i=1}^{n} \underline{X}_i = \frac{1}{n} \mathbf{X}^T \underline{1}_n$

→ Center the data $\underline{X}_{\text{centered},i} = \underline{X}_i - \hat{\underline{\mu}}_{\underline{X}}$, $\mathbf{X}_{\text{centered}} = \mathbf{X} - \underline{1}_n \hat{\underline{\mu}}_{\underline{X}}^T$

→ Compute the sample covariance matrix:

$$\hat{\Sigma}_{\underline{X}} = \frac{1}{n-1} \sum_{i=1}^{n} (\underline{X}_i - \hat{\underline{\mu}}_{\underline{X}})(\underline{X}_i - \hat{\underline{\mu}}_{\underline{X}})^T = \frac{1}{n-1} \sum_{i=1}^{n} \underline{X}_{\text{centered},i} \underline{X}_{\text{centered},i}^T$$

$$= \frac{1}{n-1} (\mathbf{X} - \underline{1}_n \hat{\underline{\mu}}_{\underline{X}}^T)^T (\mathbf{X} - \underline{1}_n \hat{\underline{\mu}}_{\underline{X}}^T) = \frac{1}{n-1} \mathbf{X}_{\text{centered}}^T \mathbf{X}_{\text{centered}}$$

→ Compute the eigendecomposition $\hat{\Sigma}_{\underline{X}} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^T$.

- **Principal Component Analysis:**

→ Given a dataset $\underline{X}$, compute the sample mean vector $\hat{\underline{\mu}}_{\underline{x}}$ and sample covariance matrix $\hat{\Sigma}_{\underline{x}}$.

→ Compute the eigendecomposition

$$\hat{\Sigma}_{\underline{x}} = V \Lambda V^T$$

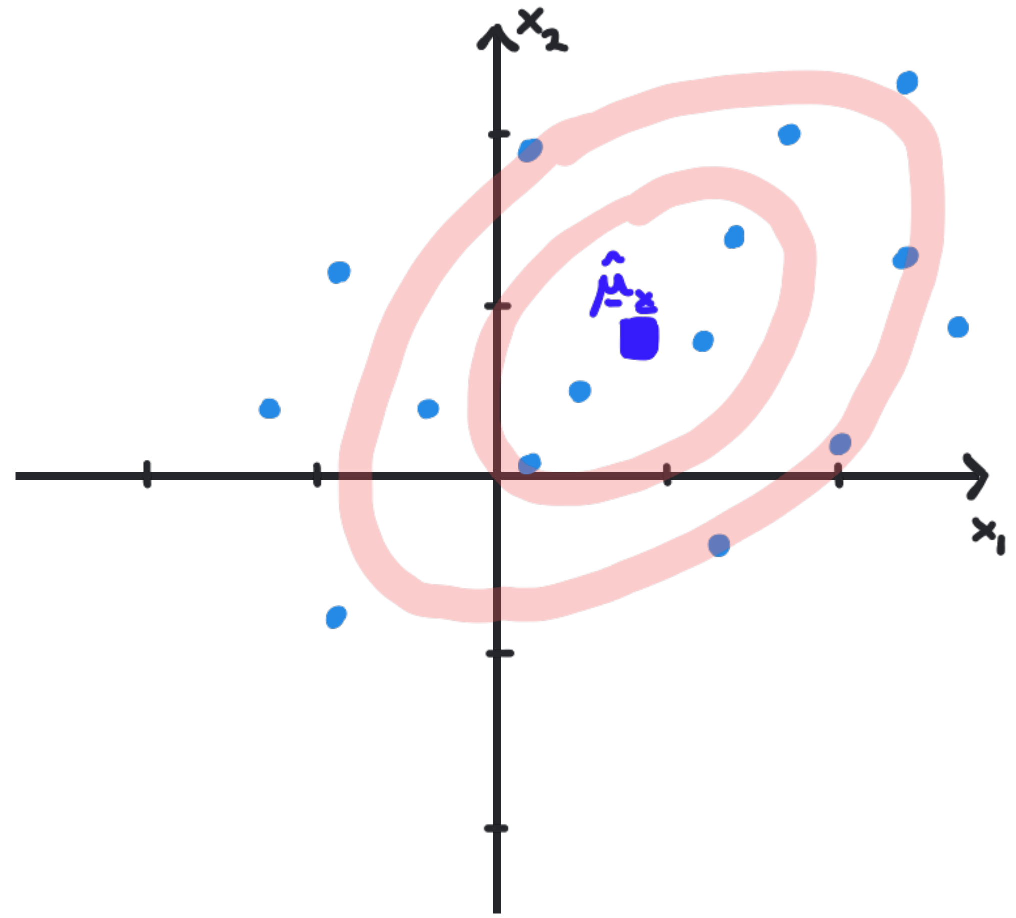
→ Keep only the first k eigenvectors

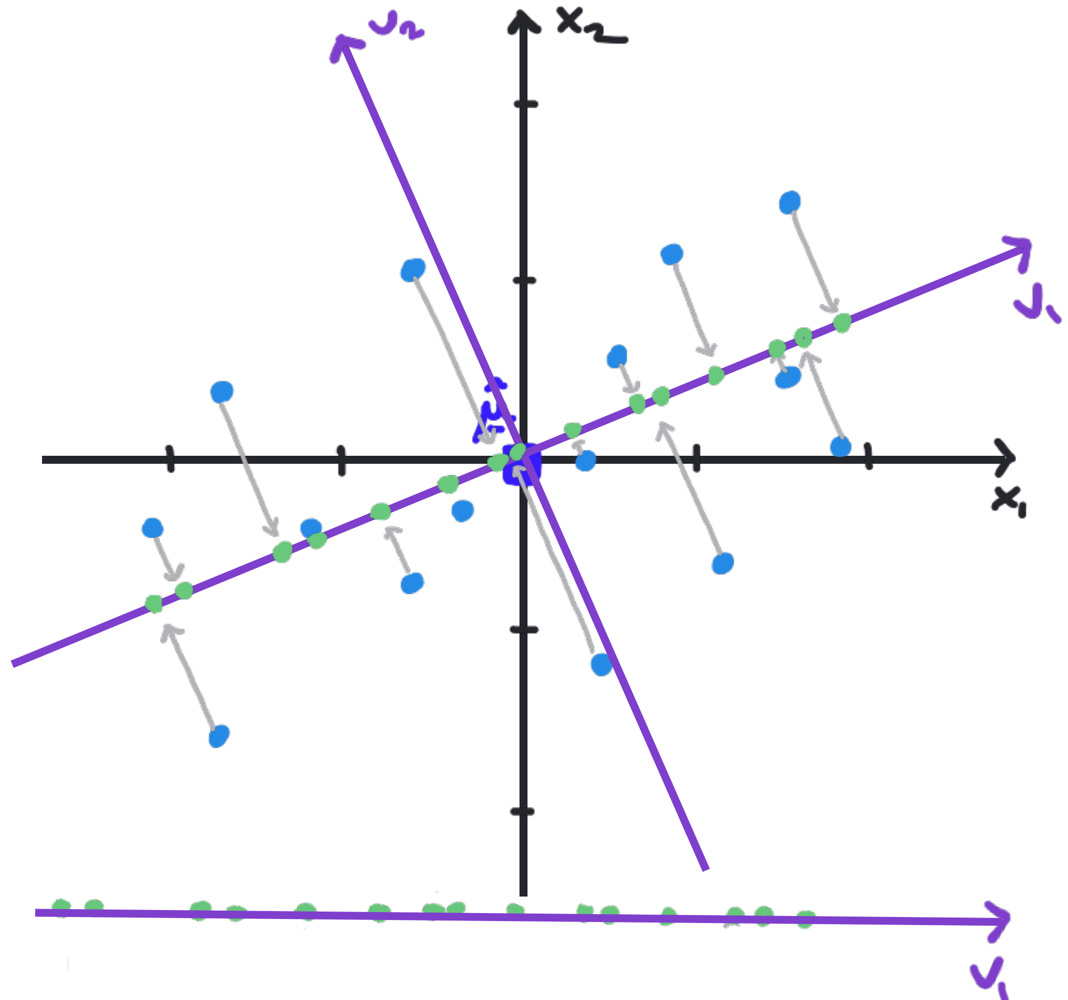
$$V_k = [\underline{v}_1 \cdots \underline{v}_k]$$

→ Center the data: $X_{centered} = X - \underline{1}_n \hat{\underline{\mu}}_{\underline{x}}^T$

→ Project onto V_k: $Z = X_{centered} V_k$

→ We can use Z as a k-dimensional representation of our original d-dimensional dataset X where $k < d$.





- One important application of PCA is visualizing high-dimensional datasets in 2 or 3 dimensions.
- Another application is as a pre-processing step for supervised learning to avoid overfitting:
  - → Estimate the sample mean vector and covariance matrix from the training data only:

$$\hat{\underline{\mu}}_{\underline{x}} = \frac{1}{n_{train}} \mathbf{X}_{train} \qquad \hat{\Sigma}_{\underline{x}} = \frac{1}{n_{train}-1}\left(\mathbf{X}_{train} - \underline{1}_{n_{train}} \hat{\underline{\mu}}_{\underline{x}}^T\right)^T \left(\mathbf{X}_{train} - \underline{1}_{n_{train}} \hat{\underline{\mu}}_{\underline{x}}^T\right)$$

  - → Compute the eigendecomposition $\hat{\Sigma}_{\underline{x}} = \mathbf{V} \Lambda \mathbf{V}^T$ and $\mathbf{V}_k = [\underline{v}_1 \cdots \underline{v}_k]$.
  - → Center and project both the training and test data:

$$\mathbf{X}_{train,reduced} = \left(\mathbf{X}_{train} - \underline{1}_{n_{train}} \hat{\underline{\mu}}_{\underline{x}}^T\right)\mathbf{V}_k \qquad \mathbf{X}_{test,\, reduced} = \left(\mathbf{X}_{test} - \underline{1}_{n_{test}} \hat{\underline{\mu}}_{\underline{x}}^T\right)\mathbf{V}_k$$

  - → Use the reduced training data $\mathbf{X}_{train,reduced}$ along with the original training labels $\underline{Y}_{train}$ to train the algorithm (such as a classifier). Test it using the reduced test data $\mathbf{X}_{test,\, reduced}$ along with the original test labels $\underline{Y}_{test}$.