Dimensionality Reduction

Part III

PCA – Pokémon: Part I

- Inspired from: https://www.kaggle.com/strakul5/d/abcsds/pokemon/principal-component-analysis-of-pokemon-data
- 800 Pokemons, 6 features for each (HP, Atk, Def, Sp Atk, Sp Def, Speed)
- How many principle components?

$$\frac{\lambda_i}{\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6}$$

	$\boxed{\lambda_1}$	$\boxed{\lambda_2}$	$\boxed{\lambda_3}$	$\boxed{\lambda_4}$	λ_5	λ_6
ratio	0.45	0.18	0.13	0.12	0.07	0.04

Using 4 components is good enough

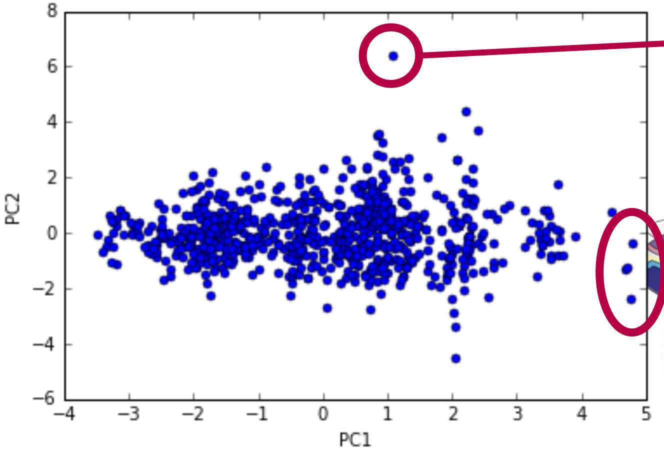
PCA – Pokémon: Part II

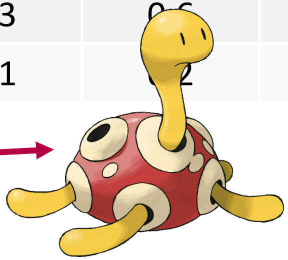
http://140.112.21.35:2880/~tlkagk/pokemon/pca.html

	HP	Atk	Def	Sp Atk	Sp Def	Speed
PC1	0.4	0.4	0.4	0.5	0.4	0.3
PC2	0.1	0.0	0.6	-0.3	0.2	-0.7
PC3	-0.5	-0.6	0.1	0.3	[illegible]	[illegible]
PC4	0.7	-0.4	-0.4	0.1	[illegible]	[illegible]

Power Level

Def sacrifice speed



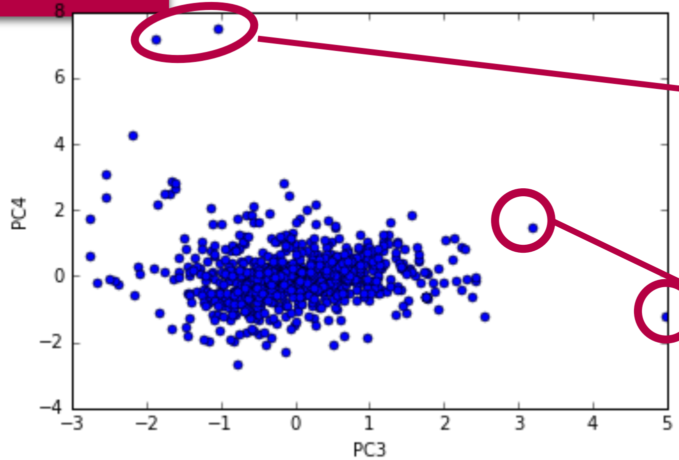


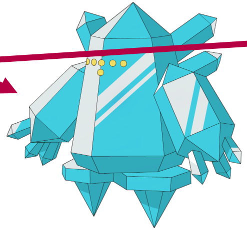
PCA – Pokémon: Part III

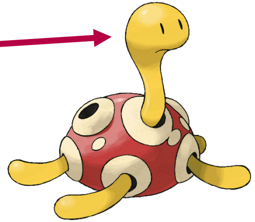
	HP	Atk	Def	Sp Atk	Sp Def	Speed
PC1	0.4	0.4	0.4	0.5	0.4	0.3
PC2	0.1	0.0	0.6	-0.3	0.2	-0.7
	-0.5	-0.6	0.1	0.3	0.6	
	0.7	-0.4	-0.4	0.1	0.2	

HP (Sacrifice Atk and Def

Sp Def(Sacrifice HP and Atk)





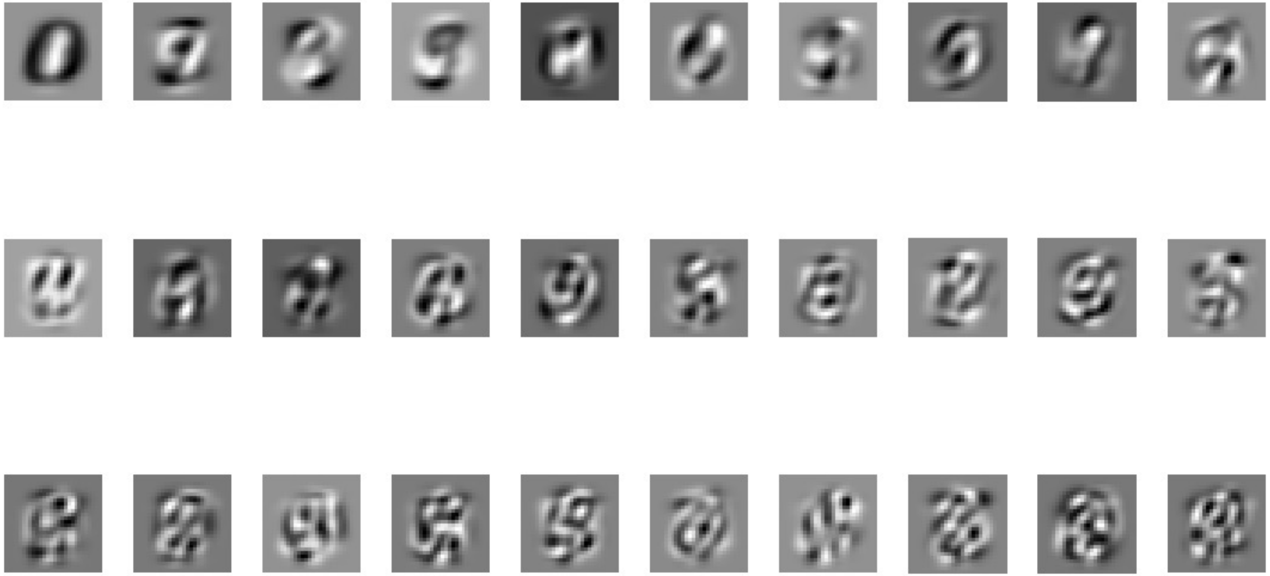


PCA - MNIST

$$= a_1 w^1 + a_2 w^2 + \cdots$$

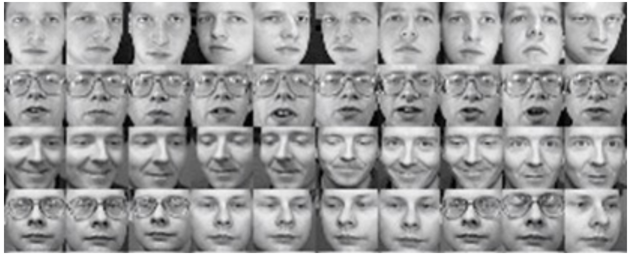
images

30 components:



Eigen-digits

PCA - Face



30 components:

http://www.cs.unc.edu/~lazebnik/research/spring08/assignment3.html

Eigen-face

Application of PCA

- Compression
  - Reduce memory/disk needed to store data
  - Speed up learning algorithm
- Visualization

Supervised Learning Speedup

$$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$$

Extract inputs:

Unlabeled dataset: $x^{(1)}, x^{(2)}, \ldots, x^{(m)} \in \mathbb{R}^{10000}$

$$\downarrow PCA$$

$$z^{(1)}, z^{(2)}, \ldots, z^{(m)} \in \mathbb{R}^{1000}$$

New training set:

$$(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \ldots, (z^{(m)}, y^{(m)})$$

Note: Mapping $x^{(i)} \rightarrow z^{(i)}$ should be defined by running PCA only on the training set. This mapping can be applied as well to the examples $x_{cv}^{(i)}$ and $x_{test}^{(i)}$ in the cross validation and test sets.

Bad use of PCA: To Prevent Overfitting

Use $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features to $k < n.$
Thus, fewer features, less likely to overfit.

This might work OK, but isn't a good way to address overfitting. Use regularization instead.

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

PCA is Sometimes Used Where it Shouldn't be

Design of ML system:

- Get training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$
- Run PCA to reduce $x^{(i)}$ in dimension to get $z^{(i)}$
- Train logistic regression on $\{(z^{(1)}, y^{(1)}), \ldots, (z^{(m)}, y^{(m)})\}$
- Test on test set: Map $x_{test}^{(i)}$ to $z_{test}^{(i)}$. Run $h_\theta(z)$ on $\{(z_{test}^{(1)}, y_{test}^{(1)}), \ldots, (z_{test}^{(m)}, y_{test}^{(m)})\}$

How about doing the whole thing without using PCA?

Before implementing PCA, first try running whatever you want to do with the original/raw data $x^{(i)}$. Only if that doesn't do what you want, then implement PCA and consider using $z^{(i)}$.