

Rozglądarka się w przestrzeni „Iris”
czyli kręcenie (głową/płaszczyzną)
w czterech wymiarach

Krzysztof Czarnowski

20 maja, 2017

- Klasyka z brodą:
 - zbiór danych *Iris*
 - analiza składowych głównych (PCA), czyli redukcja wymiaru przy pomocy algebry liniowej i przybliżenia rozkładem Gaussa
 - rzut oka na szczegóły na przykładzie Iris
- Sztuczna inteligencja, czyli jak zapomnieć matematykę i zawierzyć sieciom neuronowym
 - liniowy *auto-encoder* i PCA
 - pytania?

Zakładam, że słuchacz tylko otarł się kiedyś o algebrę wektorów i macierzy i rachunek prawdopodobieństwa. Całość jest *for dummies*. Mam nadzieję ...

Przerywać proszę! Jak coś trzeba będzie pominąć to nie szkodzi! Dyskusja z Miłymi Słuchaczami zawsze ciekawsza!

Kosaciec, czyli *Iris*

W 1936 roku *Ronald Fisher* w klasycznej pracy o klasyfikacji (liniowej) oparł się na pomiarach czterech cech trzech gatunków kwiatów z rodziny kosaćców (50 pomiarów dla każdego gatunku).

THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS

By R. A. FISHER, Sc.D., F.R.S.

I. DISCRIMINANT FUNCTIONS

WHEN two or more populations have been measured in several characters, x_1, \dots, x_n , special interest attaches to certain linear functions of the measurements by which the populations are best discriminated. At the author's suggestion use has already been made of this fact in craniometry (*a*) by Mr E. S. Martin, who has applied the principle to the sex differences in measurements of the mandible, and (*b*) by Miss Mildred Barnard, who showed how to obtain from a series of dated series the particular compound of cranial measurements showing most distinctly a progressive or secular trend. In the present paper the application of the same principle will be illustrated on a taxonomic problem; some questions connected with the precision of the processes employed will also be discussed.

180 MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS

Table I

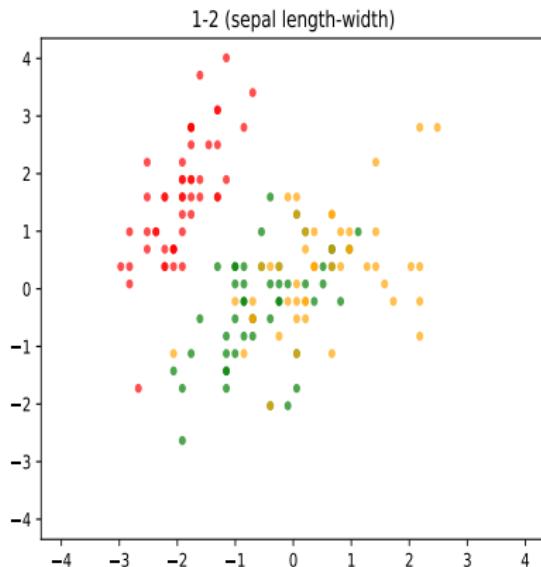
<i>Iris setosa</i>				<i>Iris versicolor</i>				<i>Iris virginica</i>			
Sepal length	Sepal width	Petal length	Petal width	Sepal length	Sepal width	Petal length	Petal width	Sepal length	Sepal width	Petal length	Petal width
5.1	3.5	1.4	0.2	7.0	3.2	4.7	1.4	6.3	3.3	6.0	2.5
4.9	3.0	1.4	0.2	6.4	3.2	4.5	1.5	5.8	2.7	5.1	1.9
4.7	3.2	1.3	0.2	6.9	3.1	4.9	1.4	5.9	2.4	5.0	1.5
4.6	3.1	1.5	0.2	5.5	2.3	4.0	1.3	6.3	2.9	5.6	1.8
5.0	3.6	1.4	0.2	6.5	2.8	4.6	1.5	6.5	3.0	5.8	2.2
5.4	3.9	1.7	0.4	5.7	2.8	4.5	1.3	7.6	3.0	6.6	2.1
4.6	3.4	1.4	0.3	6.3	3.3	4.7	1.6	4.9	2.5	4.5	1.7
5.0	3.4	1.5	0.2	4.9	2.4	3.3	1.0	7.3	2.9	6.3	1.8
4.4	2.9	1.4	0.2	6.6	2.9	4.6	1.3	6.7	2.5	5.8	1.8



iris data

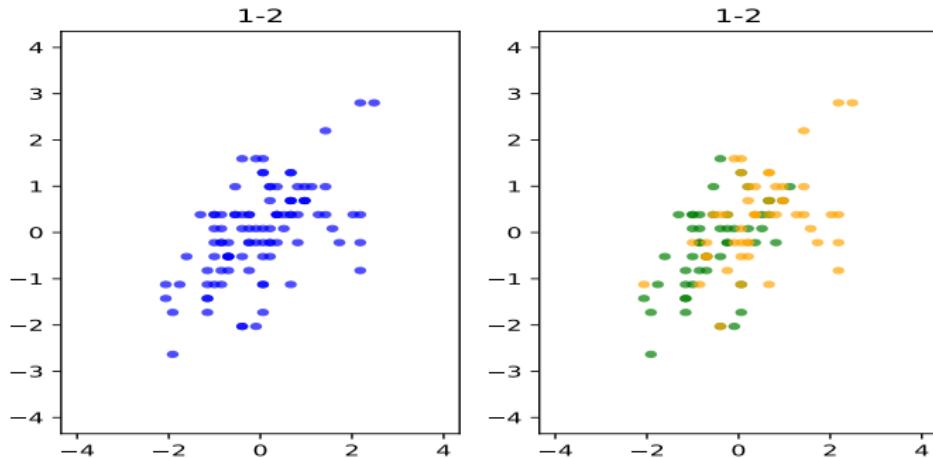
```
from sklearn import datasets, preprocessing  
irisdata = datasets.load_iris().data
```

- plus normalizacja `preprocessing.StandardScaler` — dla każdej cechy średnia 0 i wariancja 1.

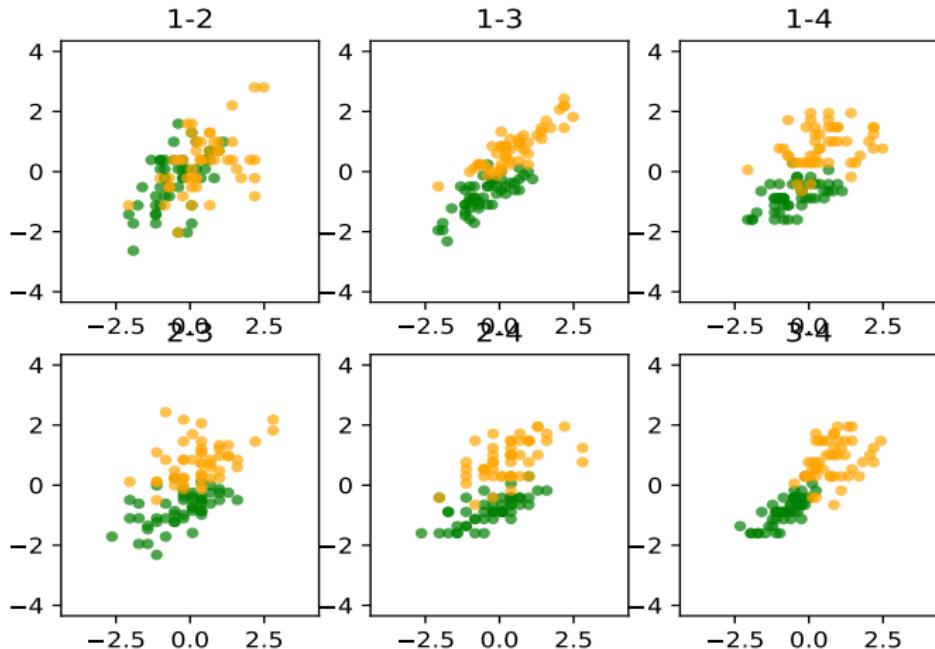


- zakres $[-4, 4]$ ustalony na sztywno dla lepszego porównywania
- w dalszym ciągu odrzucimy Iris Setosa (czerwone) bo za dobrze się oddziela
- pozostanie Versicolor i Virginica (zielone i pomarańczowe)

Ponownie dwie pierwsze cechy dla dwóch pozostawionych gatunków bez podziału i z podziałem na klasy.



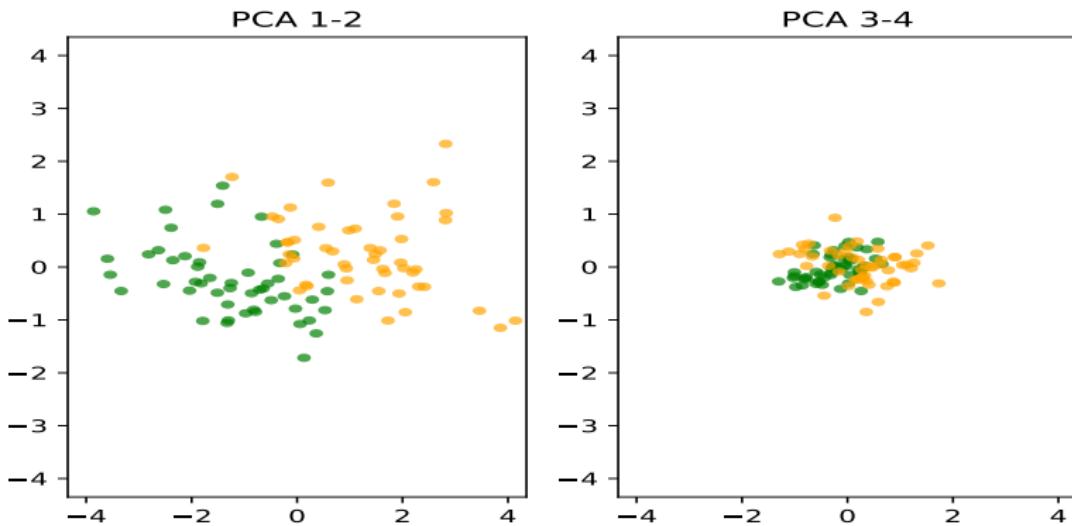
wszystkie „ściany” z podziałem na klasy



Poszukamy lepszego „punktu widzenia” ...

Kierunki najlepiej rozrzucające dane — PCA

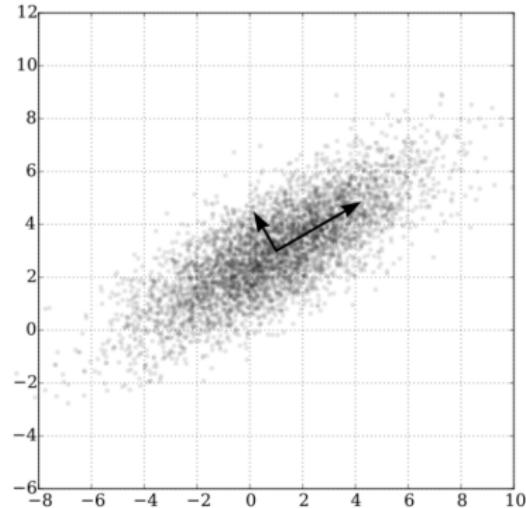
```
>>> from sklearn.decomposition import PCA  
>>> pca = PCA().fit(irisdata)  
>>> versic_pca, virgin_pca = map(  
    pca.transform, (irisversic, irisvirgin))
```



PCA — Principal Component Analysis

Próba z rozkładu gaussowskiego układają się w kształt elipsoidy. Kierunek najbardziej różnicujący dane jest kierunkiem w którym wariancja jest największa — najdłuższą osią elipsoidy.

Kolejnego kierunku szukamy w podprzestrzeni ortogonalnej do pierwszego, i tak dalej . . .



Bardziej konkretnie i abstrahując od rozkładu Gaussa, analizujemy macierz kowariancji próby (zakładam, że średnie zostały już znormalizowane):

$$S_{ij} = \frac{1}{N} \sum_{n=1}^N x_{ni} x_{nj} \quad \text{lub} \quad S = \frac{1}{N} \sum_{n=1}^N x_n^T x_n$$

Macierz kowariancji S jest symetryczna (oczywiście!) i dodatnio określona, bo wyrażenie

$$\mathbf{v}^T S \mathbf{v} = \sum_{ij} v_i S_{ij} v_j = (1/N) \sum_{nij} v_i x_{ni} x_{nj} v_j = (1/N) \sum_n (\mathbf{v} \cdot \mathbf{x}_n)^2$$

jest wariancją rzutu próby na kierunek wektora \mathbf{v}

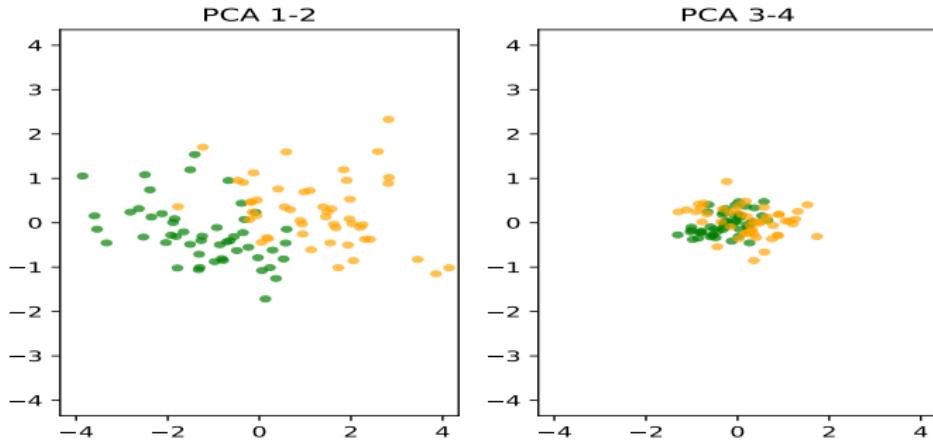
Z ogólnej teorii wynika, że:

- S ma D dodatnich wartości własnych¹ — λ_i . Założymy, że różnych — w praktyce degeneracja jest mało prawdopodobna.
- Wektory własne w_i są wzajemnie ortogonalne i w bazie z nich zbudowanej S ma postać diagonalną z wartościami własnymi na przekątnej. Elementy diagonalne to wariancje w kierunku odpowiednich wektorów własnych.

Czyli procedura jest jasna: kolejne *kierunki główne*, to kolejne wektory własne odpowiadające największym wartościom własnym!

¹ D — wymiar przestrzeni cech

Wracamy do danych . . .



```
>>> print pca.explained_variance_
[ 2.9579  0.5551  0.4057  0.0812 ]
>>> print pca.components_
[[  0.50   0.43   0.54   0.50 ],
 [  0.22  -0.88   0.37   0.13 ],
 [ -0.69  -0.05   0.01   0.71 ],
 [ -0.46   0.13   0.74  -0.45 ]]
```

Transformacja tam i z powrotem

„Składowe” rzeczywiście tworzą układ *ortonormalny*:

```
>>> U = pca.components_
>>> np.matmul(U, U.transpose())
array([[ 1.00e+00,  5.55e-17, -5.55e-17, -8.32e-17 ],
       [ 5.55e-17,  1.00e+00, -8.32e-17,  8.32e-17 ],
       [-5.55e-17, -8.32e-17,  1.00e+00, -1.66e-16 ],
       [-8.32e-17,  8.32e-17, -1.66e-16,  1.00e+00 ]])
```

Ale to także oznacza, że U^T jest macierzą odwrotną do U .

U i U^T są używane przez

`pca.transform()` i `pca.inverse_transform()`

Macierz kowariancji tu i tam

Nietrudno ręcznie przetransformować macierz kowariancji z próby:

$$S \longmapsto U S U^T$$

```
>>> S = pca.get_covariance()
>>> S
array([[ 1.   ,  0.55,  0.82,  0.59 ],
       [ 0.55,  1.   ,  0.51,  0.56 ],
       [ 0.82,  0.51,  1.   ,  0.82 ],
       [ 0.59,  0.56,  0.82,  1.   ]])
>>> S_pca = np.matmul(U, np.matmul(S, U.transpose()))
>>> S_pca
array([[ 2.95e+00,  1.11e-16, -4.99e-16, -2.60e-16 ],
       [ 1.11e-16,  5.55e-01, -6.93e-17,  1.11e-16 ],
       [ -6.66e-16, -1.11e-16,  4.05e-01, -6.93e-17 ],
       [ -1.11e-16, -2.77e-17, -5.55e-17,  8.12e-02 ]])
(pamiętamy: [ 2.9579  0.5551  0.4057  0.0812 ])
```

Jak dobrze PCA-2D tłumaczy dane?

Weźmiemy tylko dwie pierwsze składowe PCA

```
>>> N = len(irisdata)
>>> iris_pca = pca.transform(irisdata)
>>> for n in range(N): iris_pca[n,2] = iris_pca[n,3] = 0.0
```

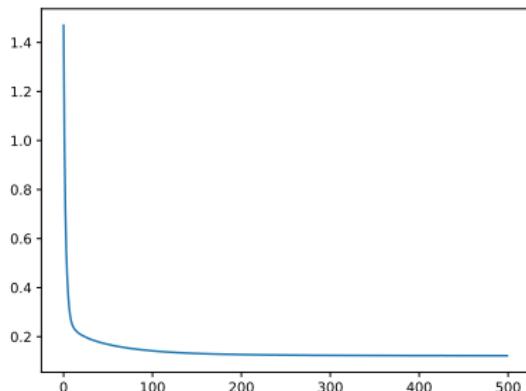
zastosujemy transformację odwrotną i obliczymy średni błąd (na wektor i cechę)

```
>>> irisdata_ = pca.inverse_transform(iris_pca)
>>> loss = sum([
... np.inner(v,v) for v in irisdata_ - irisdata ])
>>> loss, loss/4
(0.486977, 0.121744)
```

Zapamiętajmy ostatnią liczbę.

Czas na sztuczną inteligencję!

```
from keras.models import Sequential  
from keras.layers import Dense  
gmdl = Sequential()  
gmdl.add(Dense(2, input_dim=4, bias=False))  
gmdl.add(Dense(4, bias=False))  
gmdl.compile(optimizer="sgd", loss="mean_squared_error")  
gmdl.fit(irisdata, irisdata, batch_size=10, nb_epoch=1000,  
         validation_data=(irisdata, irisdata))
```



Liniowy *auto-encoder*:

$\text{loss}[100] = 0.14194$

$\text{loss}[200] = 0.12708$

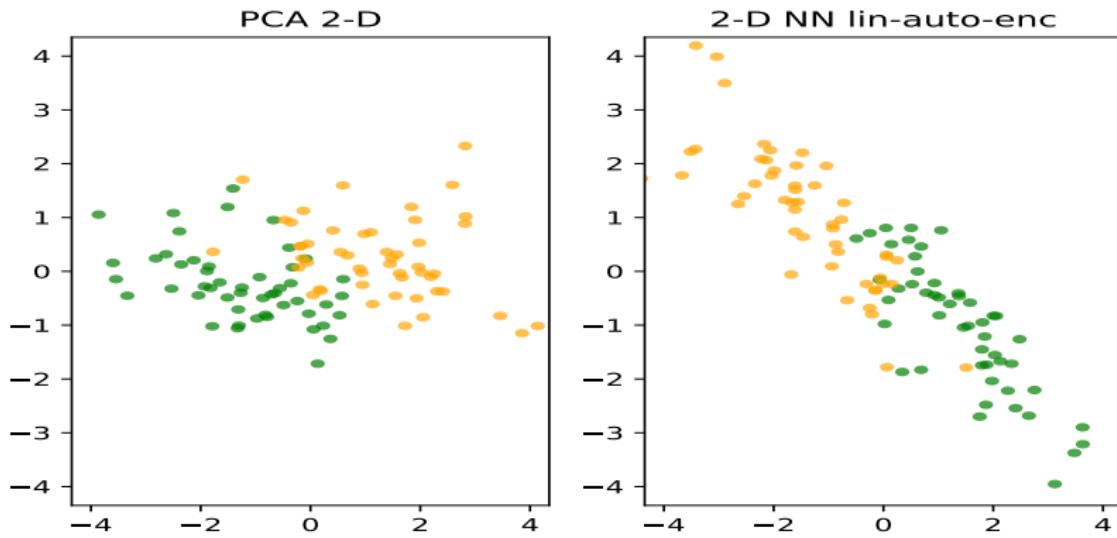
$\text{loss}[400] = 0.12243$

$\text{loss}[999] = 0.12184$

(dla PCA mieliśmy:
0.12174)

Jak wydobyć zanurzenie?

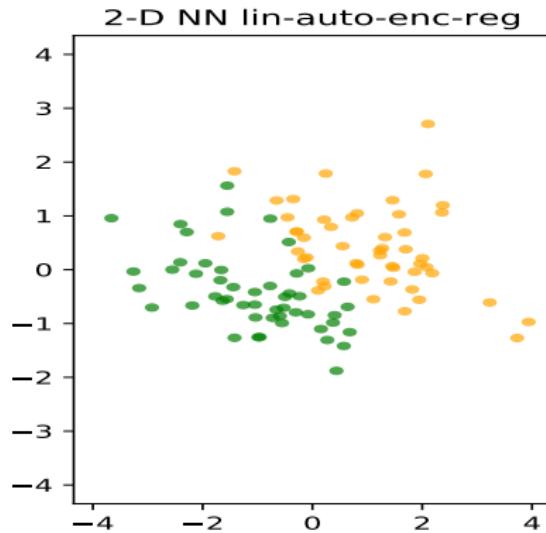
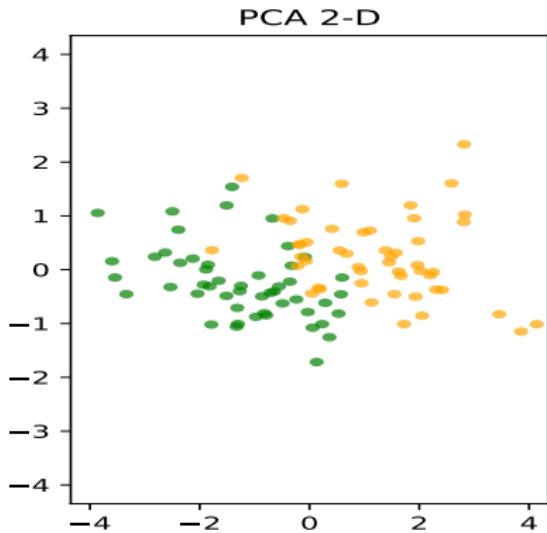
```
emdl = Sequential()  
emdl.add(Dense(2, input_dim=4, bias=False,  
               weights=gmdl.layers[0].get_weights()))  
emdl.compile(optimizer="sgd", loss="mean_squared_error")  
emdl.predict(irisdata, batch_size=10)
```



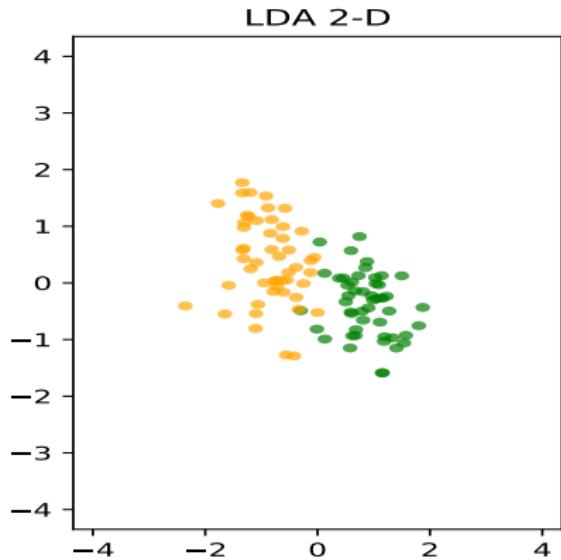
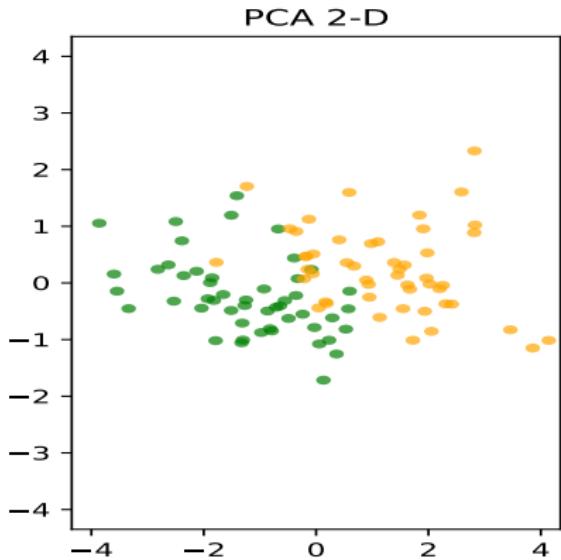
Pytania różne

- Czy jesteśmy zadowoleni z efektów działania modelu „neuronowego”?
- Może jesteśmy zdziwieni wynikiem?
- Jaki problem wiąże się z tym modelem?
- Warto przyjrzeć się jak zmienia się zanurzenie w czasie uczenia?
- Jak skłonić sieć do poszukiwania tego samego modelu co PCA?
- W jakiś „naturalny” sposób?
- Jak skłonić sieć żeby zrobiła LDA?
- Inne pytania?

Lepiej?



- Praca Fishera nie była o PCA, lecz o optymalnych *klasyfikatorach liniowych*, czyli kierunkach maksymalnie *różnicujących klasy*.
- PCA nie widzi klas! (ale rysunki są ładniejsze!)



Wikipedia credits

CC BY-SA 3.0,

<https://commons.wikimedia.org/w/index.php?curid=170298>

CC BY-SA 3.0,

<https://commons.wikimedia.org/w/index.php?curid=248095>

By Frank Mayfield—originally posted to Flickr as Iris virginica shrevei
BLUE FLAG, CC BY-SA 2.0,

<https://commons.wikimedia.org/w/index.php?curid=9805580>

CC BY-SA 3.0,

<https://commons.wikimedia.org/w/index.php?curid=109679>

By Nicoguaro - Own work, CC BY 4.0,

<https://commons.wikimedia.org/w/index.php?curid=46871195>