Embeddings

# 6. Unsupervised learning

Supervised learning

* What is the conditional distribution of ?
* How do I build a predictor / classifier?
* Appropriate plots: line plots, box plots, …

Generative (unsupervised( learning

* What is the joint distribution of ?
* How do I synthesize new datapoints?
* Appropriate plots: scatter / dot plots, histograms, splom, …

# 7. Dimension reduction

## **PCA**

In linear regression (**supervised learning**), we model the data by

and choose the parameters and to minimize

In 1-dimensional PCA (**unsupervised learning**), we model the data by

and choose the parameters and to minimize

More generally, we model -dimensional data by picking a -dimensional subspace, and representing each point by its projection onto that subspace,

Magic linear algebra trick:

* the optimal subspaces are nested,
* so the sensible choice is to let be an ordered basis, , ...

X = countries[features].values

pca = sklearn.decomposition.PCA()

pca\_result = pca.fit\_transform(X)

μ = pca.mean\_

pred = μ + np.zeros\_like(pca\_result)

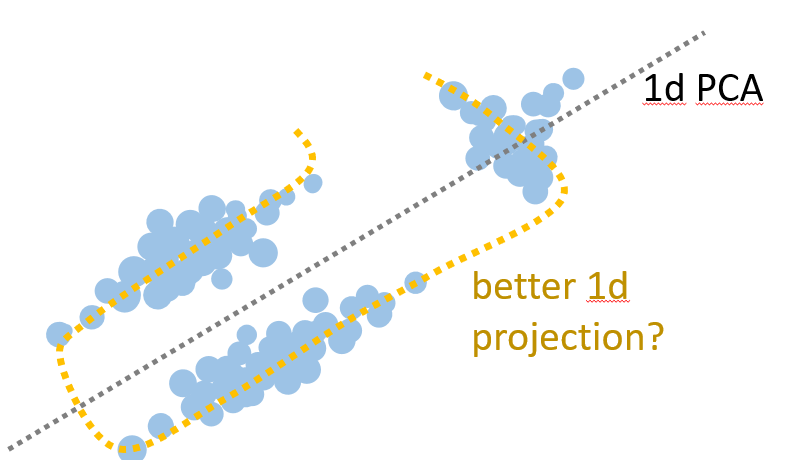
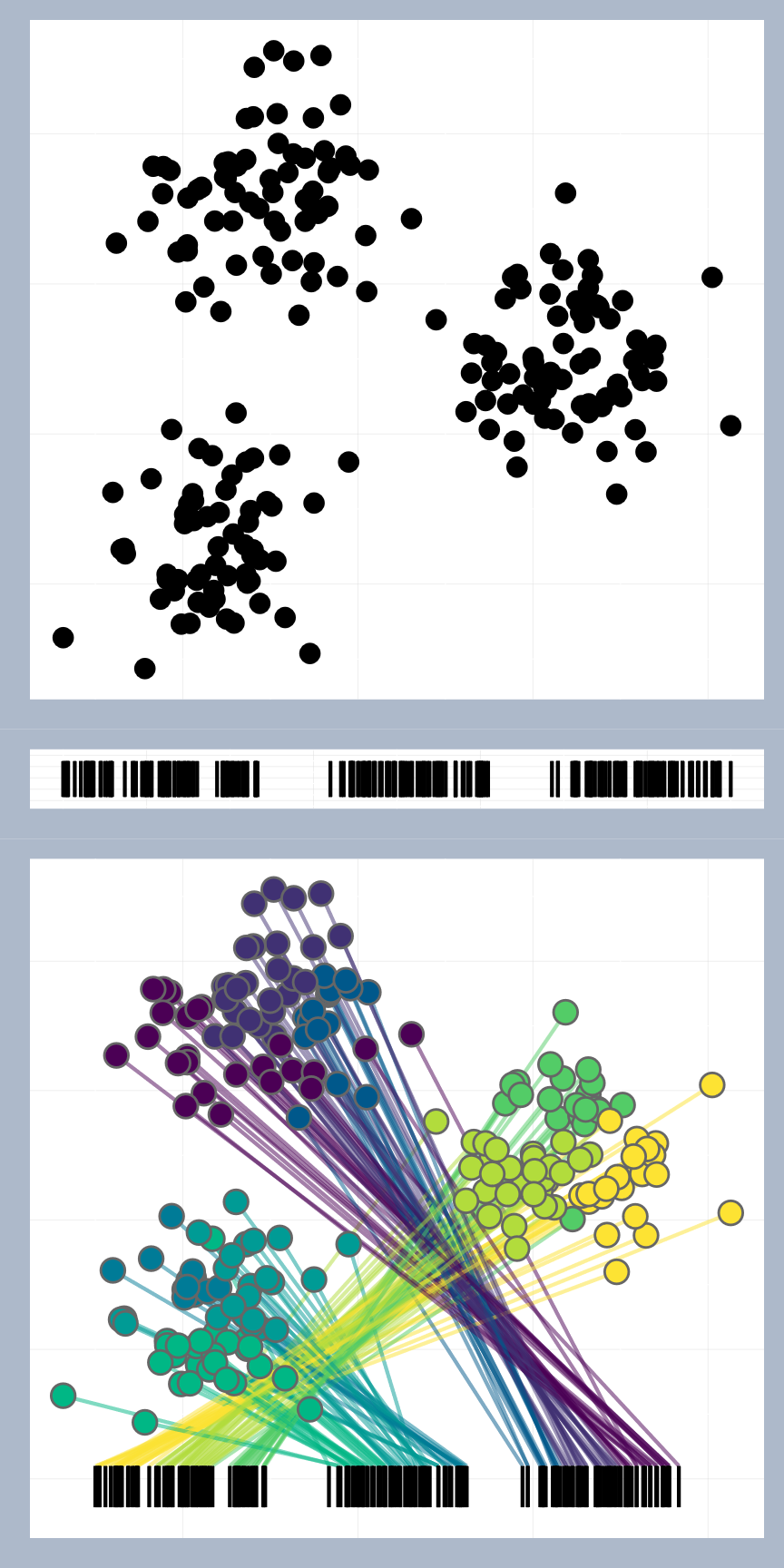
for k in range(L): # L = number of PCA components to use

λk = pca\_result[:,k]

δk = pca.components\_[k]

pred = pred + λk.reshape((-1,1)) \* δk.reshape((1,-1))

## **tSNE is more suitable for non-linear dimension reduction**

* tSNE finds a low-dimensional representation based on “make it so that the top- neighbours of a point in the low-dimensional space match its top- neighbours in the high-dimensional data space”. The hyperparameter is called *perplexity*.

X = countries[features].values

# scale the columns, so they have the same variance

for k in range(len(features)):

X[:,k] = X[:,k] / np.std(X[:,k])

# K = number of dimensions to reduce to

tsne = sklearn.manifold.TSNE(n\_components=K)

tsne\_results = tsne.fit\_transform(X)

# K=2 is nice for plotting

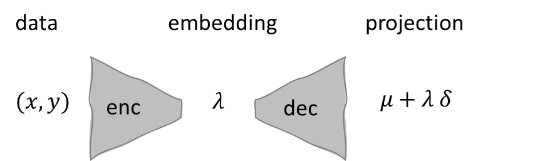
p1,p2 = tsne\_results[:,0], tsne\_results[:,1]

plt.scatter(p1, p2, alpha=.2)

## **Uses for dimension reduction**

* Split the data into qualitatively different test and validation subsets, so that validation gives a better test of generalization error
* Take a categorical feature with 100s of values, one-hot encode it to give 100s of features, dimension-reduce to a few features, and perhaps your ML system will run faster

# 8. Self-supervised learning / embedding



* Don’t think of PCA and tSNE as “tools I can run on a dataset to reduce its dimension”
* Think of them as “learning an encoder (embedding function) and a decoder (reconstruction function), from a training dataset”. The encoder and decoder can be applied to *new unseen* *datapoints*.
* This is useful for semi-supervised learning, and for other interpolation / extrapolation / interpretation questions.

# 9. Content scales

* Any nominal or ordinal scale can (and should) be embedded, based on whatever content we’re interested in.
* We don’t need a formal mathematical embedding – an embedding is really “any order that will help the viewer”.
* Examples
  + Embed a Likert 5-point scale (“strongly agree” … “strongly disagree”) into real numbers in based on frequency of responses.
  + Embed students onto a 1d scale based on clustering proximity, to more easily see patterns in turn-it-in similarity scores

# Next steps

* Style… Tufte, The visual display of quantitative information.
* How to tell a story… Berinato, *Good Charts*
* Software libraries… ggplot2 is unrivalled, and it’s not too hard to use it from Python.