Boring but important disclaimers:

- If you are not getting this from the GitHub repository or the associated Canvas page (e.g. CourseHero, Chegg etc.), you are probably getting the substandard version of these slides Don't pay money for those, because you can get the most updated version for free at

  `https://github.com/julianmak/OCES4303_ML_ocean`

  The repository principally contains the compiled products rather than the source for size reasons.
- Associated Python code (as Jupyter notebooks mostly) will be held on the same repository. The source data however might be big, so I am going to be naughty and possibly just refer you to where you might get the data if that is the case (e.g. JRA-55 data). I know I should make properly reproducible binders etc., but I didn't...
- I do not claim the compiled products and/or code are completely mistake free (e.g. I know I don't write Pythonic code). Use the material however you like, but use it at your own risk.
- As said on the repository, I have tried to honestly use content that is self made, open source or explicitly open for fair use, and citations should be there. If however you are the copyright holder and you want the material taken down, please flag up the issue accordingly and I will happily try and swap out the relevant material.

OCES 4303 :

an introduction to data-driven and ML methods in ocean sciences

Session 3: some linear models and dimension reduction

Outline

- some linear models
  → Linear Regression, Ridge, Lasso, Elastic-Net
  → gradient descent to obtain model

  (avoiding probability based ones somewhat)

- dimension reduction
  → PCAs, locally linear embedding, t-SNE

- demonstration: eigencat

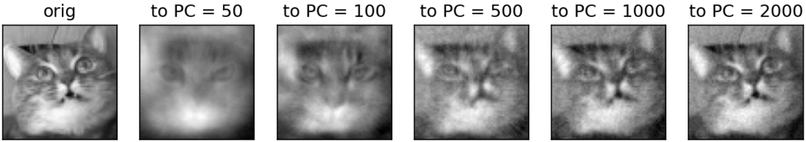


Figure: Result from an eigencat projection (it's really just PCAs).

Linear models

- given multiple features $X = (X^{(1)}, X^{(2)}, \ldots)$ and a **single** target Y, a linear model considers

  $$\hat{Y} = a_0 + a_1 X^{(1)} + a_2 X^{(2)} + \ldots a_N X^{(N)}$$

  $\rightarrow$ the model parameters or control variables are $\{a_0, a_1, \ldots, a_N\}$

  $\rightarrow$ basically a linear algebra type problem (e.g. invert the matrices)

  $\rightarrow$ different methodologies take different loss functions J (see later)

  $\rightarrow$ these were termed **multi**-linear models in OCES 3301

- NOTE: this is not multi-variate regression where we have **multiple** targets Y, but see GLMs

Linear models: Linear Regression

- standard linear regression takes

$$J = \|\hat{Y} - Y\|_{L^2}^2 = \sum_{i=0}^{M} |\hat{Y}_i - Y_i|^2,$$

where Y_i is the training data and $\hat{Y}_i$ are the predictions from the model

$\rightarrow$ note lower index, M samples, but N lots of features described by $X^{(j)}$

$\rightarrow$ this basically provides the 1d LOBF embedded in $(N+1)$d space

$\rightarrow$ can have issues when features (the $X^{(j)}$) are not independent of each other

$\rightarrow$ can be sensitive to data noise

Linear models: Ridge

- consider controlling the size of the model coefficients a by

$$J = \|\hat{Y} - Y\|_{L^2}^2 + \alpha \|a\|_{L^2}^2$$

  $\rightarrow$ this is peanlisation on a with strength α

- extra hyper-parameter α here

  $\rightarrow$ large α means small coefficients and control colinearity, but at then expense of skill

- NOTE: different αs needed if data is not standardised

Linear models: Lasso

- could even limit number of features by asking for as many zero a_j as possible, via

$$J = \frac{1}{2M} \|\hat{Y} - Y\|_{L^2}^2 + \alpha \|a\|_{L^1}$$

$\rightarrow$ large α we want to keep as many non-zero coefficients as possible, but at then expense of skill

$\rightarrow$ relations to **compressed sensing**

Linear models: Elastic-nets

- could do a bit of both via

$$J = \frac{1}{2M}\|\hat{Y} - Y\|_{L^2}^2 + \alpha\rho\|a\|_{L^1} + \frac{\alpha(1-\rho)}{2}\|a\|_{L^2}^2$$

$\rightarrow$ ρ is a ratio hyper-parameter controlling sparsity and model parameter values **(cross-validation needed!!)**

Sample loss function + gradient descent

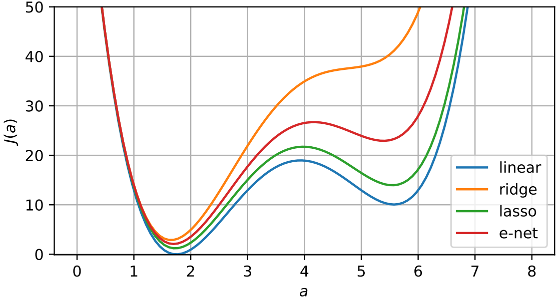


Figure: Made up 1d example of $J = J(a)$ with different choices of regularisation, $J = (1/2M)\|\hat{Y} - Y\|_{L^2}^2 + \alpha\rho\|a\|_{L^1} + (1/2)\alpha(1-\rho)\|a\|_{L^2}^2$.

- optimise by finding minimum, extremum occurs where the derivatives are zero $\Rightarrow$ root finding problem

  $\rightarrow$ **gradient** information is used to find direction of **descent**

Example: regressing to polynomial data

- going to do a demonstration case of taking

$$Y = X^2 + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma)$$

where σ controls the level of ‘noise‘, but going to fit this to

$$\hat{Y} = a_0 + a_1 X^1 + a_2 X^2 + \ldots a_{10} X^{10}$$

i.e. my features $X^{(j)}$ in this case happens to polynomials X^j (X raised to power j)

$\rightarrow$ might expect to get $a_2 = 1$ and small values elsewhere

$\rightarrow$ a slightly more subtle meaning to “linear” is needed...

Example: regressing to polynomial data

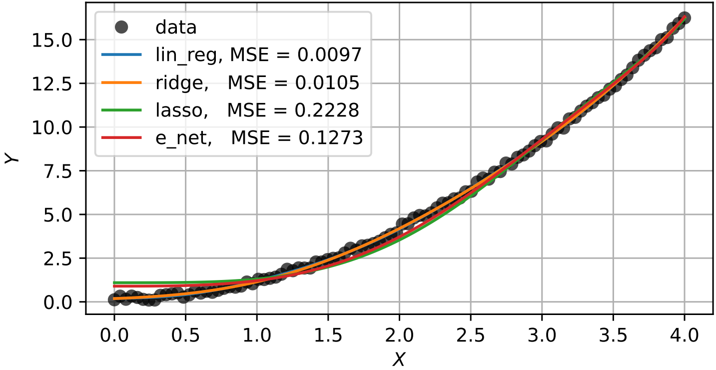


Figure: Polynomial regression for some contrived data, with MSE scores printed on.

Example: regressing to polynomial data

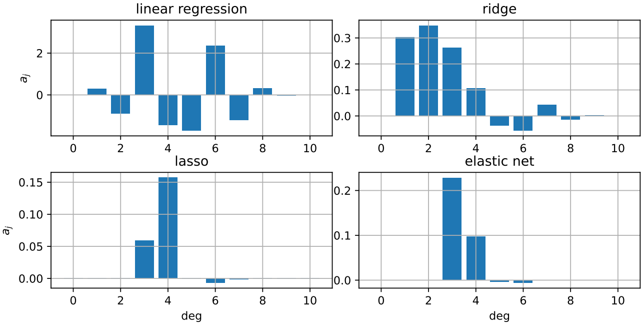


Figure: Polynomial regression for some contrived data, with MSE scores printed on.

- notice a_2 is not always largest!
- notice L^1 based penalisation (lasso and elastic net) gives more small coefficients

  $\rightarrow$ fewer features (more likely not over-fitted?)

Dimension reduction: PCAs

Q. if not all features are as important, can find the important ones *a priori*?

$\rightarrow$ would help presumably against over-fitting

Dimension reduction: PCAs

Q. if not all features are as important, can find the important ones *a priori*?

$\rightarrow$ would help presumably against over-fitting

- Principal Component Analysis (PCA) is one way

  $\rightarrow$ seen already from OCES 3301

  $\rightarrow$ shows up also as Empirical Orthogonal Functions (EOFs) particularly when pattern are involved

- considers linear combinations of features that explain the most variance in the data

  $\rightarrow$ effectively does a SVD, returns

  1. singular values σ (related to variance explained)
  2. the left and right singular vectors that gives the new co-ordinate system and relevant expansions

Dimension reduction: PCAs

- easier visually with the penguins example maybe
  $\rightarrow$ data has been standardised here

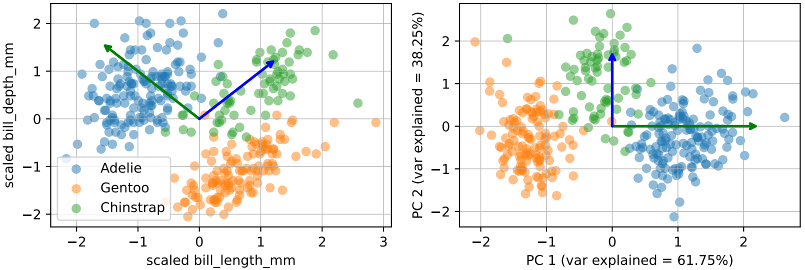


Figure: (Left) Original (but standarised) data with two choices of variables. (Right) The same but in the principal component representation.

- for this case it really is just a co-ordinate transformation

Dimension reduction: PCAs

- below shows dimension reduction (from 4 to 2)

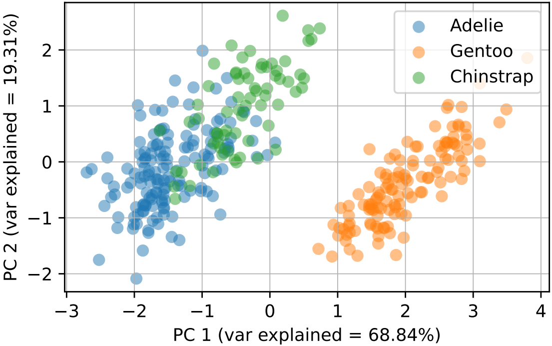


Figure: Representation in PC1 and PC2 co-ordinate system.

- note the variance explained does not sum to 100% here
  $\rightarrow$ this is now a **projection** which loses information

Dimension reduction: manifold methods

(This part makes more sense after next session)

- other manifold based methods

  → Locally Linear Embedding (preserves local distances, think local PCAs to give global co-ord)

  → *t*-distributed Stochastic Neighbor Embedding (*t*-SNE; based on pdfs, good at picking out local structures)

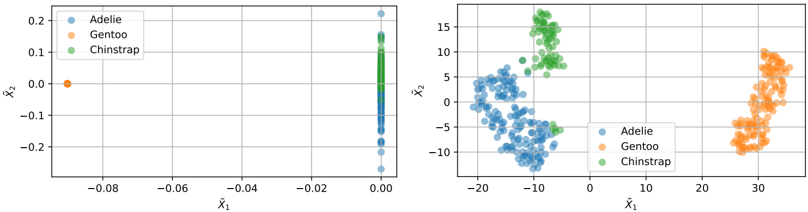


Figure: Demonstration of Locally Linear Embedding and *t*-SNE on the penguins data.

Demonstration: eigencat

- example demonstrating PCA on images of cats

  → feature extraction, cf. facial recognition
- massage data into (`n_cat,pixels`) then throw into PCA

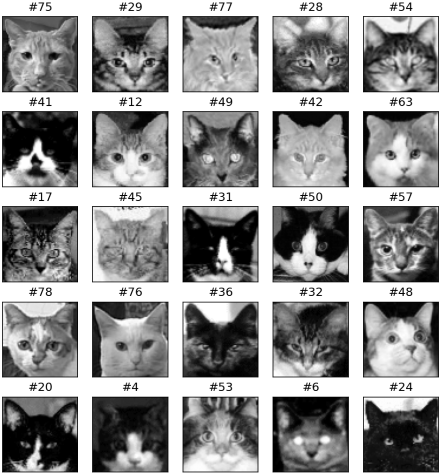


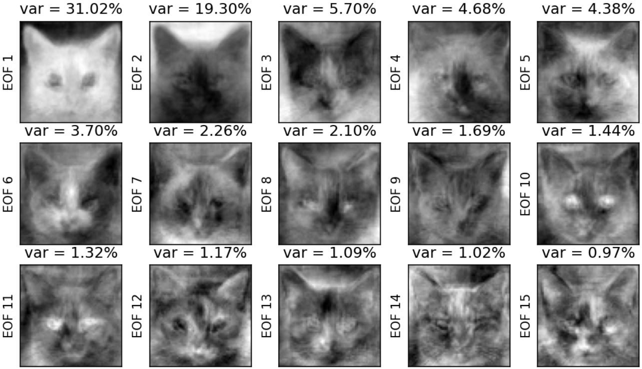
Figure: Data in `cats.csv`.

Demonstration: eigencat

- want

$$\text{Image(pixels)} = \sum_i \text{PC}_i \times \text{EOF}_i(\text{pixels}),$$

$\rightarrow$ call the patterns the EOFs, PCs are the loadings
$\rightarrow$ reshape EOFs to get "averaged" cat faces



Demonstration: eigencat

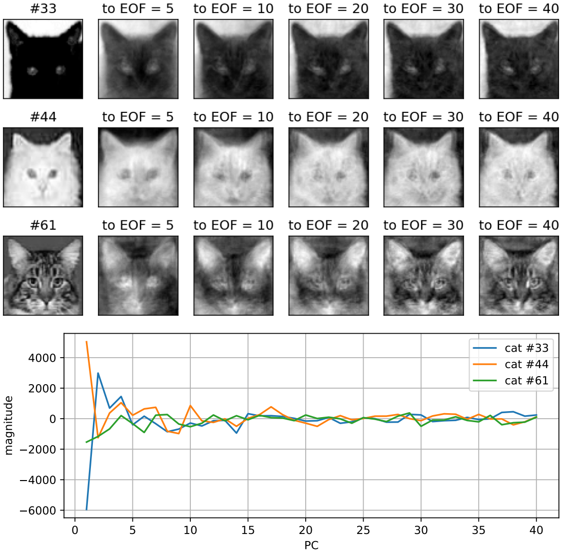


Figure: Expansion and loadings of three of the samples.

Demonstration: eigencat

- different cats will have a different PC pattern, use that as
  $\rightarrow$ signature for classification?
  $\rightarrow$ features for model training? (much lower dimension)

- warning: can start doing ridiculous things like below if my data space is large enough!
  $\rightarrow$ next few pages show the same thing if I use an expanded dataset (2000 images)

Demonstration: eigencat

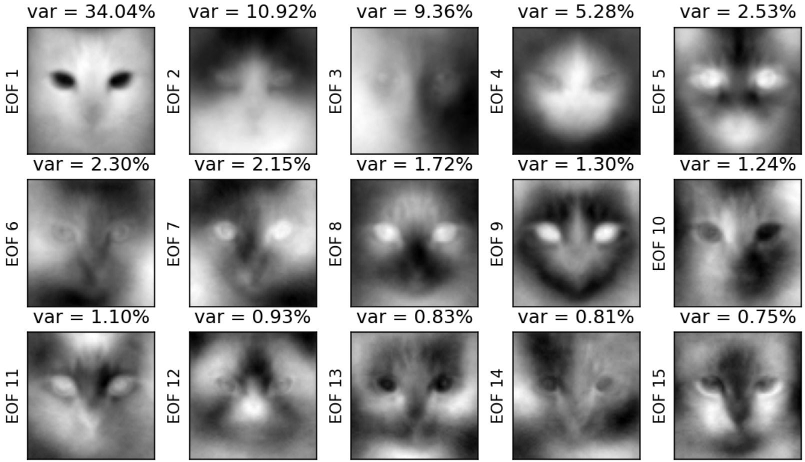


Figure: Eigencats from `cats_bw_enlarged.csv`.

Demonstration: eigencat

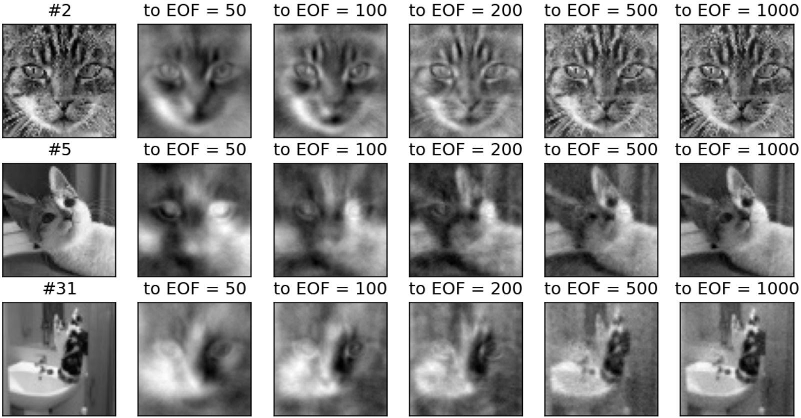


Figure: Expansion of a few selected images from the expanded dataset.

Demonstration: eigencat

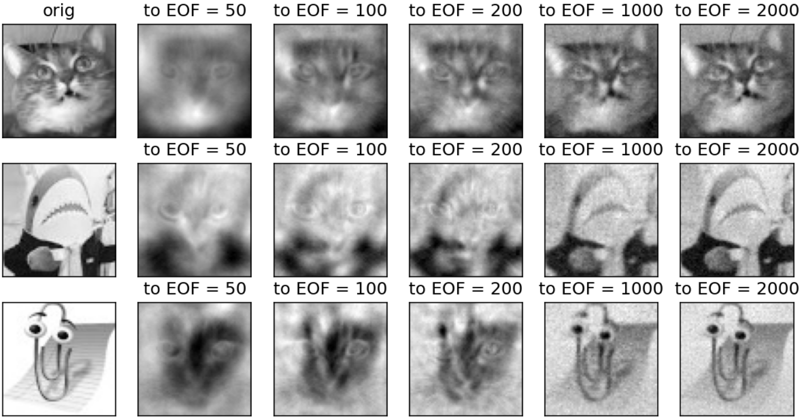


Figure: Eigencat expansion of the three ad hoc TAs (which were not in the dataset).

Demonstration

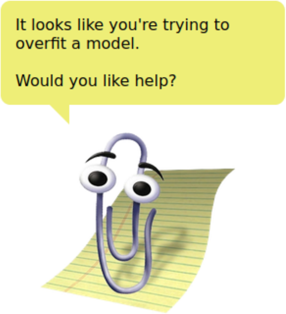


Figure: The OG AI (which was mega annoying by the way).

- demonstrating some linear models

  $\rightarrow$ avoided probability based linear models (see notebook for references)
- basics and extended example of dimensional reduction and feature identifications

Moving to a Jupyter notebook $\rightarrow$