Fighting the curse of dimensionality

Artificial Intelligence and Machine Learning for SupTech – Lecture 4

Iman van Lelyveld – Michiel Nijhuis

VU Amsterdam

Fighting the curse of dimensionality

1. How to reduce dimensionality?
   - Principal Components Analysis (PCA)
2. Feature selection and regularization
   - How to tune model input by selecting features and beat overfitting?
   - How to select the most important features?
   - Examples RIDGE, LASSO, Elastic net
3. Is a “good” model always good? What is external validity?
   - Holdout, K-fold cross validation, Stratified K-fold. Leave-one-out (LOO)

Outline

Knowledge clips

- Principal component analysis – PCA (Statquest) (link)

  The key concepts are covered until 12.35.
- RIDGE regressions (Statquest) (link)

  This video is a bit slow but does cover RIDGE in detail
- LASSO regressions (Statquest) (link)

  If you have just seen RIDGE regression, you can start at 2.40
- Bonus: Playing around with Eigenvectors (Victor Powell and Lewis Lehe)

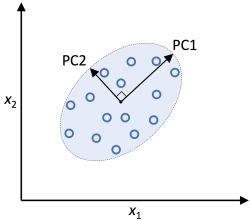
Outline

Data and feature selection

- Economic theory and practitioner knowledge is usually the starting point for looking for a parsimonious model
- Examining the correlation matrix of the features included can provide insight in (non-linear) relations/dependencies
- However, if the # of features increases, statistical methods to reduce dimensionality are called for: Principal Component Analysis (PCA)
- Also, we can aim to reduce the number of features
  - Least Absolute Shrinkage and Selection Operator (LASSO)
  - Least Angle Regression (LARS), RIDGE (Hastie et al. (2017))
- Efficient feature selection can also save computational and memory resources, in addition to improving model performance.

Principal Component Analysis (PCA)

- Find the directions of maximum variance
  - Transforming/Projecting d-dimensional data to k dimensions ($k << d$)
- Principal components: PC1 and PC2
  - First principal component will have the largest variance
  - Second principal component will have next largest variance, etc, ...
- PCA sensitive to scaling, so need to standardize features



See "Principal component analysis" – PCA (Statquest) in Knowledge clips.

A roadmap for PCA

Steps:

1. Standardize the d-dimensional dataset
2. Construct the covariance matrix
3. Decompose the covariance matrix into its eigenvectors and eigenvalues
4. Select k eigenvectors that correspond to the k largest eigenvalues, where k is the dimensionality of the new feature subspace ($k \leq d$).
5. Construct a projection matrix $\mathbf{W}$ from the "top" k eigenvectors
6. Transform the d-dimensional input dataset $\mathbf{X}$ using the projection matrix $\mathbf{W}$ to obtain the new k-dimensional feature subspace
7. See Statquest link for PCA Step-by-Step explanation.

Step 2 Variance-covariance matrix

- Symmetric $d \times d$ -dimensional matrix (d - number of dimensions)
- Pairwise covariances between the different features
- Covariance between two features $\mathbf{x}_j$ and $\mathbf{x}_k$:

$$cov(x_j, x_k) = \frac{1}{n} \sum_{i=1}^{n} \left(x_j^{(i)} - \mu_j\right)\left(x_k^{(i)} - \mu_k\right)$$

  Where μ_j and μ_k are the sample means of feature j and k (i.e., expected values)
- Covariance can be standardized to yield the correlation

$$\rho_{j,k} = \frac{cov(x_j, x_k)}{\sigma_j \sigma_k}$$

What is covariance?

- Measure of how much two random variables change together

What is covariance?

- Measure of how much two random variables change together
- Positive covariance
  - Features increase and decrease together
  - e.g. as a balloon is blown up it gets larger in all dimensions

What is covariance?

- Measure of how much two random variables change together
- Positive covariance
  - Features increase and decrease together
  - e.g. as a balloon is blown up it gets larger in all dimensions
- Negative covariance
  - Features vary in opposite directions
  - Large values of one variable correspond to small values of the other
  - e.g. if a balloon is squashed in one dimension then it will expand in the other two

What is covariance?

- Measure of how much two random variables change together
- Positive covariance
  - Features increase and decrease together
  - e.g. as a balloon is blown up it gets larger in all dimensions
- Negative covariance
  - Features vary in opposite directions
  - Large values of one variable correspond to small values of the other
  - e.g. if a balloon is squashed in one dimension then it will expand in the other two
- The magnitude of the covariance is not easy to interpret

What is covariance?

- Measure of how much two random variables change together
- Positive covariance
  - Features increase and decrease together
  - e.g. as a balloon is blown up it gets larger in all dimensions
- Negative covariance
  - Features vary in opposite directions
  - Large values of one variable correspond to small values of the other
  - e.g. if a balloon is squashed in one dimension then it will expand in the other two
- The magnitude of the covariance is not easy to interpret
- PM: The normalized version of covariance (correlation coefficient) indicates the strength of the linear relation

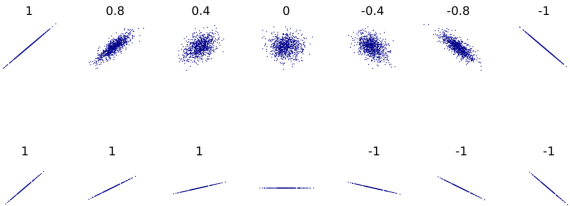
Linear dependence – Correlation

- correlation indicates the degree of the linear co-movement between two variables ...

Anscombe (1973)

Linear dependence – Correlation

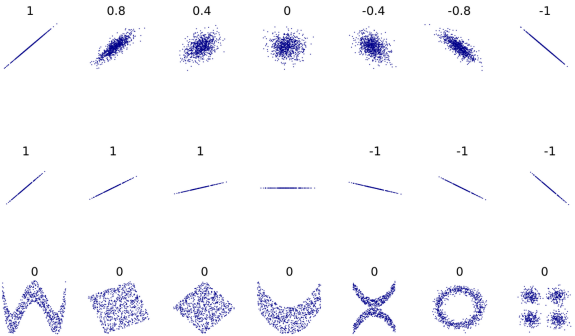
- correlation indicates the degree of the linear co-movement between two variables ...
- ... but does not say anything about the informativeness



Anscombe (1973)

Linear dependence – Correlation

- correlation indicates the degree of the linear co-movement between two variables ...
- ... but does not say anything about the informativeness
- Nor anything about non-linearity



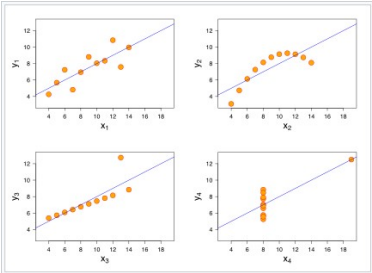
Anscombe (1973)

Outliers matter

Anscombe's quartet (Anscombe (1973))

Clockwise from top left:

- simple linear relationship
- not linear and the Pearson correlation coefficient is not relevant. Regression with non-linear features
- one high-leverage point is enough to produce a high correlation coefficient even without a relationship between the variables
- linear but with a different regression line: slope coefficient from 1 to 0.816



Step 3 Eigenvectors and eigenvalues

- For two features, covariance matrix will look like this:

$$A = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix}$$

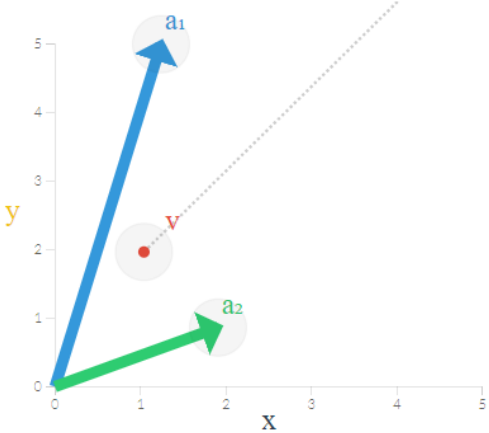
- The eigenvector of A represent the principal components: direction of maximum variance
- The corresponding eigenvalues represent their magnitude
- More formally: An Eigenvector $\mathbf{v}$ satisfies the condition:

$$A\mathbf{v} = \lambda\mathbf{v}$$

where λ is the eigenvalue (scalar)

See "Playing around with Eigenvectors" (Powell and Lehe) in Knowledge clips

Interacting with eigenvectors and eigenvalues



$$A = \begin{bmatrix} a_{1,x} & a_{2,x} \\ a_{1,y} & a_{2,y} \end{bmatrix} = \begin{bmatrix} 1.24 & 1.91 \\ 5.00 & 0.86 \end{bmatrix}$$

$$v = \begin{bmatrix} 1.04 \\ 1.97 \end{bmatrix}$$

$$Av = \begin{bmatrix} 5.04 \\ 6.90 \end{bmatrix}$$

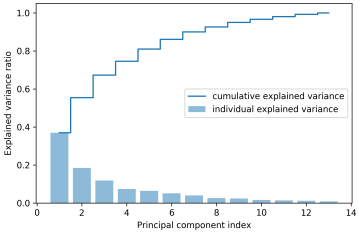
Source: Setosa.io

Variance explained ratio

- Variance explained ratio of an eigenvalue λ_j:

$$\frac{\lambda_j}{\sum_{j=1}^{d} \lambda_j}$$

- First two principal components explain about 60 percent of the variance in the data
- Choosing the 'optimal' number of PCs: elbow? ad hoc?



Outline

$L1$ regularization

Recall from our discussion on Logit, $L2$ regularization – one approach to reduce model complexity

$$L2 : \|\mathbf{w}\|_2^2 = \sum_{j=1}^{m} w_j^2$$

An alternative approach is $L1$ regularization:

$$L1 : \|\mathbf{w}\|_1 = \sum_{j=1}^{m} |w_j|$$

$L1$ regularization

- $L1$ yields sparse solutions
- Most feature weights will be zero
- Useful for high-dimensional datasets with irrelevant features
- It can be viewed as a technique for feature selection
- Some intuition as to why this is the case will follow. For OLS, the cost function becomes:

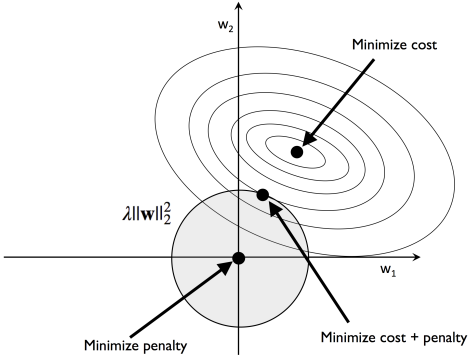
$$J(\mathbf{w}) = \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2 + \lambda \|\mathbf{w}\|$$

which becomes a regular OLS with $\lambda \to 0$

Outline

Applying regularization – Lasso

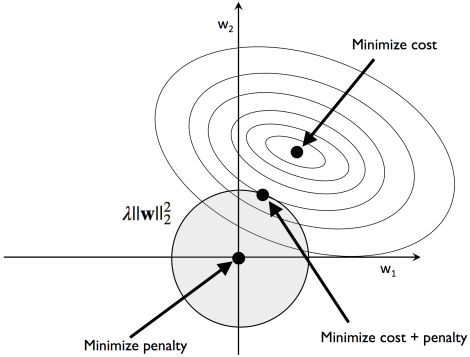
L2: RIDGE



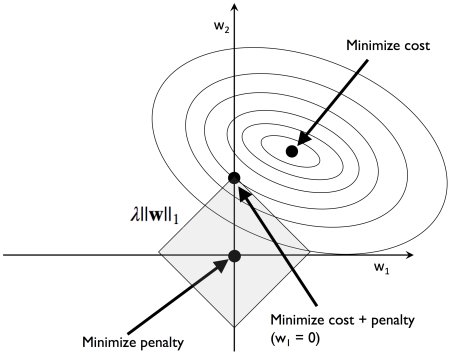
See both Statquest clips in Knowledge clips

Applying regularization – Lasso

*L*2: RIDGE



*L*1: Lasso



See both Statquest clips in Knowledge clips

Sparcity

- Regularization penalty and cost pull in opposite directions
- Regularization wants the weight to be at (0, 0)
  - i.e. regularization prefers a simpler model
  - note that Lasso can have a zero weight on a feature (because the 'triangle' has corners where one of the weights is zero)
- Decreases the dependence of the model on the training data

$L1$ in scikit-learn

Balancing Ridge and Lasso: Elastic Net

- Limitation of Ridge Regression
  - Ridge regression decreases the complexity of a model but does not reduce the number of variables since it never leads to a coefficient been zero rather only minimizes it. Hence, this model is not good for feature reduction.
- Limitation of Lasso Regression
  - If there are two or more highly collinear variables then LASSO regression select one of them randomly which is not good for the interpretation of data
  - Lasso sometimes struggles with some types of data. If the number of predictors (p) is greater than the number of observations (n), Lasso will pick at most n predictors as non-zero, even if all predictors are relevant (or may be used in the test set).

Solution: weigh Ridge and Lasso $\rightarrow$ Elastic Net

Outline

How do we know the model is working?

- Model evaluation
  - Performance metrics (discussed in Lecture 3 - ML – the basics) indicate how good the model is
- How do we obtain an unbiased estimate of model's performance?
- Key concept: estimate model performance on unseen data
- So we need to separate data for 1) finding the right model and for 2) assessing it
- Note the difference between model parameters (e.g. weights) vs hyperparameters (e.g. de k in k-nearest neighbors)

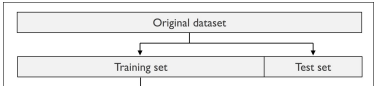
There are numerous ways to check validity

1. Holdout
2. k-fold
3. Stratified k-fold
4. Leave-one-out
5. ...

See SKLearn documentation for further approaches

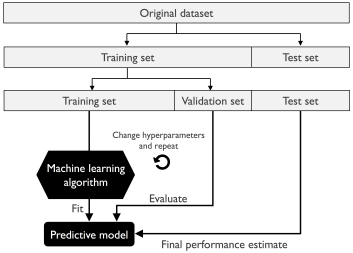
1. The holdout method

- Split data into training and test datasets



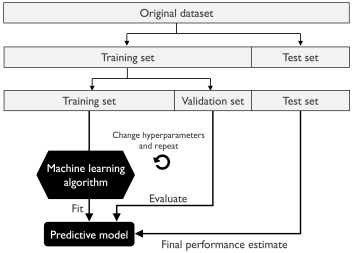
1. The holdout method

- Split data into training and test datasets
- However, typically we cannot test immediately after training
  - Need to tune the model to further improve the performance
  - Select optimial values of hyperparameters
- This step is known as model selection
- A better approach: training set + validation set + test set
  - Validation set is used for model selection
  - Test set is for model evaluation



1. The holdout method

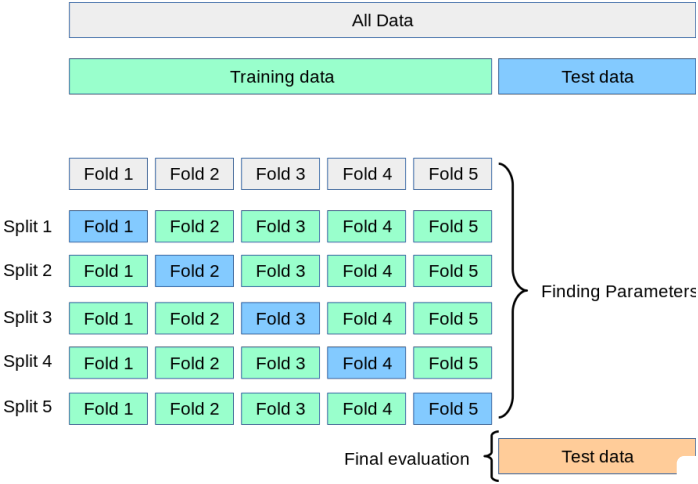
- Split data into training and test datasets
- However, typically we cannot test immediately after training
  - Need to tune the model to further improve the performance
  - Select optimial values of hyperparameters
- This step is known as model selection
- A better approach: training set + validation set + test set
  - Validation set is used for model selection
  - Test set is for model evaluation
- Let us for now assume we know the hyperparameters



2. K-fold cross-validation

- Disadvantage of the holdout method: sensitive to partitioning
- To fix this, use K-fold cross-validation
  - Randomly split the training dataset into k folds
  - Of these, $k-1$ folds are used for training and one for testing
  - Repeat this procedure k times and then average across k folds
- Each sample will be part of train and test sets
- Lower-variance estimate of the model performance (than holdout)

K-fold cross-validation

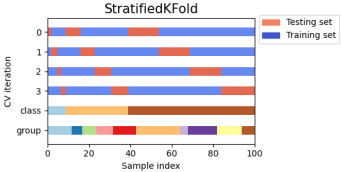


How do we pick the number of folds?

- The standard value is $k = 10$
- For small datasets, increase the number of folds
  - increases the amount of training data
- For larger datasets, we can decrease the number of folds
  - e.g. $k = 5$ is a reasonable choice

3. Stratified *k*-fold cross-validation

- *StratifiedKFold* is a variation of k-fold which returns stratified folds: Class proportions preserved in each fold
- So each fold is representative of the entire training set
- Better performance estimates for imbalanced data

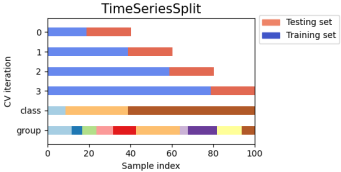


4. Leave-one-out cross-validation

- Set the number of folds equal to the number of training samples
- Only a single training sample used for testing during each iteration
- Recommended approach for very small datasets

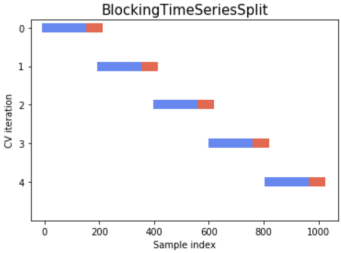
Time series make things even more complicated

- *TimeSeriesSplit* is a variation of k-fold which returns first k folds as train set and the $(k + 1)$ th fold as test set. Note that unlike standard cross-validation methods, successive training sets are supersets of those that come before them. Also, it adds all surplus data to the first training partition, which is always used to train the model.

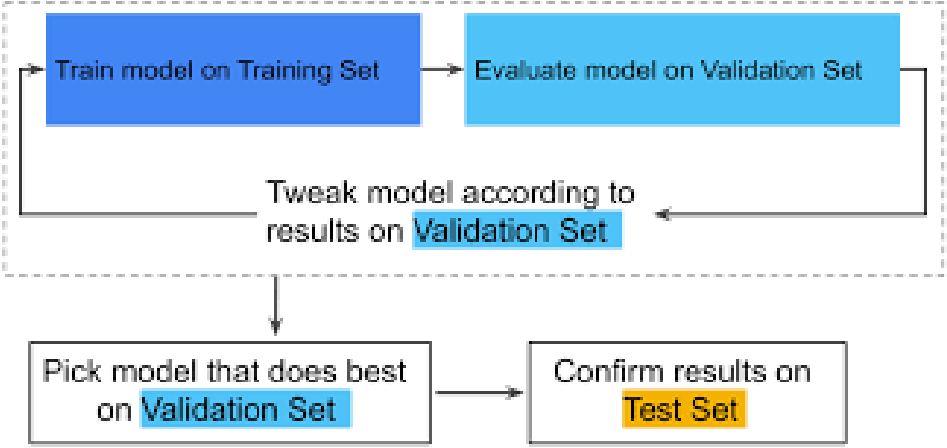


Time series make things even more complicated

- *TimeSeriesSplit* is a variation of k-fold which returns first k folds as train set and the $(k + 1)$ th fold as test set. Note that unlike standard cross-validation methods, successive training sets are supersets of those that come before them. Also, it adds all surplus data to the first training partition, which is always used to train the model.
- Alternatively you can do walk forward cross validation



How to find the right hyperparameters



How to find the right hyperparameters

- Many ML algorithms offer a number of hyperparameters but it is often unclear what the optimal set is
- Options
  1. Manual
  2. Grid search: a brute-force exhaustive search of the complete hyperparameter space
     - `GridSearchCV`
     - Obviously, this can be computationally very expensive but will find the global optimum
  3. Randomized search:
     - `RandomizedSearchCV`

Hyperparameters: Gradient Boosting Decision Trees

					xgboost			lightghm			catboost	
					Parameter Names			Parameter Names			Parameter Names	
Name	What	Allowed Range	Baseline Choice	Favorable Tuning Range	Original API	sklearn API	Default	Original API	sklearn API	Default	Original API & sklearn API	Default
1. Most important!												
Maximum Depth	Maximum depth of each trained tree.	[0, ∞]	5 or 6	[3, 12]	max_depth	max_depth	6	max_depth	max_depth	-1	depth	6
2. Tune at earlier phase.												
Row Sampling	Percentage of rows used per iteration frequency.	(0, 1]	0.8	[0.6, 1.0]	subsample	subsample	1.0	bagging_fraction	subsample	1.0	subsample	Depends
Column Sampling by Tree	Percentage of columns used per iteration.	(0, 1]	0.8	[0.6, 1.0]	colsample_bytree	colsample_bytree	1.0	feature_fraction	colsample_bytree	1.0	NULL	
Column Sampling by Level	Percentage of columns used per split selection.	(0, 1]	0.5	[0.6, 1.0]	colsample_bylevel	colsample_bylevel	1.0	NULL			rsm	1.0
Hessian Regularization	Prune by minimum hessian requirement.	[0, ∞]	1.0	[0.1, 10.0]	min_child_weight	min_child_weight	1.0	min_sum_hessian_in_leaf	min_child_weight	0.001	NULL	
3. Tune at intense tuning phase.												
Minimum Data per Leaf	Prune by minimum number of observations requirement.	[0, ∞]	Depends on data size. 10 or 20 for small data (#~100), hundreds/thousands for large data.		NULL			min_data_in_leaf	min_child_samples	20	min_data_in_leaf *	1
Maximum Leaves	Maximum leaves for each trained tree.	[1, ∞]	255	xgboost, catboost: 255/ (2^d) ~ 255* (2^d) lightgbm: Less than 2^(max_depth)	max_leaves	max_leaf_nodes	0	num_leaves	num_leaves	31	max_leaves *	31
4. Parameters to tune when other parameters are tuned enough.												
L1 Regularization	L1 Regularization for boosting.	[0, ∞]	Default Value		alpha	reg_alpha	0.0	lambda_l1	reg_alpha	0.0	NULL	
L2 Regularization	L2 Regularization for boosting.	[0, ∞]	Default Value		lambda	reg_lambda	1.0	lambda_l2	reg_lambda	0.0	l2_leaf_reg	3.0
Loss Regularization	Prune by minimum loss requirement.	[0, ∞]	Default Value		gamma	gamma	0.0	min_gain_to_split	min_split_gain	0.0	NULL	
5. Leave baseline at first, smaller at the time of final tuning.												
Learning rate	Multiplication performed on each boosting iteration.	(0, 1]	0.1	[0.01, 0.05] as phase advances.	eta	learning_rate	0.3	learning_rate	learning_rate	0.1	learning_rate	Auto or 0.03
6. Too many iterations can cause overfitting. Number of iterations can be large if early stopping is enabled, and we should do so :)												
Number of Iterations	Number of boosting iterations.	[1, ∞]	1000 or 10000 with early stopping		nrounds	n_estimators	NULL	num_iterations	n_estimators	100	iterations	1000
Early Stopping	Number of maximum iterations without improvements.	[0, ∞]	10 divided by 'learning rate'		early_stopping_rounds	early_stopping_rounds	NULL	early_stopping_round	early_stopping_rounds	0	early_stopping_rounds	False

* Some catboost parameters are only available in traing on GPU.

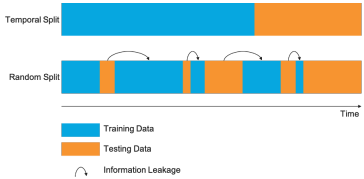
Outline

Information leakage

- Explicit: some features represent transformation of or a proxy for the target variable
  - Number of months behind in interest payments

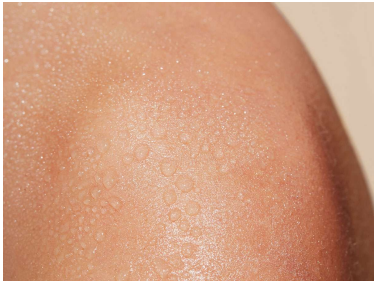
Information leakage

- Explicit: some features represent transformation of or a proxy for the target variable
  - Number of months behind in interest payments
- Implicit: training has info unavailable for the test observations
  - timeseries: if a feature is measured with a lag and/or is revised often (e.g. GDP, news), be careful to use the right value



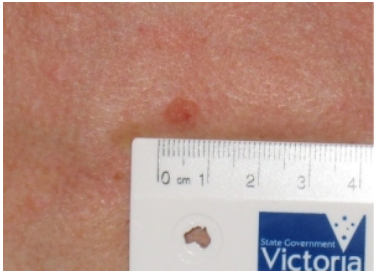
Use case: rulers cause skin cancer

- Skin cancer, the most common human malignancy, is primarily diagnosed visually, beginning with an initial clinical screening and followed potentially by more (invasive) tests
- Esteva et al. (2017, Nature), train a CNN using a dataset of 129,450 clinical images consisting of 2,032 different diseases. We test its performance against 21 board-certified dermatologists on biopsy-proven clinical images
- It would be great if this method could be rolled out to your smartphone ...



Use case: rulers cause skin cancer

- Skin cancer, the most common human malignancy, is primarily diagnosed visually, beginning with an initial clinical screening and followed potentially by more (invasive) tests
- Esteva et al. (2017, Nature), train a CNN using a dataset of 129,450 clinical images consisting of 2,032 different diseases. We test its performance against 21 board-certified dermatologists on biopsy-proven clinical images
- It would be great if this method could be rolled out to your smartphone ...
- ... but turns out that a ruler is bad for you

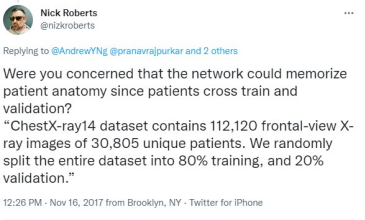


Use case: the random shuffle

- CheXNet: algorithm to detect pneumonia from chest X-rays at a level exceeding radiologists (**rajpurkarCheXNetRadiologistLevelPneumonia2017**)
- 121-layer convolutional neural network trained on largest chest X-ray dataset (ChestX-ray14)
- Radiologists annotate a test set (n=4), on which we compare the performance of CheXNet to that of radiologists
- CheXNet > average radiologist performance on the F1 metric
  - 100,000 frontal-view X-rays with 14 diseases
  - 30,000 patients
  - random train-test

Use case: the random shuffle

- CheXNet: algorithm to detect pneumonia from chest X-rays at a level exceeding radiologists (**rajpurkarCheXNetRadiologistLevelPneumonia2017**)
- 121-layer convolutional neural network trained on largest chest X-ray dataset (ChestX-ray14)
- Radiologists annotate a test set (n=4), on which we compare the performance of CheXNet to that of radiologists
- CheXNet > average radiologist performance on the F1 metric
  - 100,000 frontal-view X-rays with 14 diseases
  - 30,000 patients
  - random train-test



Summary

In this lecture we covered:

1. We looked at methods to reduce the complexity to models
   - Dimension reduction through PCA
   - Feature selection with RIDGE and LASSO
2. We also discussed ways of assessing how well a model performs
   - holdout
   - K-fold cross validation

References I

- Anscombe, F. (1973). Graphs in Statistical Analysis. American Statistician, 27, 17–21.
- Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks [Publisher: Nature Publishing Group]. Nature, 542(7639), 115–118.
- Hastie, T., Tibshirani, R., & Friedman, J. (2017). The Elements of Statistical Learning: Data Mining, Inference, and Prediction [arXiv: 1011.1669v3 ISSN: 03436993].