Building a Robot Judge: Data Science for Decision-Making

12. Encoders and Explanations

Q&A Page

https://bitly.com/BRJ_Padlet12

Encoders and Explanations

This lecture is about:

1. encoding high-dimensional datasets down to lower dimensions (dimensionality reduction)
2. explaining the predictions of classifiers and regressors

Encoders and Explanations

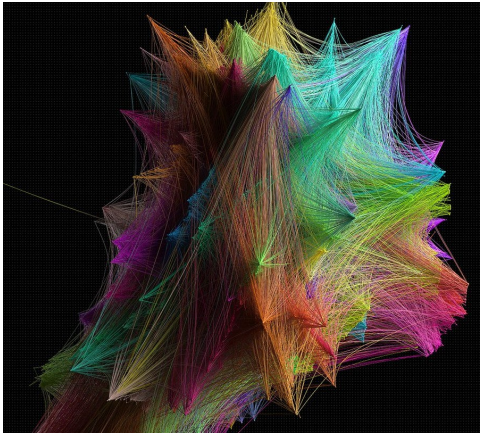
This lecture is about:

1. encoding high-dimensional datasets down to lower dimensions (dimensionality reduction)
2. explaining the predictions of classifiers and regressors

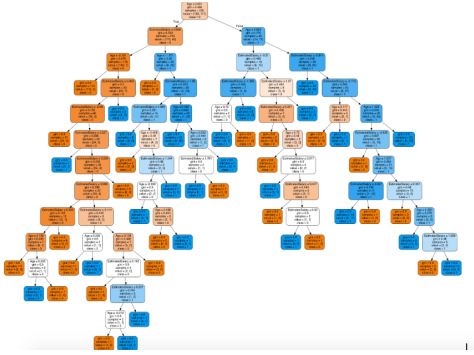
We will see that these are highly overlapping tasks. Interpretability/explainability is a major benefit of dimensionality reduction.

High-dimensional datasets and machine learning models are black boxes

High-Dimensional Datasets



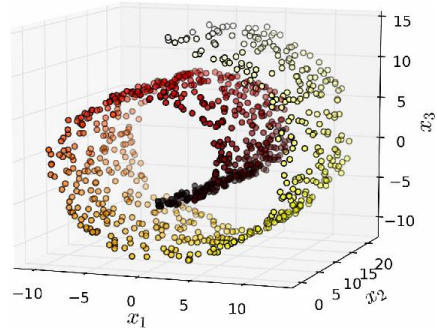
Big ML Models



Dimensionality Reduction (DR)

- Real-world datasets are not distributed uniformly across the feature space.
- They have a lower-dimensional latent structure – a **manifold** – that can be learned.

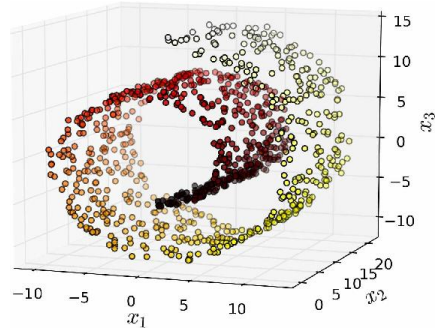
"The Swiss Roll"

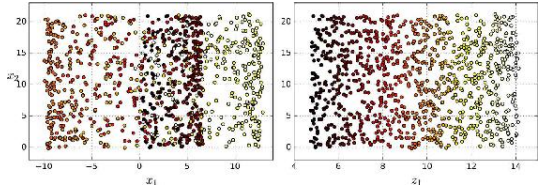


Dimensionality Reduction (DR)

- Real-world datasets are not distributed uniformly across the feature space.
- They have a lower-dimensional latent structure – a **manifold** – that can be learned.

"The Swiss Roll"





- DR makes data more interpretable – for example by projecting down to two dimensions for visualization.
- improves computational tractability.
- can improve model performance.

Explaining Model Predictions

(a) Husky classified as wolf

(b) Explanation

Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

- Machine learning models often make decisions for the wrong reasons.

← for example, classifying a dog image as a wolf because there is snow in the background (a correlated feature).

Explaining Model Predictions

(a) Husky classified as wolf

(b) Explanation

Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

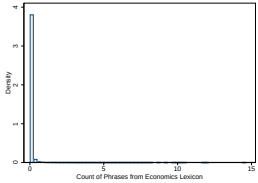
- Machine learning models often make decisions for the wrong reasons.

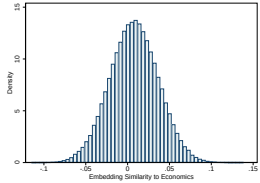
← for example, classifying a dog image as a wolf because there is snow in the background (a correlated feature).

- The presence of these problems, along with black box nature of ML models, is a major hurdle to making these technologies trustworthy enough to use to support high-stakes decisions, like those in courts.

A lot of what we do is interpreting/explaining

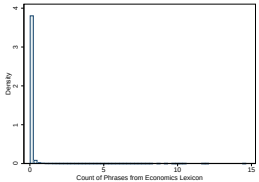
Summary statistics / plotting data:



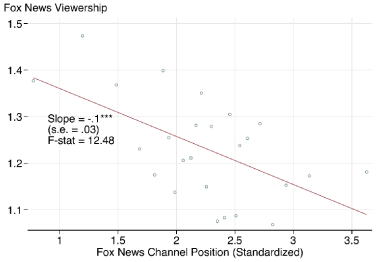


A lot of what we do is interpreting/explaining

Summary statistics / plotting data:

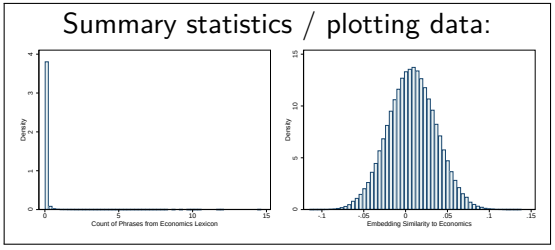


Scatter plots / regression plots:

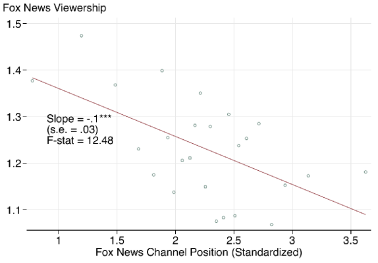


A lot of what we do is interpreting/explaining

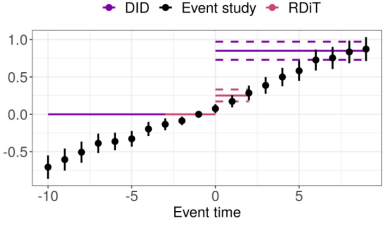
Summary statistics / plotting data:



Scatter plots / regression plots:

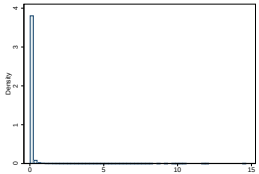


Event study plots "explain" difference-in-difference regression estimates:

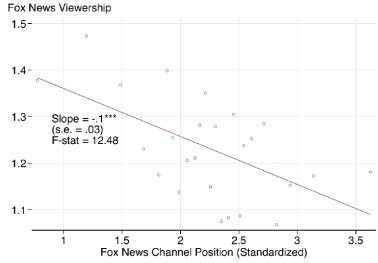


A lot of what we do is interpreting/explaining

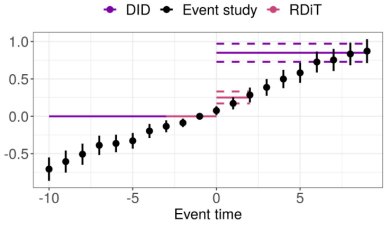
Summary statistics / plotting data:



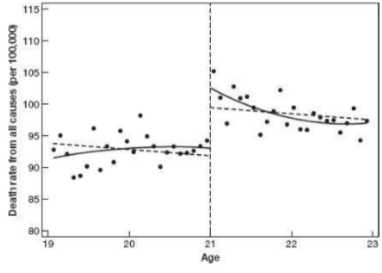
Scatter plots / regression plots:



Event study plots "explain" difference-in-difference regression estimates:



Regression discontinuity plots "explain" regression discontinuity estimates:



“Good” explanations

“Good” explanations

- **Selective**: explanations should be short
  - i.e. low-dimensional.

“Good” explanations

- **Selective**: explanations should be short
  - i.e. low-dimensional.
- **Social**: explanations should be targeted to the relevant audience.
  - e.g., different explanation for data scientists vs for lawyers.

"Good" explanations

- **Selective**: explanations should be short
  - i.e. low-dimensional.
- **Social**: explanations should be targeted to the relevant audience.
  - e.g., different explanation for data scientists vs for lawyers.
- **Contrastive**: explains not just why a certain prediction was made, but why it was made instead of other predictions.

Linear Models

- Linear models (e.g. linear regression, logistic regression) are (relatively) interpretable:
  - coefficients (and t-statistics) provide some idea of the important features.

Linear Models

- Linear models (e.g. linear regression, logistic regression) are (relatively) interpretable:
  - coefficients (and t-statistics) provide some idea of the important features.
- Caveats:
  - have to scale features for coefficients to be comparable
  - only small models (few predictors) are interpretable

Linear Models

- Linear models (e.g. linear regression, logistic regression) are (relatively) interpretable:
  - coefficients (and t-statistics) provide some idea of the important features.
- Caveats:
  - have to scale features for coefficients to be comparable
  - only small models (few predictors) are interpretable
  - excludes interactions $\rightarrow$ often have bad ML performance

Feature Importance / Selection

- For supervised learning tasks, doing feature selection to drop weak predictors has intuitive appeal.
  - Lasso models (with L1 penalty) do feature selection and produce sparse models.

Feature Importance / Selection

- For supervised learning tasks, doing feature selection to drop weak predictors has intuitive appeal.
  - Lasso models (with L1 penalty) do feature selection and produce sparse models.

Feature selection can be done as a pre-processing step:

```
from sklearn.feature_selection import SelectKBest, chi2
selector = SelectKBest(chi2, k=10)
X_filtered = selector.fit_transform(X,y)
```

- χ^2 (used for classification) is fast but features must be non-negative. With negative predictors, use `f_classif`. For regression, use `f_regression`.

Feature Importance / Selection

- For supervised learning tasks, doing feature selection to drop weak predictors has intuitive appeal.
  - Lasso models (with L1 penalty) do feature selection and produce sparse models.

Feature selection can be done as a pre-processing step:

```
from sklearn.feature_selection import SelectKBest, chi2
selector = SelectKBest(chi2, k=10)
X_filtered = selector.fit_transform(X,y)
```

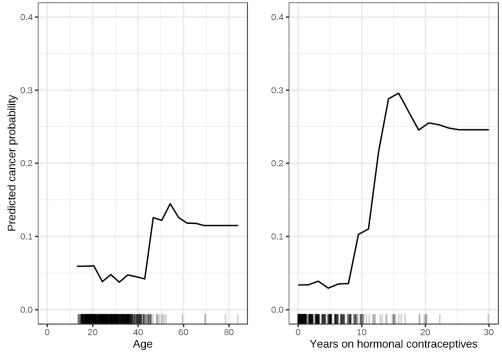
- χ^2 (used for classification) is fast but features must be non-negative. With negative predictors, use **`f_classif`**. For regression, use **`f_regression`.**
- `chi2, f_classif, and f_regression` measure covariance. Mutual information captures higher-order dependencies (**`mutual_info_classif`**, **`mutual_info_regression`**). Slower to compute.

Visualizing Marginal Effects on Predictions: Partial Dependence Plots

- Select feature(s) to analyze. Take averages of all other features, and then form predictions $\hat{y}$ along the range of the analyzed feature. Tells you how model uses the feature.

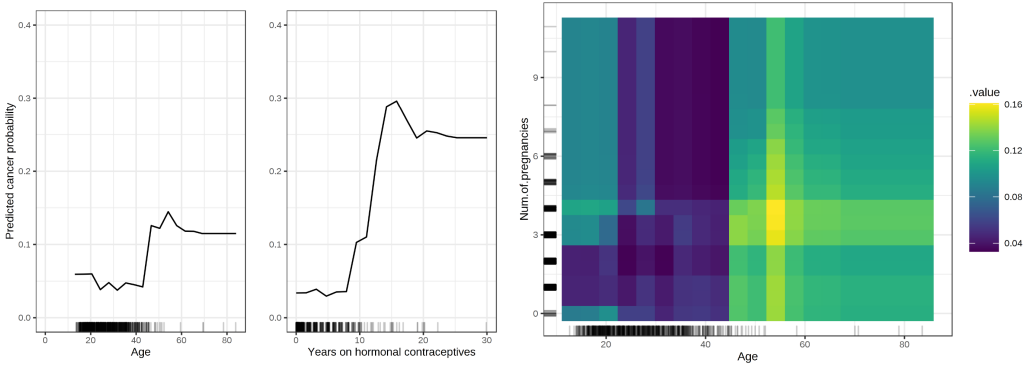
Visualizing Marginal Effects on Predictions: Partial Dependence Plots

- Select feature(s) to analyze. Take averages of all other features, and then form predictions $\hat{y}$ along the range of the analyzed feature. Tells you how model uses the feature.



Visualizing Marginal Effects on Predictions: Partial Dependence Plots

- Select feature(s) to analyze. Take averages of all other features, and then form predictions $\hat{y}$ along the range of the analyzed feature. Tells you how model uses the feature.



Permutation Feature Importance

- What features are most important for prediction?
  - `sklearn.feature_selection` metrics can be done before training a model, but they exclude any interaction effects between predictors.

Permutation Feature Importance

- What features are most important for prediction?
  - `sklearn.feature_selection` metrics can be done before training a model, but they exclude any interaction effects between predictors.

Solution: **Permutation feature importance** (Fisher, Rudin, and Dominici 2018):

Permutation Feature Importance

- What features are most important for prediction?
  - `sklearn.feature_selection` metrics can be done before training a model, but they exclude any interaction effects between predictors.

Solution: **Permutation feature importance** (Fisher, Rudin, and Dominici 2018):

1. Estimate any model, compute performance metric.
2. For each feature j:
   - generate new dataset where feature j is permuted (scrambled)
   - generate predictions and estimate new metric.
   - feature importance of j is decrease in performance.

Permutation Feature Importance

- What features are most important for prediction?
  - `sklearn.feature_selection` metrics can be done before training a model, but they exclude any interaction effects between predictors.

Solution: **Permutation feature importance** (Fisher, Rudin, and Dominici 2018):

1. Estimate any model, compute performance metric.
2. For each feature j:
   - generate new dataset where feature j is permuted (scrambled)
   - generate predictions and estimate new metric.
   - feature importance of j is decrease in performance.

Apply to trained `model` using test set:

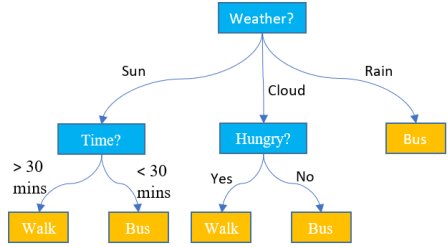
```
from eli5.sklearn import PermutationImportance
perm = PermutationImportance(model)
perm.fit(X_test, y_test)
eli5.show_weights(perm)
```

Out[20]:

Weight	Feature
0.4700 ± 0.0614	OverallQual
0.1439 ± 0.0156	GrLivArea
0.0499 ± 0.0034	2ndFlrSF
0.0363 ± 0.0091	TotalBsmtSF
0.0294 ± 0.0032	1stFlrSF
0.0271 ± 0.0102	BsmtFinSF1
0.0166 ± 0.0028	Fireplaces
0.0130 ± 0.0068	GarageArea
0.0130 ± 0.0044	YearBuilt
0.0115 ± 0.0071	LotArea
0.0105 ± 0.0048	GarageCars
0.0105 ± 0.0048	YearRemodAdd

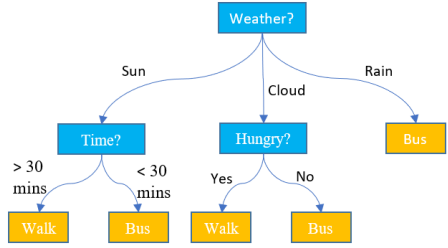
Trees and Tree Ensembles

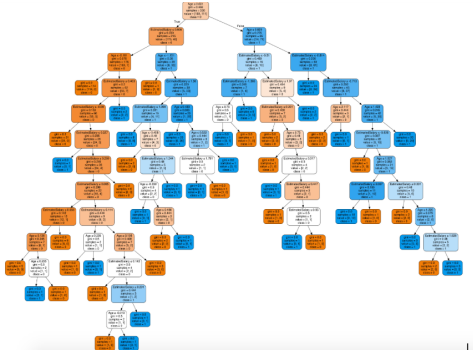
- Small decision trees have the advantage of being highly interpretable.



Trees and Tree Ensembles

- Small decision trees have the advantage of being highly interpretable.





- Larger trees and ensembles (e.g. XGBoost) lose this nice feature.
- Best-performing ML models are hard to interpret because they use lots of features and exploit non-linearities and interactions.

Interpreting Tree Ensembles

XGBoost's Feature Importance Metric is computed as:

1. In each tree, and at each decision node, compute **information gain** for feature j, which is equal to the **change in predicted probability** for one or the other outcome class.
2. Average across all nodes and all trees for each j.

This metric gives a ranking across predictors by their relative contributions.

```
from xgboost import plot_importance
plot_importance(xgb_reg, max_num_features=10)
```

Example: Most Important Budget Features for Corruption Prediction

(Ash, Galletta, Giommoni 2020)

Example: Most Important Budget Features for Corruption Prediction

(Ash, Galletta, Giommoni 2020)

Category	Macro Category	Weight	Category	Macro Category	Weight
Assets	Assets	330	Outstanding loan credit	Assets	69.4
Financial assets	Assets	182	Tax on industrialized products	Revenue	69
Population		142.6	**Property tax on land/buildings**	Revenue	68
Cash	Assets	116.4	Liquid assets	Assets	67.8
Spending in agriculture	Expenditure	94.8	Civil servant per diems	Expenditure	67.4
Property tax on rural land	Revenue	89.6	Spending for legislative procedure	Expenditure	65
Bank deposit	Assets	85.4	Taxes	Revenue	64.4
Motor vehicle property tax (from FG)	Revenue	72.8	Budget deficit		63
Transf. of ownership tax	Revenue	72	Non financial current asset	Assets	60.6
Spending in transportation	Expenditure	72	**Capital expenditure**	Expenditure	60

Important Features tend to show up in Audit Reports

We scraped all of the municipal audit reports from the agency web site.

- After converting the PDFs to text and some mild pre-processing, we counted the mentions of different budget features in the reports.

Important Features tend to show up in Audit Reports

We scraped all of the municipal audit reports from the agency web site.

- After converting the PDFs to text and some mild pre-processing, we counted the mentions of different budget features in the reports.

Produce dataset: {budget feature, audit report mentions, feature importance}

- Regress **audit report mentions** against **XGBoost feature importance.**

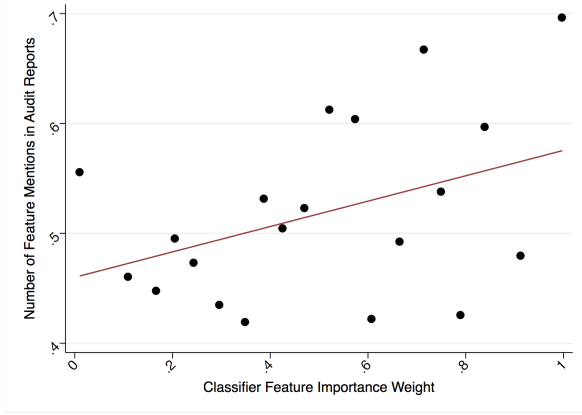
Important Features tend to show up in Audit Reports

We scraped all of the municipal audit reports from the agency web site.

- After converting the PDFs to text and some mild pre-processing, we counted the mentions of different budget features in the reports.

Produce dataset: {budget feature, audit report mentions, feature importance}

- Regress **audit report mentions** against **XGBoost feature importance.**

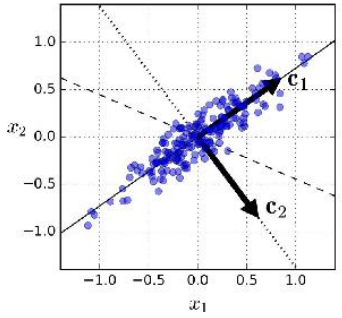


Notes: Binscatter for frequency that budget feature appears in the municipal audit reports (vertical axis) against binned feature importance weights for each feature (horizontal axis). Pearson's correlation is 0.17 (.24 for the log measures, rather than ranks). Slope coefficient is 0.112 with p=.03 (robust standard errors).

Activity

PCA (principal component analysis) / SVD (singular value decomposition)

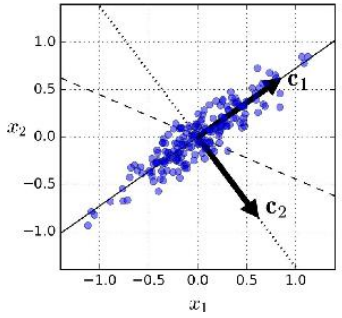
PCA (principal component analysis) / SVD (singular value decomposition)



- PCA computes the dimension in data explaining most variance.

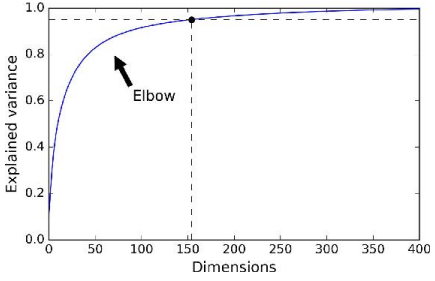
```
from sklearn.decomposition import PCA
pca = PCA(n_components=10)
X_pca = pca.fit_transform(X)
```

PCA (principal component analysis) / SVD (singular value decomposition)



- PCA computes the dimension in data explaining most variance.

```
from sklearn.decomposition import PCA
pca = PCA(n_components=10)
X_pca = pca.fit_transform(X)
```



- after the first component, subsequent components learn the (orthogonal) dimensions explaining most variance in dataset after projecting out first component.

PCA for Dimension Reduction

- Data can be reduced by projecting down to first principal component dimensions.
  - can be used as predictors instead of the original matrix.
  - distance metrics between observations are approximately preserved.
- For ML pre-processing, should learn components using only training set data.

PCA for Dimension Reduction

- Data can be reduced by projecting down to first principal component dimensions.
  - can be used as predictors instead of the original matrix.
  - distance metrics between observations are approximately preserved.
- For ML pre-processing, should learn components using only training set data.
- Can produce decompressed *"reconstruction"* back in original space (with error) using `pca.inverse_transform()` method.

PCA for Dimension Reduction

- Data can be reduced by projecting down to first principal component dimensions.
  - can be used as predictors instead of the original matrix.
  - distance metrics between observations are approximately preserved.
- For ML pre-processing, should learn components using only training set data.
- Can produce decompressed *"reconstruction"* back in original space (with error) using `pca.inverse_transform()` method.
- Standard PCA requires whole dataset in memory and is computationally costly.
  - use sklearn's `IncrementalPCA` to train in mini-batches.
  - `MiniBatchSparsePCA` learns regularized sparse components using an L1 penalty.

PCA for Dimension Reduction

- Data can be reduced by projecting down to first principal component dimensions.
  - can be used as predictors instead of the original matrix.
  - distance metrics between observations are approximately preserved.
- For ML pre-processing, should learn components using only training set data.
- Can produce decompressed *"reconstruction"* back in original space (with error) using `pca.inverse_transform()` method.
- Standard PCA requires whole dataset in memory and is computationally costly.
  - use sklearn's `IncrementalPCA` to train in mini-batches.
  - `MiniBatchSparsePCA` learns regularized sparse components using an L1 penalty.

Caveats:

- PCA might destroy (a lot of) predictive information in your dataset.
  - compromise: use feature selection to keep strong predictors, and take principal components of weak predictors.

PCA for Dimension Reduction

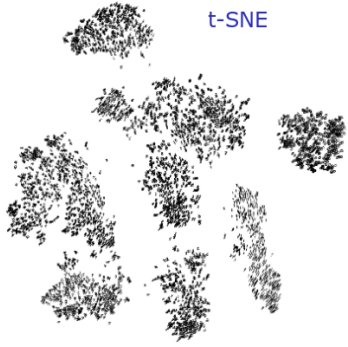
- Data can be reduced by projecting down to first principal component dimensions.
  - can be used as predictors instead of the original matrix.
  - distance metrics between observations are approximately preserved.
- For ML pre-processing, should learn components using only training set data.
- Can produce decompressed *"reconstruction"* back in original space (with error) using `pca.inverse_transform()` method.
- Standard PCA requires whole dataset in memory and is computationally costly.
  - use sklearn's `IncrementalPCA` to train in mini-batches.
  - `MiniBatchSparsePCA` learns regularized sparse components using an L1 penalty.

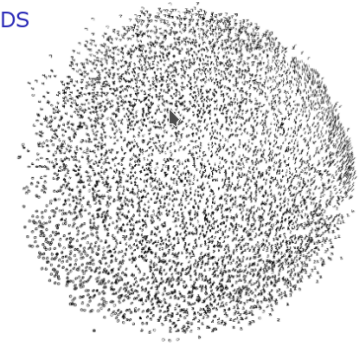
Caveats:

- PCA might destroy (a lot of) predictive information in your dataset.
  - compromise: use feature selection to keep strong predictors, and take principal components of weak predictors.
- dimensions are not interpretable.
  - For non-negative data (e.g. counts or frequencies), **Non-negative Matrix Factorization (NMF)** provides more interpretable factors than PCA.

t-SNE and MDS

t-SNE and MDS





From: L. Van der Maaten & G. Hinton, Visualizing Data using t-SNE, Journal of Machine Learning Research 9 (2008) 2579-2605

- **t-Distributed Stochastic Neighbor Embedding (t-SNE)** reduces dimensionality while trying to keep similar observations close and dissimilar observations apart.
  - Useful for visualizing clusters of observations in high-dimensional space
- **Multidimensional Scaling (MDS)** reduces dimensionality while trying to preserve the distances between the observations .

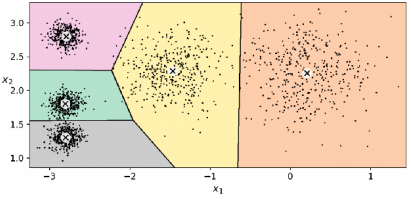
k-means clustering separates observations into k groups

k-means clustering separates observations into k groups

- Matrix of predictors treated as a Euclidean space (should standardize all columns)
- algorithm: initialize cluster centroids randomly, then shift around to minimize sum of within-cluster squared distance

k-means clustering separates observations into k groups

- Matrix of predictors treated as a Euclidean space (should standardize all columns)
- algorithm: initialize cluster centroids randomly, then shift around to minimize sum of within-cluster squared distance

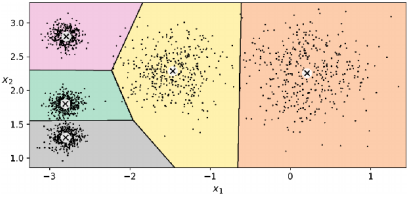


K-Means decision boundaries (Voronoi tessellation)

```
from sklearn.cluster import Kmeans
kmeans = KMeans(n_clusters=10)
kmeans.fit(X)
assigned_cluster = kmeans.labels_
```

k-means clustering separates observations into k groups

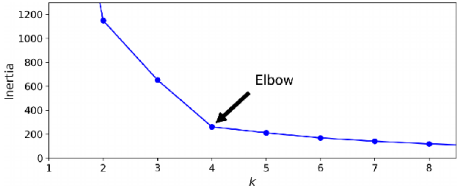
- Matrix of predictors treated as a Euclidean space (should standardize all columns)
- algorithm: initialize cluster centroids randomly, then shift around to minimize sum of within-cluster squared distance



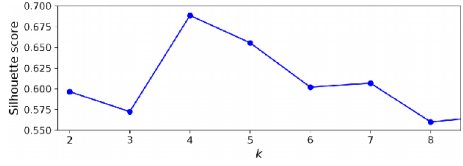
K-Means decision boundaries (Voronoi tessellation)

```
from sklearn.cluster import Kmeans
kmeans = KMeans(n_clusters=10)
kmeans.fit(X)
assigned_cluster = kmeans.labels_
```

k (number of clusters) is the only hyperparameter, can select using:



Selecting the number of clusters k using the "elbow rule"



Selecting the number of clusters k using the silhouette score

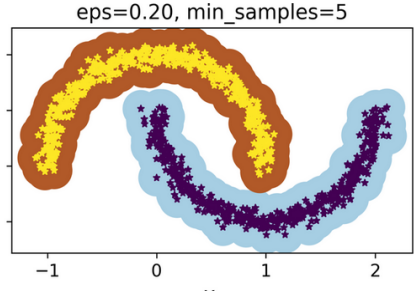
Other clustering algorithms

- "k-medoid" clustering use L1 distance rather than Euclidean distance; produces the "medoid" (median vector) for each cluster rathe than "centroid" (mean vector).
  - less sensitive to outliers, and medoid can be used as representative data point.

Other clustering algorithms

- "k-medoid" clustering use L1 distance rather than Euclidean distance; produces the "medoid" (median vector) for each cluster rathe than "centroid" (mean vector).
  - less sensitive to outliers, and medoid can be used as representative data point.

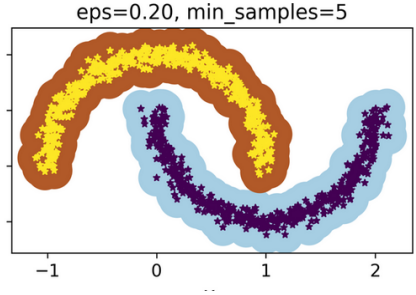
- DBSCAN defines clusters as continous regions of high density.
  - detects and excludes outliers automatically



Other clustering algorithms

- "k-medoid" clustering use L1 distance rather than Euclidean distance; produces the "medoid" (median vector) for each cluster rathe than "centroid" (mean vector).
  - less sensitive to outliers, and medoid can be used as representative data point.

- DBSCAN defines clusters as continous regions of high density.
  - detects and excludes outliers automatically



- Agglomerative (hierarchical) clustering makes nested clusters.
- Van Gansbeke et al (2020) add an entropy-reward parameter that works to distribute data points across clusters more equally.

Clusters Provide Prototypes

- Clustering can be used for description / explanation:
  - show selected variable values from centroid (even better, medoid) data points
  - medoids for large clusters are representative "prototypes" for the whole dataset (Molnar ch. 6.3).

Clusters Provide Prototypes

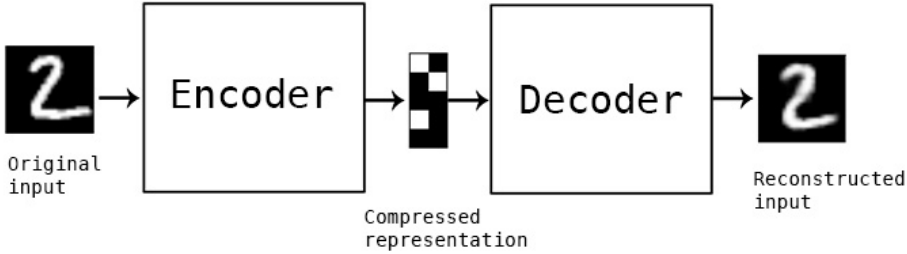
- Clustering can be used for description / explanation:
  - show selected variable values from centroid (even better, medoid) data points
  - medoids for large clusters are representative "prototypes" for the whole dataset (Molnar ch. 6.3).
- Conversely, data points that don't fit well into a cluster (far from any centroid, or dropped by dbscan) are outliers ("criticisms").

Which of the following number sequences do you find the easiest to memorize?

- 40, 27, 25, 36, 81, 57, 10, 73, 19, 68
- 50, 48, 46, 44, 42, 40, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, 18, 16, 14

Autoencoders: Optimal Compression Algorithms

- Autoencoders are neural networks that perform optimal domain-specific lossy compression:



  - Learn efficient encodings that can then be decoded back to a *reconstruction* – a (minimally) lossy representation of the original data.

Autoencoders can memorize complex data

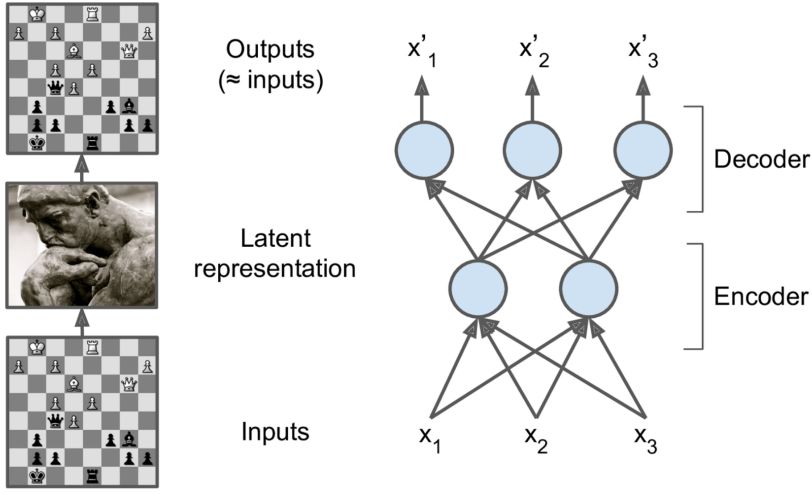


Figure 17-1. The chess memory experiment (left) and a simple autoencoder (right)

Stacked Autoencoders

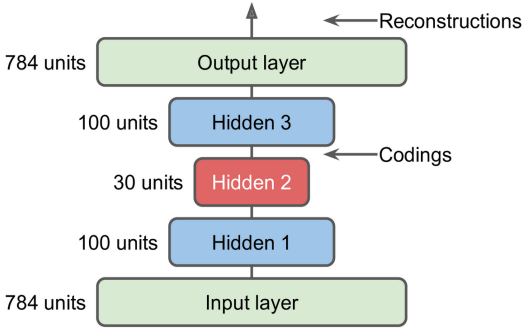


Figure 17-3. Stacked autoencoder

- Autoencoders work by stacking layers that gradually decrease in dimensionality to create the compressed representation (Z), and then gradually increase in dimensionality to try to reconstruct the input.

- for symmetic autoencoders, **tying weights** of the encoding and decoding segments will speed up training and tends to improve performance.

Reconstruction from encoded vector

Figure 17-4. Original images (top) and their reconstructions (bottom)

Autoencoders for Data Visualization

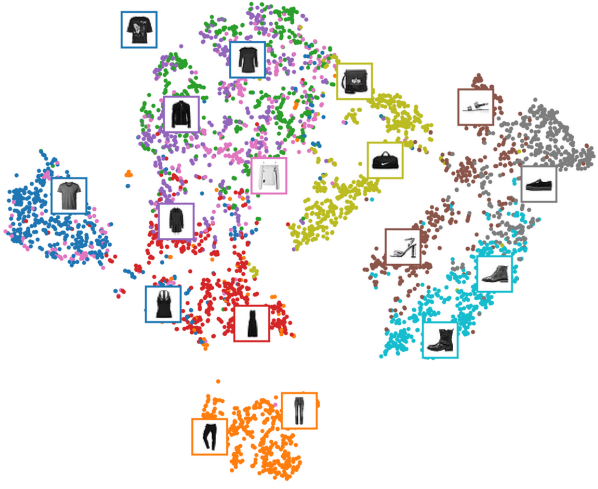
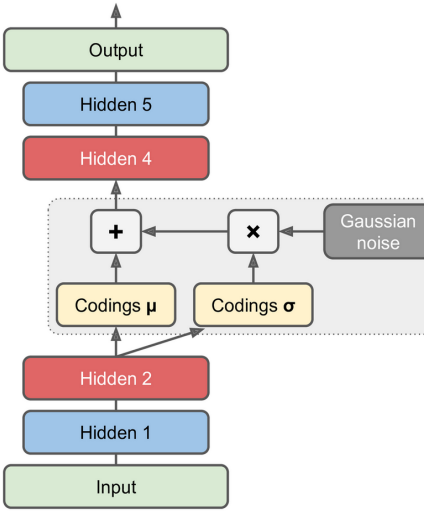


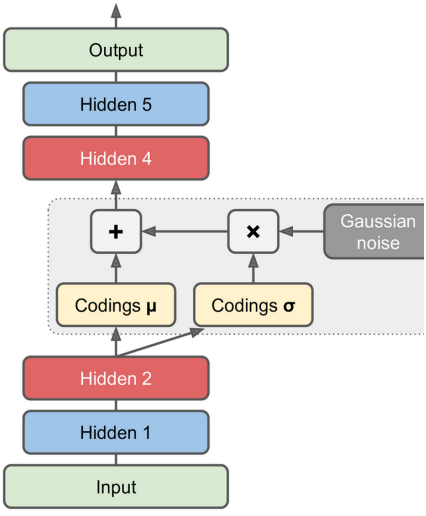
Figure 17-5. Fashion MNIST visualization using an autoencoder followed by t-SNE

- Decent baseline for visualizing the encodings:
  - use an autoencoder to compress your data to relatively low dimension (e.g. 32 dimensions)
  - then use t-SNE for mapping the compressed data to a 2D plane.

A Variational Autoencoder transforms low-dimensional encodings to the parameters of a gaussian (means μ and variances σ^2), then draws from the distribution to produce first layer for the decoder.

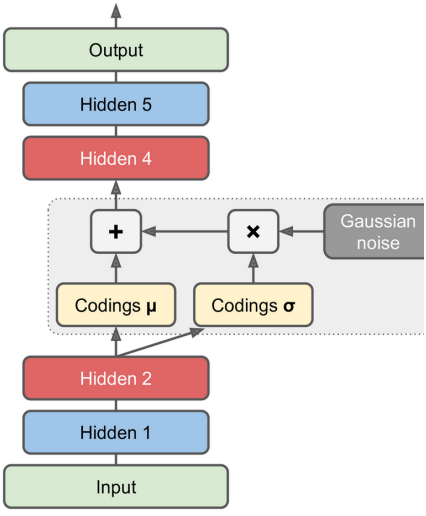


A Variational Autoencoder transforms low-dimensional encodings to the parameters of a gaussian (means μ and variances σ^2), then draws from the distribution to produce first layer for the decoder.



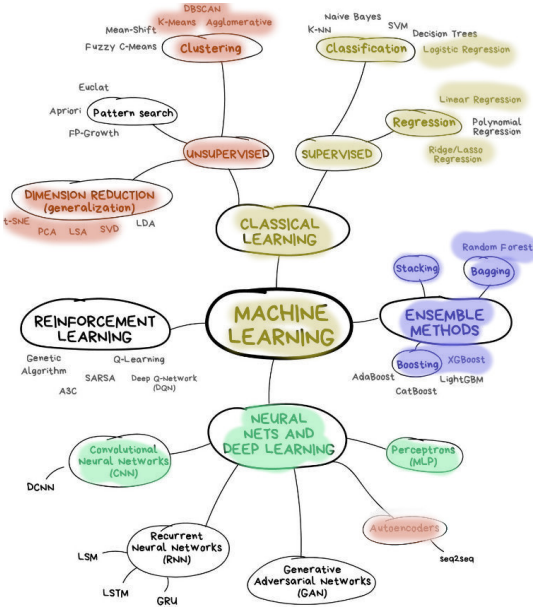
- Can then sample from the normal distribution (or just choose numbers) and generate reconstructions.

A Variational Autoencoder transforms low-dimensional encodings to the parameters of a gaussian (means μ and variances σ^2), then draws from the distribution to produce first layer for the decoder.



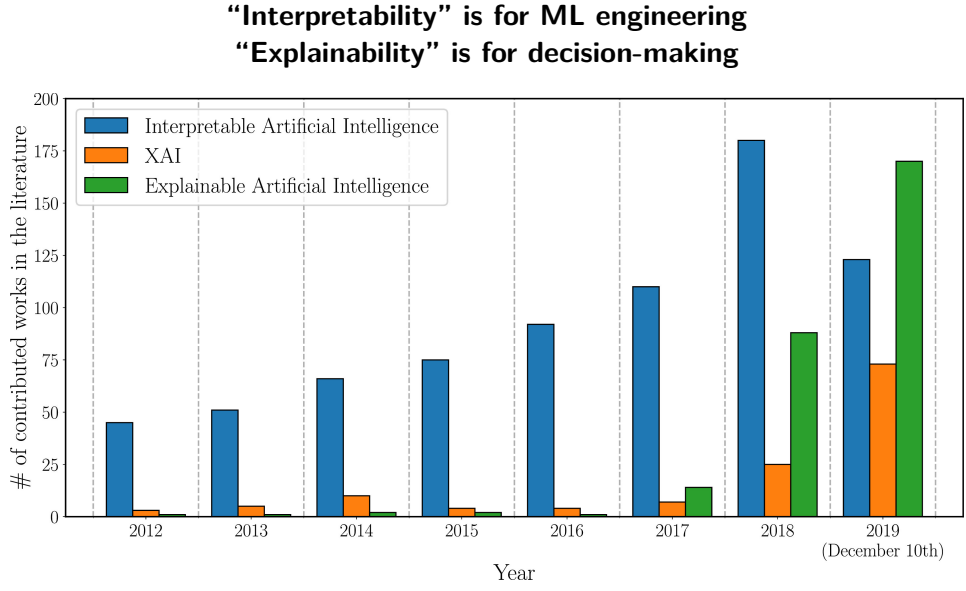
- Can then sample from the normal distribution (or just choose numbers) and generate reconstructions.
- VAE's do *semantic interpolation*: picking an encoding vector between two encodings will produce a reconstruction that is "between" the associated images

Recap: The ML Landscape



Activity

“Interpretable ML” and “Explainable ML” are distinct



Global Surrogate Model

Approximate a black box model with an interpretable model

Global Surrogate Model

Approximate a black box model with an interpretable model

1. Get predictions $\hat{y}$ of the black box model from the data X.

Global Surrogate Model

Approximate a black box model with an interpretable model

1. Get predictions $\hat{y}$ of the black box model from the data X.
2. Train an interpretable model (lasso, decision tree, etc) on X with $\hat{y}$ as the label.

Global Surrogate Model

Approximate a black box model with an interpretable model

1. Get predictions $\hat{y}$ of the black box model from the data X.
2. Train an interpretable model (lasso, decision tree, etc) on X with $\hat{y}$ as the label.
3. Validate that the surrogate model replicates the predictions of the black box model
   - e.g., compute R^2 between black box $\hat{y}$ and surrogate $\hat{\hat{y}}$

Local Feature Importance

Local Surrogate Model
(LIME = local interpretable model-agnostic explanations.)

Local Feature Importance

Local Surrogate Model
(LIME = local interpretable model-agnostic explanations.)

1. Select data point to explain

Local Feature Importance

Local Surrogate Model
(LIME = local interpretable model-agnostic explanations.)

1. Select data point to explain
2. Get black box predictions for data point and for a sample of randomly perturbed points in its neighborhood.

Local Feature Importance

Local Surrogate Model
(LIME = local interpretable model-agnostic explanations.)

1. Select data point to explain
2. Get black box predictions for data point and for a sample of randomly perturbed points in its neighborhood.
3. Train interpretable model on perturbed dataset.
   - e.g., lasso with high L1 penalty.

Local Feature Importance

Local Surrogate Model

(LIME = local interpretable model-agnostic explanations.)

1. Select data point to explain
2. Get black box predictions for data point and for a sample of randomly perturbed points in its neighborhood.
3. Train interpretable model on perturbed dataset.
   - e.g., lasso with high L1 penalty.

Shapley Values

Assigns importance to features by relative contribution to prediction (complicated formula based on solution concept in game theory)

Local Feature Importance

Local Surrogate Model

(LIME = local interpretable model-agnostic explanations.)

1. Select data point to explain
2. Get black box predictions for data point and for a sample of randomly perturbed points in its neighborhood.
3. Train interpretable model on perturbed dataset.
   - e.g., lasso with high L1 penalty.

Shapley Values

Assigns importance to features by relative contribution to prediction (complicated formula based on solution concept in game theory)

- (sometimes) better than LIME because accounts for interactions
- slower to compute
- default local importance metric on Google AI Platform

Interpreting Image Classifiers: Gradient Heat Maps

Example from "DeepConnection" paper predicting whether couples in images are happy or unhappy:

Interpreting Image Classifiers: Gradient Heat Maps

Example from "DeepConnection" paper predicting whether couples in images are happy or unhappy:

Counterfactual Explanations (Wachter et al 2017)

- Find the "closest" data point to the current one that would change the prediction.

Counterfactual Explanations (Wachter et al 2017)

- Find the "closest" data point to the current one that would change the prediction.
- For a given point X, ML prediction function $\hat{y}(\cdot)$, and alternative outcome y^*, find counterfactual X' that solves

$$\min_{X'} \lambda(\hat{y}(X') - y^*) + d(X, X')$$

where $d(\cdot)$ is a distance metric and $\lambda \geq 0$ calibrates the relative importance of label change and feature distance.

Counterfactual Explanations (Wachter et al 2017)

- Find the "closest" data point to the current one that would change the prediction.
- For a given point X, ML prediction function $\hat{y}(\cdot)$, and alternative outcome y^*, find counterfactual X' that solves

$$\min_{X'} \lambda(\hat{y}(X') - y^*) + d(X, X')$$

  where $d(\cdot)$ is a distance metric and $\lambda \geq 0$ calibrates the relative importance of label change and feature distance.
- To improve simplicity/interpretability, Wacher et al (2017) suggest:
  1. defining $d(\cdot)$ as the simple sum of distances between X and X' in each dimension – that is, the Manhattan (L1) distance $d(X, X') = \sum_{j=1}^{n_x} |x_j - x'_j|$.

Counterfactual Explanations (Wachter et al 2017)

- Find the "closest" data point to the current one that would change the prediction.
- For a given point X, ML prediction function $\hat{y}(\cdot)$, and alternative outcome y^*, find counterfactual X' that solves

$$\min_{X'} \lambda(\hat{y}(X') - y^*) + d(X, X')$$

  where $d(\cdot)$ is a distance metric and $\lambda \geq 0$ calibrates the relative importance of label change and feature distance.
- To improve simplicity/interpretability, Wacher et al (2017) suggest:
  1. defining $d(\cdot)$ as the simple sum of distances between X and X' in each dimension – that is, the Manhattan (L1) distance $d(X, X') = \sum_{j=1}^{n_x} |x_j - x'_j|$.
  2. Standardize all variables by their respective *mean absolute deviation*, so dimensions are comparable in L1 space.

Counterfactual Explanations (Wachter et al 2017)

- Find the "closest" data point to the current one that would change the prediction.
- For a given point X, ML prediction function $\hat{y}(\cdot)$, and alternative outcome y^*, find counterfactual X' that solves

$$\min_{X'} \lambda(\hat{y}(X') - y^*) + d(X, X')$$

  where $d(\cdot)$ is a distance metric and $\lambda \geq 0$ calibrates the relative importance of label change and feature distance.
- To improve simplicity/interpretability, Wacher et al (2017) suggest:
  1. defining $d(\cdot)$ as the simple sum of distances between X and X' in each dimension – that is, the Manhattan (L1) distance $d(X, X') = \sum_{j=1}^{n_x} |x_j - x'_j|$.
  2. Standardize all variables by their respective *mean absolute deviation*, so dimensions are comparable in L1 space.
- The mlxtend package makes this easy to do:

```
from mlxtend.evaluate import create_counterfactual
x_prime = create_counterfactual(x_reference=x_ref, # starting point
                                y_desired=1, # new class
                                model=clf, # trained model
                                X_dataset=X, #dataset
                                lammbda=1) #hyperparameter
```

Extensions

- Dandl et al (2020) improve on the Wachter et al objective in three ways:
  1. use Gower distance, rather than L1 distance, to allow for categorical features.

Extensions

- Dandl et al (2020) improve on the Wachter et al objective in three ways:
  1. use Gower distance, rather than L1 distance, to allow for categorical features.
  2. penalizes the number of predictors that are changed, to reward sparse changes.

Extensions

- Dandl et al (2020) improve on the Wachter et al objective in three ways:
  1. use Gower distance, rather than L1 distance, to allow for categorical features.
  2. penalizes the number of predictors that are changed, to reward sparse changes.
  3. rewards counterfactuals that are likely to be possible, measured by how close they are to at least one observed data point.

Activity