Building a Robot Judge: Data Science for the Law

6. Regression, Classification, and Regularization

Elliott Ash

OLS Regression Baseline

- Consider the linear model

$$Y_i = X_i'\theta + \epsilon_i$$

  where Y_i and all elements of X_i have been de-meaned and standardized to $s.d. = 1$.

- OLS assumptions:
  - X_i uncorrelated with ϵ_i
    - Let's just assume this for now; will come back later.
  - Columns of X_i are not highly collinear.
    - In the case of word/n-gram frequency data, this is not a good assumption.

Univariate OLS to Rank Predictive Features

- Consider the univariate regression

$$Y_i = \theta_w x_i^w + \epsilon_i$$

for each text feature w (e.g., relative word or n-gram frequency).

- Can be estimated with OLS.
  - Can add fixed effects:

$$Y_i = \alpha_i + \theta_w x_i^w + \epsilon_i$$

    where α_i is a vector of dummy variables (fixed effects) for categories, e.g. location, year.
  - Even better, can residualize Y and X on fixed effects before running any regressions.
    - That is, regress $Y_i = \alpha_i + \epsilon_i$ and $x_i^w = \alpha_i + \epsilon_i, \forall w$, take residuals $\tilde{Y}_i = Y_i - \hat{\alpha}_i$ and $\tilde{x}_i^w = x_i^w - \hat{\alpha}_i$
    - then regress

$$\tilde{Y}_i = \theta_w \tilde{x}_i^w + \epsilon_i$$

Estimating OLS

- OLS minimizes the mean squared error:

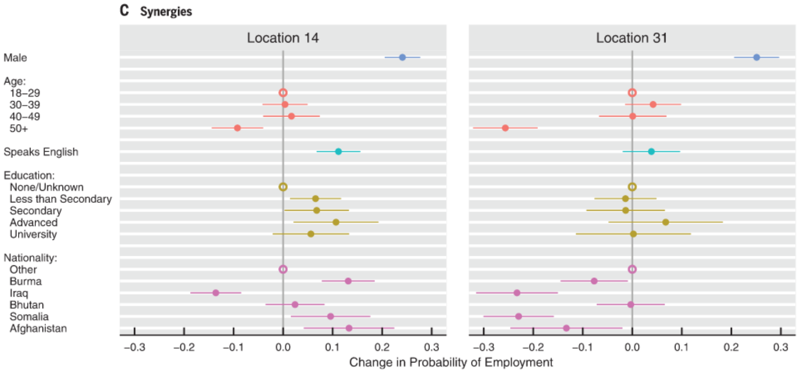
$$\min_{\hat{\theta}} \frac{1}{m} \sum_{i=1}^{m} (\boldsymbol{x}_i' \hat{\theta} - y_i)^2$$

  where $\boldsymbol{x}_i = (x_{i1}, x_{i2}, ..., x_{ip})$
- This has a closed form solution

$$\hat{\theta} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}$$

- Most machine learning models do not have a closed form solution.
  - objective must be minimized using some other optimization algorithm.

OLS Example: Refugee Resettlement and Employment Rates



OLS coefficient plots from Bansak et al, 2018 – effect of refugee characteristics on employment rates, separately by location.

Gradient Descent

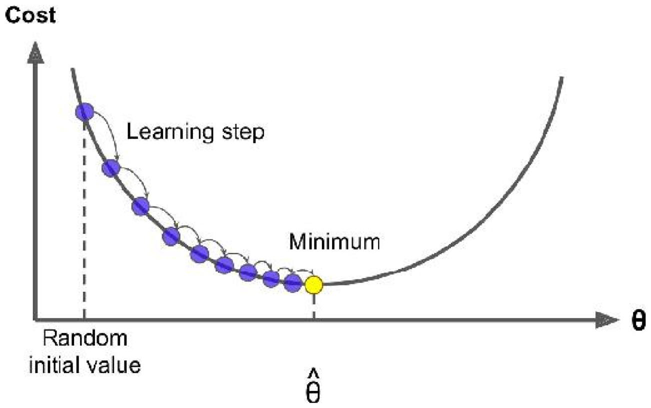
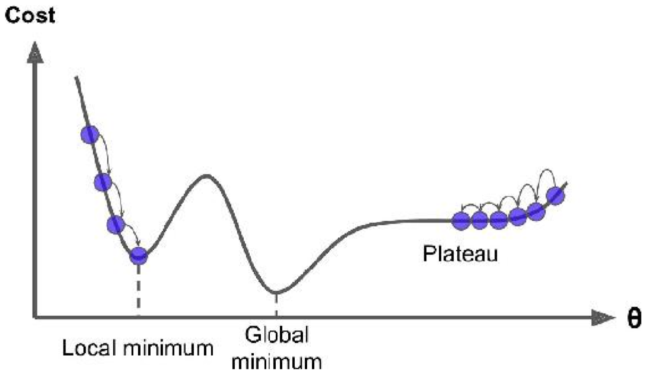


Figure 4-3. Gradient Descent

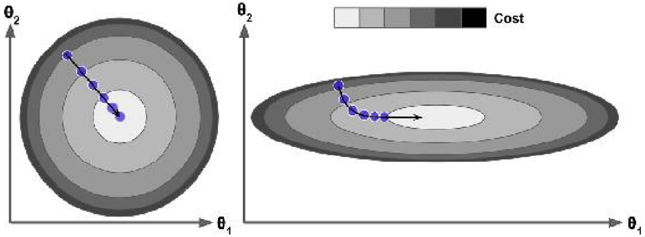
- Gradient descent measures the local gradient of the error function, and then steps in that direction.
  - Once the gradient equals zero, you have reached a minimum.

Gradient Descent Pitfalls



- Gradient descent will find a local minimum, not necessarily a global minimum. And it can get stuck on plateaus.

Gradient Descent and Scaling



- When using gradient descent, all features should be standardized to the same scale to speed up convergence.

Gradient Descent Implementation

- The partial derivative of MSE for feature j is

$$\frac{\partial \text{MSE}}{\partial \theta_j} = \frac{2}{m} \sum_{i=1}^{n} (\boldsymbol{x}_i' \hat{\theta} - y_i) x_{ij}$$

- The gradient is the vector of these partial derivatives is

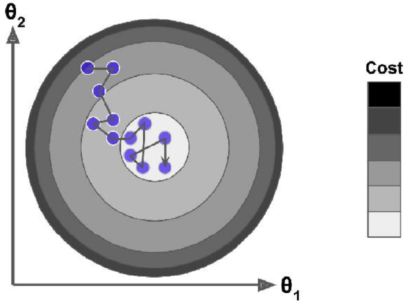
$$\nabla_\theta \text{MSE} = \begin{bmatrix} \frac{\partial \text{MSE}}{\partial \theta_0} \\ \frac{\partial \text{MSE}}{\partial \theta_0} \\ \vdots \\ \frac{\partial \text{MSE}}{\partial \theta_j} \end{bmatrix} = \frac{2}{m} \boldsymbol{X}' (\boldsymbol{X}' \theta - \boldsymbol{y})$$

- Gradient descent step:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \text{MSE}$$

Stochastic Gradient Descent

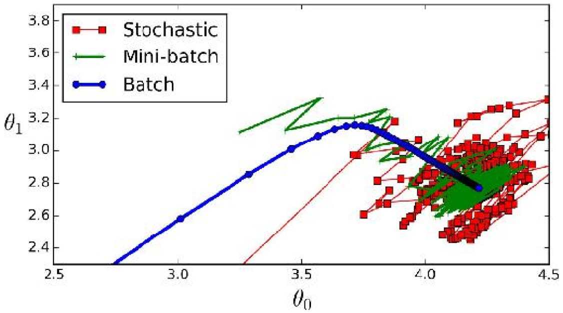
- SGD picks a random instance in the training set and computes the gradient only for that single instance.



- Much faster to train, but can still bounce around even after it is close to the minimum.

Mini-Batch Gradient Descent

- A compromise between gradient descent and stochastic gradient descent is mini-batch gradient descent, which selects a small number of rows (a "mini-batch") for gradient compute, rather than a single row.



Document Classification

- Say you have a set of cases where the defendent is held innocent or guilty.
  - You only know the outcome for a subset of documents which have been hand-coded.
  - Can we use the existing labels, and the text of all cases, to machine-code the labels for the unlabelled cases?
- sklearn has a number of classifiers for this purpose, e.g. LogisticRegression.

Confusion Matrix

- Use **cross_val_predict()** to obtain a “clean” prediction for each row, in the sense that it is trained on data outside that row’s fold.
- A confusion matrix is a nice way to visualize classifier performance:

		Predicted Class	
		Negative	Positive
True Class	Negative	# True Negatives	# False Positives
	Positive	# False Negatives	# True Positives

- The values in the table give counts in the evaluation set.

Precision and Recall

		Predicted Class	
		Negative	Positive
True Class	Negative	# True Negatives	# False Positives
	Positive	# False Negatives	# True Positives

- Two alternative metrics used to understand classifers:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- **Precision decreases with false positives.**
- **Recall decreases with false negatives.**

F1 Score

- The F_1 score provides a single combined metric – it is the harmonic mean of precision and recall:

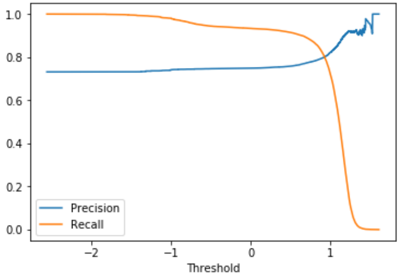
$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{Total Positives}}{\text{Total Positives} + \frac{1}{2}(\text{False Negatives} + \text{False Positives})}$$

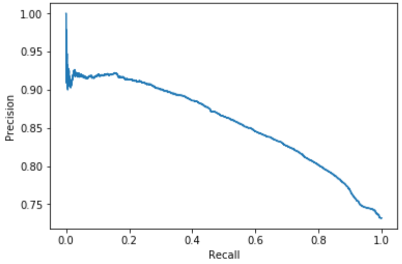
- What about True Negatives?

The Precision/Recall Tradeoff

- In general, one can tweak a classifier to increase precision at the cost of reducing recall, and vice versa.
  - The F1 score values them symmetrically.
- One can imagine contexts where they should be valued asymmetrically:
  - in the case of deciding "guilty" in court, you might prefer a model that lets many actual-guilty go free (high false negatives $\leftrightarrow$ low recall) but has very few actual-innocent put in jail (low false positives $\leftrightarrow$ high precision).
  - in the case of detecting bombs during flight screening, you might prefer a model that has many false alarms (low precision) to minimize the number of misses (high recall).

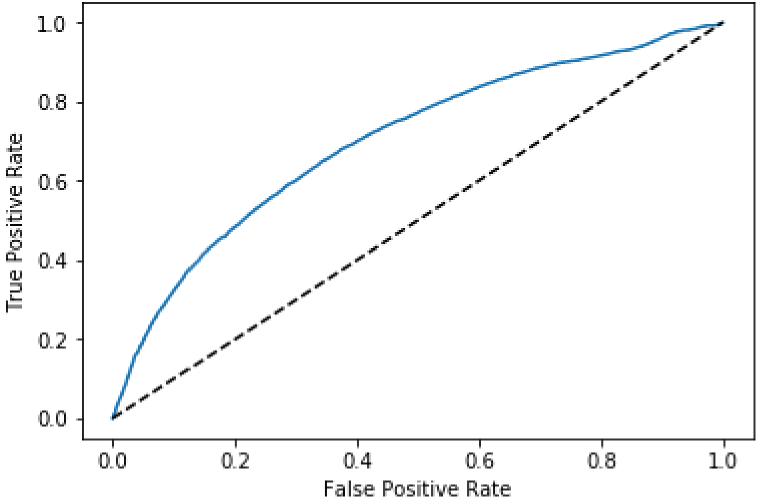
The Precision/Recall Tradeoff





ROC Curve and AUC

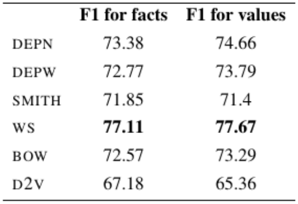
- Plots true positive rate (recall) against the false positive rate (FP / (FP + TN)):



Application: Facts vs. Law

- A crucial distinction in law practice, and in applying caselaw, is to distinguish the law from the facts.
- In Cao, Ash, Chen (2018), we made a sample of fact and law sections in court cases and trained a model to predict them based on the text.
- Corpus:
  - all cases from courtlistener.com with an annotated "FACTS" section (extracted using regular expressions).
  - 23,497 sections, split into 1.3M paragraphs
  - 36% facts, 64% laws

Facts vs. Law: Supervised Learning Methods



	F1 for facts	**F1 for values**
DEPN	73.38	74.66
DEPW	72.77	73.79
SMITH	71.85	71.4
WS	**77.11**	**77.67**
BOW	72.57	73.29
D2V	67.18	65.36

Table 2: 5-fold cross validation.

DEPN: Document represented as a sequence of word-dependency pairs, fed to a MLP classifier with two 500-dim hidden layers.
DEPW: a variant of DEPN using Shulayeva et al.'s (2017) dependency features.
SMITH : Implementation of Smith's (2014) method: Bag of Words, but filter words that are statistically related to either fact statements or law statements. Fed to a Logistic Regression classifier.
WS: Laver et al.'s (2003) Wordscore algorithm (cf. Sarel and Demirtas, 2017). Bag of Words, but each word assigned a score based on relation to fact or law. Document is represented as a (re-scaled) score that sums up the pre- calculated scores of the words it contains, weighted by their frequencies.
BOW : Bag of words representation of documents, fed to a Logistic Regression classifier.
D2V: 500-dim Doc2Vec vectors (Le and Mikolov, 2014), fed to MLP.

The Problem of Overfitting

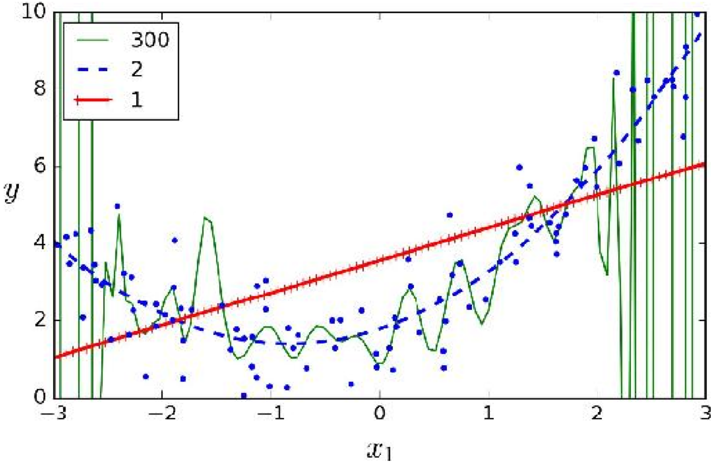


Figure 4-14. High-degree Polynomial Regression

Bias-Variance Tradeoff

- *Bias*
  - Error due to wrong assumptions, such as assuming data is linear when it is quadratic.
  - Will underfit the training data
- *Variance*
  - Error due to excess sensitivity to small variations in the training data.
  - A model with high variance is likely to overfit the training data.
- *Irreducible error*
  - Error due to noise in the data.

Training Notes

- In general, increasing a model's complexity will increase variance and reduce bias.
- If the model is underfitting:
  - adding more training data will not help.
  - use a more complex model
- If the model is overfitting:
  - adding more training data may help
  - or use regularization

Lasso, Ridge, and Elastic Net

- Lasso and ridge regression are tools for dealing with large feature sets where:
  - models have multicollinearity that causes bias
  - models tend to overfit
  - models are computationally costly to fit
- These algorithms work by constraining estimated parameter sizes.

Lasso Regresison

- The Lasso cost function is

$$J(\theta) = \text{MSE}(\theta) + \alpha_1 \sum_{i=1}^{n} |\theta_i|$$

  - i indexes over n features
  - α_1 is a hyperparameter setting the strength of the L1 penalty
- Lasso automatically performs feature selection and outputs a sparse model.

Ridge Regression

- The Ridge cost function is

$$J(\theta) = \text{MSE}(\theta) + \alpha_2 \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

  - i indexes over n features
  - α_2 is a hyperparameter setting the strength of the L2 penalty

- It turns out that the Ridge estimator, like OLS, has a closed-form solution:

$$\hat{\theta}_{\text{Ridge}} = (X'X + \alpha_2 \boldsymbol{I}_n)^{-1} X' \boldsymbol{y}$$

  where $\boldsymbol{I}_n$ is the identity matrix.

  - But it can also be solved by (stochastic) gradient descent.

Elastic Net

- Elastic Net uses both L1 and L2 penalties:

$$J(\theta) = \text{MSE}(\theta) + \alpha_1 \sum_{i=1}^{n} |\theta_i| + \alpha_2 \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

  - in general, elastic net is preferred to lasso, which can behave erratically when the number of features is greater than the number of rows, or when some features are highly collinear.
- For elastic net, as with Lasso and Ridge, the hyperparameters can be selected by cross-validation.

Scaling while maintaining sparsity

- Regularization penalties are designed to work with scaled data.
  - An important feature of text data is sparsity, which is lost when taking out the mean:

$$\tilde{x}_i = \frac{x_i - \bar{\boldsymbol{x}}}{\mathrm{SD}[\boldsymbol{x}]}$$

- Solution:

$$\tilde{x}_i = \frac{x_i}{\mathrm{SD}[\boldsymbol{x}]}$$

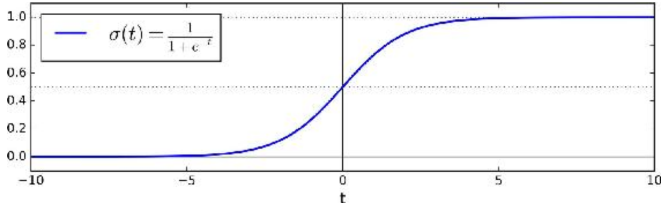
Logistic Regression is for Classification

- Like OLS, logistic "regression" computes a weighted sum of the input features to predict the output.
  - But rather than output the sum directly, it transforms the sum using the logistic function.

$$\hat{p} = \Pr(Y_i = 1) = \sigma(\theta' \boldsymbol{x})$$

where $\sigma(\cdot)$ is the signmoid function

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$



- For the binary version of the classifier, the prediction $\hat{Y} \in \{0, 1\}$ is determined by whether $\hat{p} \gtrless .5$.

Logistic Regression Cost Function

- The cost function to minimize is

$$J(\theta) = \underbrace{-\frac{1}{m}}_{\text{negative}} \sum_{i=1}^{m} [\underbrace{y_i}_{y_i=1} \underbrace{\log(\hat{p}_i)}_{\text{log prob} y_i=1} + \underbrace{(1-y_i)}_{y_i=0} \underbrace{\log(1-\hat{p}_i)}_{\text{log prob} y_i=0}]$$

  - this does not have a closed form solution
  - but it is convex, so gradient descent will find the global minimum.

- Just like linear models, logistic can be regulared with L1 or L2 penalties, e.g.:

$$J_2(\theta) = \mathrm{J}(\theta) + \alpha_2 \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

Multi-Class Models

- Many interesting machine learning problems involve multiple un-ordered categories:
  - categorizing a case by area of law
  - predicting the political party of a speaker in a proportional representation system

Multi-Class Confusion Matrix

		Predicted Class		
		Class A	Class B	Class C
True Class	Class A	Correct A	A, classed as B	A, classed as C
	Class B	B, classed as A	Correct B	B, classed as C
	Class C	C, classed as A	C, classed as B	Correct C

- More generally, can have a confusion matrix M with items M_{ij} (row i, column j).

Multi-Class Performance Metrics

Confusion matrix M with items M_{ij} (row i, column j).

$$\text{Precision for } k = \frac{\text{True Positives for } k}{\text{True Positives for k} + \text{False Positives for } k} = \frac{M_{kk}}{\sum_j M_{kj}}$$

$$\text{Recall for } k = \frac{\text{True Positives for } k}{\text{True Positives for } k + \text{False Negatives for } k} = \frac{M_{kk}}{\sum_i M_{ik}}$$

$$F_1(k) = 2 \times \frac{\text{precision}(k) \times \text{recall}(k)}{\text{precision}(k) + \text{recall}(k)}$$

Metrics for whole model

- Macro-averaging:
  - average of the per-class precision, recall, and F1, e.g.

$$F_1 = \frac{1}{n}\sum_{k=1}^{n} F_1(k)$$

  - treats all classes equally
- Micro-averaging:
  - Compute model-level sums for true positives, false positives, and false negatives; compute precision/recall from model sums.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}, \text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

  - favors bigger classes
- "Weighted": same as macro averaging, but classes are weighted by number of true instances in data.

Multinomial Logistic Regression

- Logistic can be generalized to multiple classes.
  - When given an instance $\boldsymbol{x}_i$, multinomial logistic computes a score $s_k(\boldsymbol{x}_i)$ for each class k,

$$s_k(\boldsymbol{x}_i) = \theta_k' \boldsymbol{x}_i$$

  - If there are n features and K output classes, there is a $K \times n$ parameter matrix Θ, where the parameters for each class are stored as rows.
- Using the scores, probabilities for each class are computed using the softmax function

$$\hat{p}_k(\boldsymbol{x}_i) = \Pr(Y_i = k) = \frac{\exp(s_k(\boldsymbol{x}_i))}{\sum_{j=1}^{K} \exp(s_j(\boldsymbol{x}_i))} = \frac{e^{\theta_k \boldsymbol{x}_i}}{\sum_{j=1}^{K} e^{\theta_j \boldsymbol{x}_i}}$$

  - And the prediction $Y_i \in \{1, ..., K\}$ is determined by the highest-probability category.

Multinomial Logistic Cost Function

- The binary cost function generalizes to the cross entropy

$$J(\theta) = \underbrace{-\frac{1}{m}}_{\text{negative}} \sum_{i=1}^{m} \sum_{k=1}^{K} \underbrace{\mathbf{1}[y_i = k]}_{y_i=k} \underbrace{\log(\hat{p}_k(\boldsymbol{x}_i))}_{\text{log prob} y_i=k}$$

  - again, this is convex, so gradient descent will find the global minimum.

Application: Ash, Morelli, Osnabrugge 2018

- Use multinomial logistic regression to classify policy topics from plain text.

Comparative Manifesto Project Corpus

- 44,020 annotated English-language political statements
  - hundreds of political party platforms from English-speaking countries.
- Each statement gets a CMP code, e.g. "decentralization", "education"
  - 45 topics
    - some topics are somewhat esoteric, such as "marxist analysis"
  - We normalized to 19 broader, more interpretable topics

Featurizing the Statements

- Each row includes a CMP code and a statement.
  - The statements are plain text.
- Standard featurization steps:
  - remove capitalization, punctuation, stopwords
  - construct n-grams up to length 3
  - remove n-grams appearing in less than 10 statements or more than 40 percent of statements
    - $M = 7,646$ features
  - compute tf-idf-weighted n-gram frequencies

Regularized Logit Model

- N rows, M text features, K policy topics
- Probability model:

$$P(Y_i = c) = \frac{e^{\beta_c X_i}}{\sum_{k=1}^{K} e^{\beta_k X_i}},$$

  where $c \in 1,...,K$ are the topic labels and β is an $M \times K$ matrix of parameters.
- Cost function:

$$J(\beta) = -\frac{1}{N}\left[\sum_{i=1}^{N}\sum_{k=1}^{K} \mathbf{1}\{y^i = k\} \log \frac{e^{\beta_k X_i}}{\sum_{l=1}^{K} e^{\beta_l X_i}}\right] + \gamma \sum_{j=1}^{M}\sum_{k=1}^{K} \beta_{jk}^2$$

- γ = strength of L2 penalty
  - $\gamma^* = 1/2$ selected by 3-fold cross-validation grid search.

Prediction Model

- Given a chunk of text, the logistic model computes a probability distribution over policy topics.
  - harnesses expert knowledge about political topics from Manifesto Project
- Validation of accuracy: predict the CMP code in a held-out sample of manifesto corpus statements
  - In-sample accuracy = 71%, Out-of-sample accuracy = 53%
    - quite good given there are 19 policy areas – choosing randomly would be correct 5% of the time; choosing top category (other topic) would be correct 15% of the time.

Confusion Matrix

	Admin	Agric	Culture	Econ	Educ	Free-dom	Intl	Law / Order	Way of Life	Other	Party Politics	Quality of Life	Target Groups	Tech	Welfare	Total True
Administration	348	5	8	117	12	47	9	20	10	94	21	43	1	33	118	**886**
Agriculture	6	110	2	34	2	2	5	1	6	23	4	49	0	14	18	**276**
Culture	12	0	155	14	13	15	5	1	17	52	2	16	8	15	44	**369**
Economics	59	12	3	961	11	18	21	7	21	107	35	95	1	43	176	**1570**
Education	24	1	12	30	481	6	6	5	9	66	14	6	0	35	88	**783**
Freedom	51	0	10	24	0	240	29	14	23	69	28	3	6	2	59	**558**
Internationalism	15	3	2	41	2	16	453	12	17	75	14	21	4	8	44	**727**
Law and Order	18	1	0	19	10	13	24	361	9	83	12	2	1	14	58	**625**
Nat'l Way of Life	18	1	8	31	14	25	20	14	133	73	26	26	3	3	91	**486**
Other	55	18	15	128	61	34	52	64	35	1239	33	86	11	57	185	**2073**
Party Politics	19	3	3	51	10	18	17	7	22	90	181	25	1	5	59	**511**
Quality of Life	40	24	4	130	4	7	24	2	10	96	11	619	2	43	45	**1061**
Target Groups	16	1	7	13	13	7	8	10	13	35	3	1	57	7	71	**262**
Technology	28	7	6	63	29	3	6	4	8	73	7	52	0	397	48	**731**
Welfare	67	4	10	151	52	29	16	26	38	161	20	22	12	34	1454	**2096**
Total Predicted	**776**	**190**	**245**	**1807**	**714**	**480**	**695**	**548**	**371**	**2336**	**411**	**1066**	**107**	**710**	**2558**	**13014**

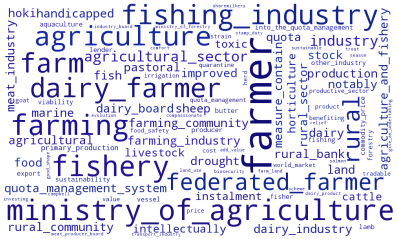
Interpreting the Topics

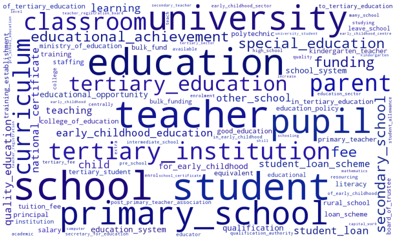
- In the analysis of New Zealand parliamentary speeches, we looked at the n-grams that were predictive of the topic classification.
    - Formally, for each topic k and each phrase m, regress

      $$\Pr(y_i = k) = \alpha + \delta_m x_i^m + \epsilon_i$$

      where $\Pr(y_i = k)$ is the predicted probability that speech i is topic k, and x_i^m is the tf-idf frequency of phrase m in speech i.
        - select phrases with highest positive t-statistic for δ_m

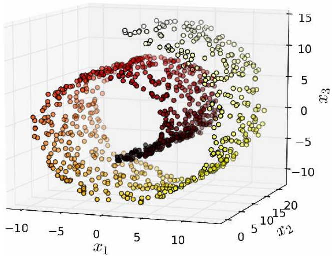
Agriculture and Education Topics

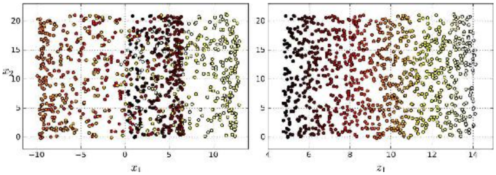




Dimensionality Reduction

- Especially in the case of text data, machine learning problems often involve thousands of features.
- Dimension reduction methods are needed:
  - not just for computational tractability, but also to help find a good solution.
  - also for data visualization – for example, to plot data in two dimensions.



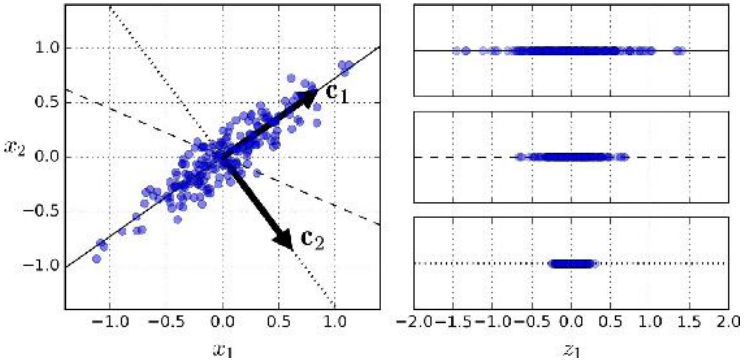


The dimension reduction process matters: projecting down to two dimensions directly (left panel) might not isolate the variation we are interested in (as done in the right panel, which unrolls the Swiss Roll)

Manifold Learning

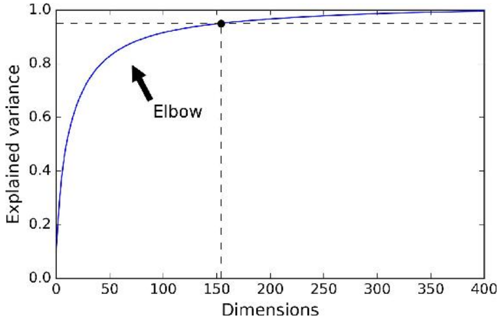
- The swiss roll is an example of a 2D manifold:
  - like many real-world data sets, the data are not uniformly distributed across the space.
  - can be modeled in a lower-dimensional subspace while retaining most of the information
- Dimension reduction methods in machine learning are motivated by this "manifold hypothesis."

PCA



- PCA (principal components analysis), a popular dimension reduction technique.
    - identifies the axis that accounts for the largest amount of variance in the training set.
    - finds a second axis, orthogonal to the first, that accounts for the largest amount of the remaining variance, and so on
- The unit vector defining the *i*th axis is called the *i*th principal component.

Choosing the number of dimensions



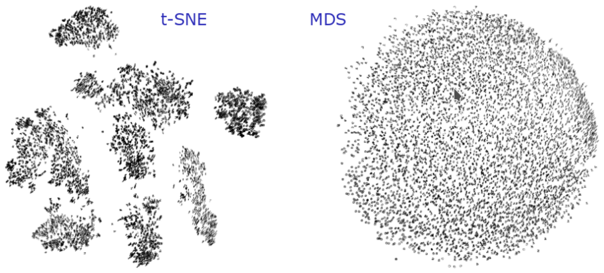
Incremental PCA and numpy `memmap`

- Standard PCA requires you to load the whole data set into memory.
- numpy memmap loads big arrays from disk as needed
- sklearn.IncrementalPCA splits the data into mini-batches and trains gradually.

Pros and Cons of PCA

- Advantages:
  - fast to compute
  - good performance on many tasks in practice
  - components are orthogonal by construction
- Disadvantages:
  - lose (potentially a lot of) predictive information from X
  - Coefficients are not easily interpretable.
- Compromise:
  - keep strong predictors in feature set; take principal components of weak predictors

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

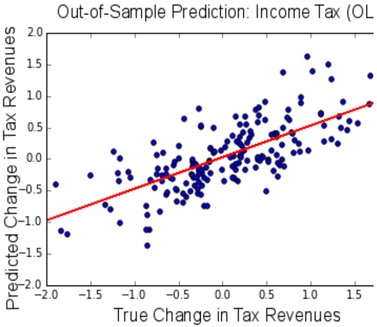
- **t-Distributed Stochastic Neighbor Embedding (t-SNE)** reduces dimensionality while trying to keep similar instances close and dissimilar instances apart.
  - Useful for visualizing clusters of instances in high-dimensional space
- **Multidimensional Scaling (MDS)** reduces dimensionality while trying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

Partial Least Squares

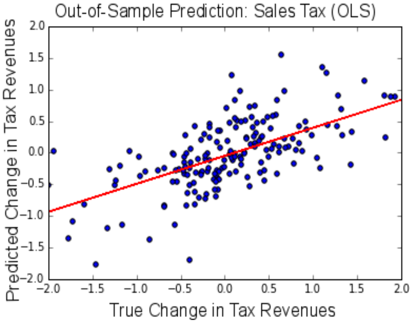
- Partial Least Squares is another technique to determine linear combinations of the predictive variables.
- Unlike PCA, the PLS technique works by successively extracting factors from both predictive and target variables such that covariance between the extracted factors is maximized.

Ash 2016: PLS predictions of tax revenue changes using tax code text

Income Tax

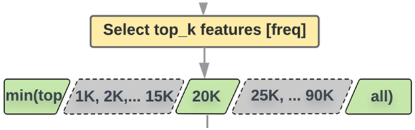


Sales Tax



Weak predictors filtered out; 80% training, 20% testing sample.

- Predicted change in revenue (vertical axis), plotted against true change in revenue (horizontal axis).
- Correlations between truth and prediction: 0.89 and 0.84.



- The Google text classification guide recommends a maximum 20,000 items in your feature set.

Feature selection using L1 Penalty

- A popular way to reduce dimensionality is to run lasso or elastic net, and exclude any predictors whose weights are regularized to zero.

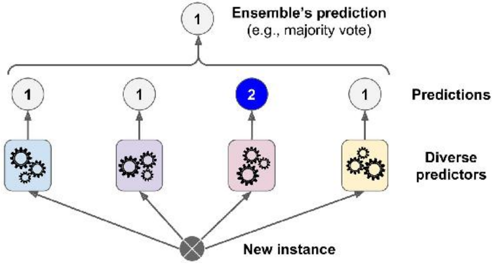
Feature selection using univariate comparisions

- χ^2 is a very fast feature selection routine for classification tasks
  - features must be non-negative
  - works on sparse matrices
- With negative predictors, use `f_classif`.
- For regression tasks, use `f_regression`.
- There are also `mutual_info_regression` and `mutual_info_classif`, which will recover non-linear relations between predictor and outcome
  - they are much slower, can run on a sample of the data.

Graph-based feature selection

- Graph-based feature selection works by clustering collinear features and then iteratively removing features.
  - much slower than univariate comparisons, but probably gives better performance in text classification tasks.
  - See Zhang and Hancock 2011.
  - There is an alpha implemention in python (not maintained) available at `https://github.com/danilkolikov/fsfc`.

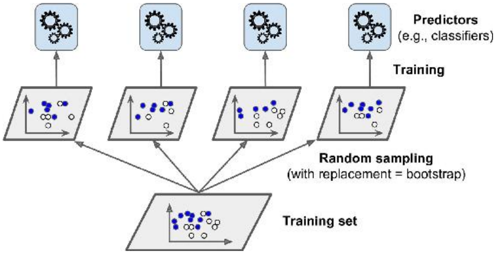
Voting Classifier



- voting classifiers generally out-perform the best classifier in the ensemble.
    - more diverse algorithms will make different types of errors, and improve your ensemble's robustness.

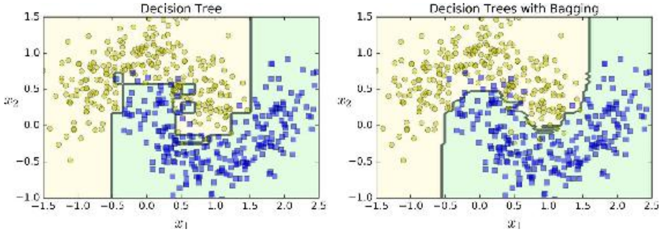
Bagging and Pasting

- Rather than use the same data on different classifiers, one can use different subsets of the data on the same classifier:



  - This is called **bagging** (bootsrap aggregating, when sampling with replacement) or **pasting** (when sampling without replacement).
  - can also use different subsets of features across subclassifiers.
- The ensemble predicts by aggregating the predictions:
  - for classification, use the most frequent prediction
  - for regression, use the average output

Bagging Benefits



- While the individual predictors have a higher bias than a predictor trained on all the data, aggregation reduces both bias and variance.
  - Generally, the ensemble has a similar bias but lower variance than a single predictor trained on all the data.
- Predictors can be trained in parallel using separate CPU cores.

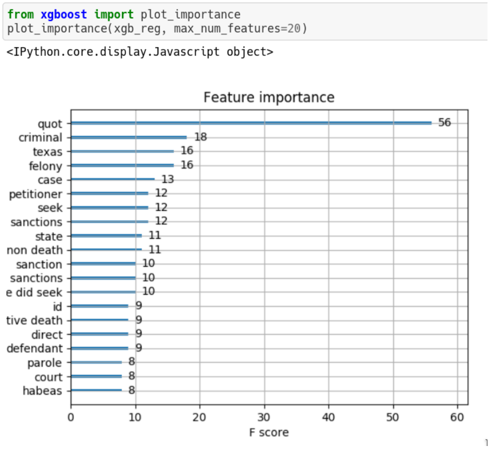
Random Forests

- Now you know how random forests work:
  - Random Forests are optimized ensembles of decision trees with bagging.
- Good prediction performance – due to out-of-sample validation being baked into the training process.

Gradient Boosted Machines and XGBoost

- Gradient boosting works by sequentially adding predictors to an ensemble – it fits the new predictor to the residual errors made by the previous predictor to gradually improve the model.
- As of 2014, gradient boosted machines are considered better than random forests – they represent the state-of-the-art besides deep neural nets.

Feature Importance



```
from xgboost import plot_importance
plot_importance(xgb_reg, max_num_features=20)
```

```
<IPython.core.display.Javascript object>
```

- Random forests and boosted trees provide a metric of feature importance that summarizes how well each feature contributes to predictive accuracy.

Ensemble Application: Jelveh, Kogut, and Naidu (2016)

- This paper looks at political language and ideology in the economics literature.
- They use data on campaign contributions to assign a subset of economists to Republican or Democrat.
- Then they train a classifier to predict party based on the text of written articles
  - They use an ensemble PLS model, that "votes" in the same way as random forests, but the constituent voters are PLS regressors, rather than decision trees.
  - They control for topic choices using JEL K codes and LDA topics.
  - The model predicts with 70% accuracy.

JKN 2016: Results

- There is significant ideological sorting across fields:
  - law and economics is the most-right wing field, labor economics is the most left-wing field
- Right-wing economists report a higher labor supply elasticity than left-wing economists
- The ideology of editors does not affect ideology of published articles.