CHARTA LEARNING · LESSON NOTES

Machine-Learning Methods

FRM Part I

1 A story: two analysts, one question

- Analyst one: theory, few variables, tests coefficients
- Analyst two: data first, many inputs, best forecast
- One explains why; the other predicts what next
- Machine learning is the second analyst's toolkit

IN THE LESSON

- Imagine two analysts asked about loan defaults.
- The first begins with a theory.
- Defaults rise when unemployment rises and when interest rates climb, so she writes a small model with those two variables, estimates the coefficients and tests whether they are significant.
- She can tell you why defaults move and what happens if unemployment jumps.
- The second analyst has no theory.
- He feeds hundreds of columns of data into a flexible algorithm and keeps whichever version predicts best on loans it has never seen.
- He may not be able to explain the forecast, but it may be more accurate.
- Both are useful, and they ask different questions.

2 Classical econometrics versus ML

Two philosophies, two ways of judging success

	Econometrics | ML
Starts from	Theory | Data
Variables	Modest number | Very many
Judged by	t-stats | Out-of-sample
Bias	Unbiased | Some accepted
Answers	Why, what if | What next

IN THE LESSON

- Here are the two approaches side by side.
- Econometrics starts from a theory, specifies a model such as a linear regression, and uses a modest number of variables.
- Machine learning starts from the data and lets flexible algorithms find the structure, often with far more features than a person could specify by hand.
- The judge differs too.
- Econometrics values interpretable coefficients, standard errors, significance and unbiased estimation.
- Machine learning is judged by how well it predicts on new data, and it will accept some bias to get better forecasts.
- Econometrics answers the questions why and what if.
- Machine learning answers the question what will happen.

3 In-sample fit proves little for ML

- In-sample fit can be bought by memorizing the data
- The judge is accuracy on data the model has not seen
- Tools: holdout samples and regularization penalties
- Cost: some bias and often less interpretability

IN THE LESSON

- There is one point to carry out of this section.
- A flexible algorithm can fit the data it was trained on almost perfectly, simply by memorizing it, so a good in-sample fit says very little.
- What matters is out-of-sample performance, meaning accuracy on data the model has never seen.
- That is why machine learning practice uses holdout samples, and why it uses regularization, a penalty that restrains the model from becoming too elaborate.
- It is also why interpretability is often given up for forecasting power.
- A model with a thousand tuned parameters may predict well and still be impossible to explain in a sentence.
- In the FRM, expect questions that reward the out-of-sample answer.

4 Three families: what does the model learn from?

- Supervised: labeled examples, predicts the label
- Regression: a number; classification: a category
- Unsupervised: no labels, finds structure in inputs
- Reinforcement: an agent learns from rewards
- Test: labels, only data, or rewards from actions?

IN THE LESSON

- Machine learning methods fall into three families, and the quickest way to tell them apart is to ask what the algorithm learns from.
- Supervised learning trains on labeled examples, inputs paired with the correct answer, and learns to predict the label for new inputs.
- Predicting a number is regression, and predicting a category, such as default or no default, is classification.
- Unsupervised learning has no labels.
- It hunts for structure in the inputs alone, for example by grouping similar observations or by compressing many features into a few.
- Reinforcement learning also has no labeled answers.
- An agent takes actions, earns rewards, and learns a policy that maximizes reward over time, where today's action changes tomorrow's situation.

5 Exercise: which family is it?

Ask: labels, nothing but the data, or rewards from its own actions?

Case	Family
Predict loan default	Supervised (classification)
Forecast inflation number	Supervised (regression)
Group bonds, no categories	Unsupervised (clustering)
Compress 50 ratios to 5	Unsupervised (PCA)
Trading agent, learns on P&L	Reinforcement learning

IN THE LESSON

- Let's practice with five cases, and for each, ask what the model learns from.
- First, predicting whether a loan will default, trained on past loans marked default or not.
- The labels are the correct answers, so this is supervised classification.
- Second, forecasting next month's inflation as a number from past data with known outcomes.
- Also supervised, and this time regression.
- Third, grouping bonds into similar sets when nobody has told the model any categories.
- No labels, so unsupervised.
- Fourth, squeezing fifty financial ratios into five summary variables.
- Unsupervised again, because compression looks at the inputs only.
- Fifth, a trading agent that learns from the profit its own actions produce.
- That is reinforcement learning.

6 Why we rescale features

- Revenue in millions swamps a leverage ratio near 2
- K-means, penalties and neural nets need fair scales
- Method one: min-max, values in 0 to 1
- Method two: standardize, mean 0 and std dev 1

IN THE LESSON

- Now to preparing data.
- Suppose one feature is a company's revenue in millions of dollars and another is its leverage ratio, a number around two.
- Many algorithms, including K means, regularized regression and neural networks, treat numbers at face value.
- Distances and penalties would be swamped by the revenue column simply because its numbers are bigger, not because it matters more.
- So features measured on different scales are rescaled before these algorithms see them.
- There are two standard methods.
- Min-max normalization squeezes every value into the range from zero to one.
- Standardization recenters each variable to a mean of zero and a standard deviation of one.
- Next we meet the formulas.

7 The two rescaling formulas

Min-max = (x − min) ÷ (max − min)
Standardized = (x − mean) ÷ std dev

Value to rescale	The raw value, x
Minimum	Smallest value; outliers set range
Maximum	Largest value; outliers set range
Mean	Training mean
Standard deviation	Training std dev; no fixed bounds

IN THE LESSON

- Min-max normalization takes a value, subtracts the minimum, and divides by the range, which is the maximum minus the minimum.
- Every value lands between zero and one.
- Because the extremes define the range, one outlier can squash everyone else into a narrow band.
- Standardization takes a value, subtracts the mean, and divides by the standard deviation.
- The result has a mean of zero and a standard deviation of one, and its values are not bounded.
- The most important rule is where the minimum, maximum, mean and standard deviation come from.
- They must be computed on the training set only.
- Calculating them on the whole sample lets information about the test data leak into the model.

8 Example 1: min-max normalization

Illustration: training values 2, 4, 4, 4, 5, 5, 7, 9. Min 2, max 9, range 7

Training value 9	(9 − 2) ÷ 7	**1.00**
Training value 5	(5 − 2) ÷ 7	**0.43**
Training value 2	(2 − 2) ÷ 7	**0.00**
New test value 11	(11 − 2) ÷ 7 (frozen)	**1.29**

IN THE LESSON

- Here is an illustration.
- A feature has eight training values: two, four, four, four, five, five, seven and nine.
- The minimum is two and the maximum is nine, so the range is seven.
- The training value nine becomes nine minus two, over seven, which is one.
- The training value five becomes three over seven, about point four three, and the minimum, two, becomes zero.
- Now a new value of eleven arrives in the test sample.
- Using the same training minimum and maximum, it becomes eleven minus two, over seven, which is one point two nine.
- That is above one, and it is correct.
- A test value can legitimately fall outside zero to one, because the parameters are frozen from training and never recomputed.

9 Example 2: standardization

Same training values: 2, 4, 4, 4, 5, 5, 7, 9. Population standard deviation

Mean	40 ÷ 8	**5**
Std deviation	√(32 ÷ 8)	**2**
Training value 9	(9 − 5) ÷ 2	**2.0**
New test value 11	(11 − 5) ÷ 2 (frozen)	**3.0**

IN THE LESSON

- Now standardize the same training data.
- The mean is forty divided by eight, which is five.
- The deviations from five are minus three, minus one three times, zero twice, two and four.
- Their squares add to thirty two.
- Dividing by eight gives a variance of four, so the standard deviation is two.
- Take the training value nine.
- Nine minus five, over two, is two.
- It sits two standard deviations above the mean.
- The training value four becomes minus point five.
- Finally the test value of eleven becomes eleven minus five, over two, which is three.
- We use the training mean and standard deviation, never the test sample's own.
- This illustration uses the population standard deviation, dividing the squared deviations by the number of observations, eight.
- Using the number of observations minus one would give a slightly larger standard deviation.

10 Training, validation and test samples

- Training: estimate the model's parameters
- Validation: tune hyperparameters, pick a model
- Test: touched once, at the end, unbiased estimate
- Typical: 60 to 80 percent of data trains the model

IN THE LESSON

- Now the splitting of data.
- The guiding rule is that a model must never be judged on data it has already seen.
- So the data is divided into three samples, each with a distinct job.
- The training sample is used to estimate the model's parameters.
- The validation sample is used to tune the hyperparameters, such as the size of a penalty or the number of clusters, and to choose between competing models.
- Because the model never fits to it, the validation sample reveals overfitting.
- The test sample is held back and touched once, at the very end, to give an unbiased estimate of how the final model will do on new data.
- Typically most of the data goes to training, perhaps sixty to eighty per cent.

11 Example 3: splitting a time series

Illustration: 120 monthly observations, split 60 / 20 / 20 in time order

Training	months 1 to 72	**72 (60%)**
Validation	months 73 to 96	**24 (20%)**
Test (most recent)	months 97 to 120	**24 (20%)**
Check	72 + 24 + 24	**120**

IN THE LESSON

- Illustration.
- We have one hundred and twenty months of returns data and choose a sixty, twenty, twenty split.
- Sixty per cent of one hundred and twenty is seventy two, so the training sample is months one to seventy two.
- Twenty per cent is twenty four months, so validation is months seventy three to ninety six.
- The remaining twenty four months, ninety seven to one hundred and twenty, are the test sample.
- Notice the order.
- With time series the sets follow chronological order, and the test set is the most recent data.
- Seventy two plus twenty four plus twenty four is one hundred and twenty, so nothing is lost or counted twice.
- Random shuffling is wrong here, because it would leak future information into training.

12 Rules that protect the test sample

- Learn scaling and other parameters from training only
- Make every tuning decision on the validation sample
- Touch the test sample once, at the end
- Repeated peeking turns the test set into validation

IN THE LESSON

- Three rules keep the process honest.
- First, fit all preparation on the training sample alone.
- The scaling parameters, and anything else learned from data, come from training and are then applied unchanged to validation and test.
- Second, use the validation sample for every decision about tuning, so the choice of penalty or number of clusters never touches the test data.
- Third, touch the test sample once.
- If you keep going back to the test sample, trying a change and checking again, you are quietly tuning against it, and its estimate stops being unbiased.
- It becomes just another validation set.
- The test sample earns its value by staying unseen until the very end.

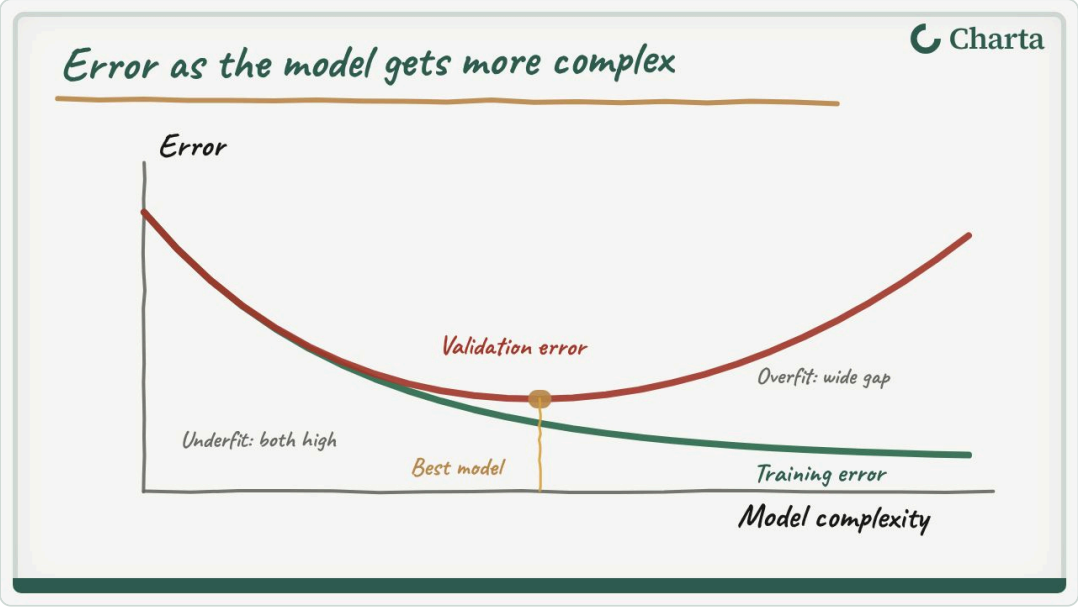
13 Underfitting versus overfitting

- Underfit: too simple, high error on every sample
- High bias; the gap between the two errors is small
- Overfit: learned noise, tiny training error
- High variance; the gap between errors is large
- Diagnostic: training error versus validation error

IN THE LESSON

- Now the central trade-off.
- Underfitting means the model is too simple to capture the real pattern, like drawing a straight line through a clearly curved relationship.
- Error is high on the training data and equally high on new data.
- This is a problem of high bias.
- Overfitting is the opposite.
- The model is too complex and has learned the noise as well as the signal, like a student who memorizes last year's answers.
- Training error is very low, but error on validation or test data is much higher.
- That is a problem of high variance.
- The diagnostic is the gap between training error and validation error.
- A small gap with both high means underfitting.
- A big gap means overfitting.

14 Error as the model gets more complex



IN THE LESSON

- Here is the picture.
- Along the bottom the model becomes more complex, from very simple on the left to very flexible on the right.
- Up the side is error.
- The training error, in green, falls all the way, because a flexible model can always fit what it has seen.
- The validation error, in red, falls at first, then turns back up.
- On the left both errors are high and close together: underfitting.
- On the right the training error is tiny but the validation error has climbed, so the gap is wide: overfitting.
- The best model sits at the bottom of the validation curve, where the model is as complex as the data can support.
- That point is what the validation sample helps us find.

15 Example 4: diagnosing three models

Illustration: error rates on training and validation samples

A: simple	train 20%, valid 21%	**Underfit**
B: moderate	train 9%, valid 11%	**Choose B**
C: flexible	train 1%, valid 15%	**Overfit**
Gap, valid − train	A: 1, B: 2, C: 14	**Points**

IN THE LESSON

- Illustration with made up errors.
- A team fits three models to loan data, and records the error rate on the training sample and on the validation sample.
- Model A, the simplest, has a training error of twenty per cent and a validation error of twenty one.
- Both are high and the gap is one point, so it is underfitting.
- Model B has nine per cent and eleven per cent, a gap of two.
- Model C, the most flexible, has just one per cent in training but fifteen per cent on validation, a gap of fourteen.
- That is overfitting.
- The right choice is model B, because it has the lowest validation error, eleven per cent, and that is what predicts the future.

16 Remedies for each problem

Overfitting: restrain the model. Underfitting: free it up

Problem	Remedies
Overfit: data	More training data
Overfit: complexity	Fewer features, simpler model
Overfit: control	Lasso or ridge, early stopping
Underfit: model	More features, nonlinear terms
Underfit: penalty	Weaker penalty, flexible model

IN THE LESSON

- What do we do about each problem?
- For overfitting, the remedies all reduce the model's freedom or give it more evidence.
- Collect more training data, use fewer features, choose a simpler model, add regularization such as a lasso or ridge penalty, or use early stopping, which halts training before the model starts memorizing noise.
- For underfitting, the remedies add freedom.
- Supply more features, add interaction or nonlinear terms, choose a more flexible model, or weaken the penalty.
- Notice that the remedies are opposites, and the penalty is a dial.
- Too much regularization can swing a model from overfit all the way to underfit, so we tune that dial against the validation sample.

17 Principal components: the idea

- Compresses many correlated features into a few
- Each is a linear combination of the original features
- Uncorrelated components, ordered by variance explained
- Works: correlated features overlap in information
- Unsupervised: it ignores the target variable

IN THE LESSON

- Principal components analysis, or P C A, compresses many correlated features into a few new variables.
- Think of five bond yields at different maturities.
- They move together, so they carry overlapping information, and most of their movement can be described by a couple of directions, say a general shift up or down and a tilt.
- Each principal component is a linear combination of the original features.
- The components are uncorrelated with each other, and they are ordered.
- The first captures the largest share of the total variance, the second captures the largest share of what remains, and so on.
- You keep enough to reach a target share and replace the original features with their scores.
- P C A is unsupervised: it never looks at the target.

18 How much variance does a component explain?

Share = eigenvalue ÷ sum of eigenvalues
Standardized data: sum = number of features

Eigenvalue	One component's eigenvalue
Sum of eigenvalues	Feature count, if standardized
Share of variance	Fraction of total variance

IN THE LESSON

- Each component comes with an eigenvalue, which measures how much variance it captures.
- The share of total variance explained by a component is its eigenvalue divided by the sum of all the eigenvalues.
- There is a handy fact.
- When the data have been standardized, each feature contributes a variance of one, so the sum of the eigenvalues equals the number of features.
- With five standardized features, the eigenvalues add to five.
- That makes the shares quick to compute.
- You then add the shares from the largest downward until you reach a target, such as ninety per cent, and that tells you how many components to keep.
- This is why rescaling matters here as well.

19 Example 5: choosing the components

Illustration: eigenvalues 3.6, 0.9, 0.3, 0.15, 0.05; target 90%

Sum of eigenvalues	3.6+0.9+0.3+0.15+0.05	**5.0**
Component 1 share	3.6 ÷ 5	**72%**
Component 2 share	0.9 ÷ 5	**18%**
Cumulative, keep 2	72% + 18%	**90%**

IN THE LESSON

- Illustration.
- Five standardized yields give five eigenvalues: three point six, point nine, point three, point one five and point zero five.
- They add to five, which equals the number of features, a useful check.
- The first component explains three point six over five, which is seventy two per cent of the variance.
- The second explains point nine over five, eighteen per cent.
- Together that is ninety per cent.
- Our target is ninety, so we keep two components and replace five columns of data with two columns of scores, losing only ten per cent of the variation.
- Those two components might be read as a level and a slope, but the algorithm does not label them.
- We interpret them afterward.

20 Cumulative variance explained

Eigenvalues 3.6, 0.9, 0.3, 0.15, 0.05 out of 5.0

Component	Share | Cumulative
1	72% | 72%
2	18% | 90%
3	6% | 96%
4	3% | 99%
5	1% | 100%

IN THE LESSON

- Here is the whole list of shares for the same example.
- Component one explains seventy two per cent, and the running total is seventy two.
- Component two adds eighteen, for ninety.
- Component three adds six, for ninety six.
- Component four adds three, for ninety nine.
- Component five adds the last one per cent, for one hundred.
- Notice the pattern: each extra component adds less.
- That is exactly why compression works, since a few directions hold most of the variation.
- If the target had been ninety five per cent, we would need three components.
- A higher target keeps more detail but compresses less.
- Remember too that because the compression ignores the target, the components kept are not guaranteed to be the best predictors.

21 K-means: how it assigns observations

- Analyst sets K; pick K starting centroids at random
- Assign each point to its nearest centroid
- Recompute each centroid as the mean of its members
- Repeat until the assignments stop changing
- Rescale first; run several starts

IN THE LESSON

- K means is an unsupervised method that splits observations into a chosen number, K, of clusters.
- The analyst must set K, because the algorithm cannot choose it.
- The recipe has four steps.
- Pick K starting centroids, often at random.
- Assign every observation to its nearest centroid, usually by straight-line, Euclidean distance.
- Recompute each centroid as the mean of the observations assigned to it.
- Then repeat the assignment and the update until the assignments stop changing.
- Because the result can depend on the starting centroids, the algorithm is often run several times.
- And since distance drives everything, features should be rescaled first, or the large-scale variable decides the clusters.

22 Example 6: K-means, round one

Illustration: points 1, 2, 4, 9, 11, 12; K = 2; start centroids 2 and 4

Assign to centroid 2	1, 2	**Cluster A**
Assign to centroid 4	4, 9, 11, 12	**Cluster B**
New centroid A	(1 + 2) ÷ 2	**1.5**
New centroid B	(4+9+11+12) ÷ 4	**9**

IN THE LESSON

- Illustration with one feature and six observations: one, two, four, nine, eleven and twelve.
- Set K to two, with starting centroids at two and four.
- Assignment.
- The value one is nearer to two, so it joins the first centroid, and so does two itself.
- Four is exactly on the second centroid, and nine, eleven and twelve are all nearer to four than to two, so they join the second.
- The first cluster is one and two, the second is four, nine, eleven and twelve.
- Now update.
- The first centroid is the mean of one and two, which is one point five.
- The second is the mean of four, nine, eleven and twelve, thirty six over four, which is nine.

23 Example 7: K-means, round two and finish

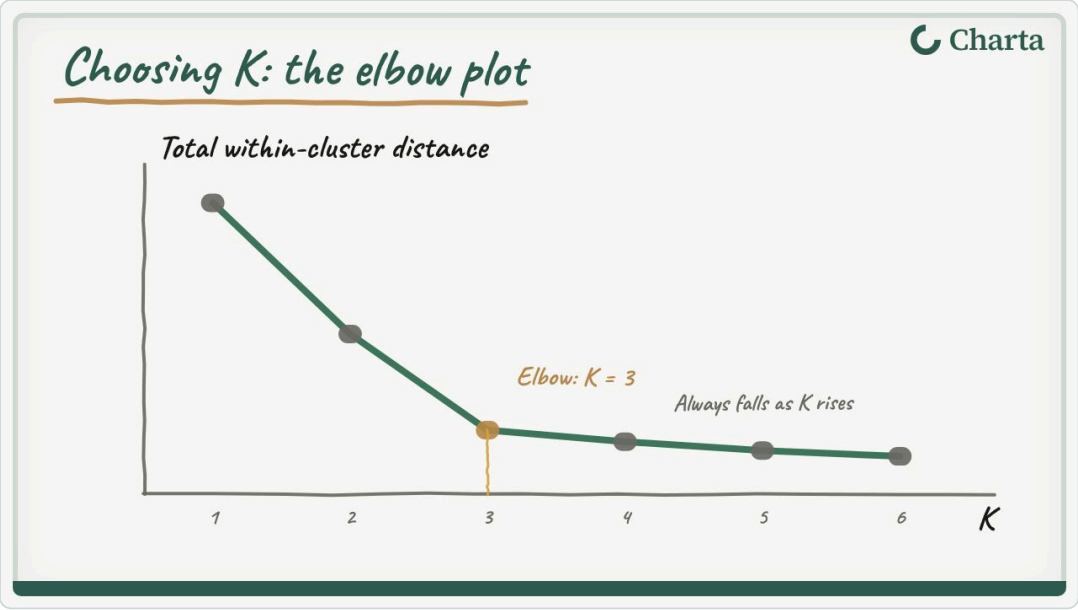
Centroids now 1.5 and 9. Reassign every point, update, check again

Point 4 re-assigned	2.5 from A, 5 from B	**Moves to A**
Clusters	A: 1, 2, 4 B: 9, 11, 12	**Updated**
Centroids	A: 7/3, B: 32/3	**2.33, 10.67**
Next pass	no point changes cluster	**Converged**

IN THE LESSON

- Now repeat with the new centroids, one point five and nine.
- The value four is two point five from the first centroid, but five from the second, so it moves to cluster A.
- The values one and two stay with A, and nine, eleven and twelve stay with B.
- Cluster A is now one, two and four, with a mean of seven thirds, about two point three three.
- Cluster B is nine, eleven and twelve, with a mean of thirty two thirds, about ten point six seven.
- One more pass changes nothing, since four is nearer to two point three three than to ten point six seven.
- The assignments have stopped changing, so the algorithm has converged.
- Two tidy groups emerged without a single label.

24 Choosing K: the elbow plot



IN THE LESSON

- How do we choose K?
- We cannot simply pick the K with the smallest total distance from points to their centroids, because that total always falls as K rises.
- With one cluster per observation, the distance would be zero, and the clusters would be useless.
- Instead we draw an elbow plot.
- On the horizontal axis is K, and up the side is the total within-cluster distance.
- In this illustration it drops steeply from one cluster to two, and again from two to three, then flattens.
- The bend, or elbow, is at three clusters.
- Beyond that, extra clusters buy little.
- We choose K at the elbow, a judgment about diminishing returns, not a mechanical minimum.

25 Natural language processing: text to numbers

- Lower-case; tokenize; drop stop words (the, of)
- Reduce words to a stem or lemma: default, defaulted
- Bag of words: count each term in each document
- TF-IDF: term frequency scaled by rarity
- A model then classifies or scores each document

IN THE LESSON

- Natural language processing, or N L P, turns unstructured text into numbers that a model can use.
- The preparation is a pipeline.
- First lower-case the text.
- Then split it into tokens, which are usually words.
- Remove stop words such as the and of, which carry little meaning.
- Reduce words to a stem or a lemma, so that default, defaults and defaulted count as one.
- The cleaned text is then converted to features.
- The simplest is a bag of words, a count of each term in each document.
- A refinement is the T F I D F weight, which scales a term's frequency by how rare the term is across documents.
- Models then classify or score the documents, for instance as positive or negative.

26 Example 8: a T F I D F weight

Illustration: 10 documents; document of 100 words; IDF = ln(docs ÷ docs with term)

'default' term frequency	5 ÷ 100	**0.05**
'default' rarity (IDF)	ln(10 ÷ 2)	**1.61**
'default' TF-IDF	0.05 × 1.61	**0.08**
'the': 6 times, in all 10	0.06 × ln(10 ÷ 10)	**0**

IN THE LESSON

- Illustration.
- We have a collection of ten documents, say ten earnings call transcripts.
- One of them has one hundred words, and the word default appears in it five times.
- The term frequency is five over one hundred, or point zero five.
- The word default shows up in only two of the ten documents, so it is fairly rare.
- One common version of the rarity factor is the natural log of ten over two, about one point six one.
- The weight is point zero five times one point six one, roughly point zero eight.
- Now take the word the, a stop word.
- Even if it were not removed, it appears six times but in every document.
- Its rarity factor is the log of ten over ten, which is zero, so its weight is zero.
- Common words are switched off, rare informative words count.

27 Uses and weaknesses of N L P in finance

- Uses: sentiment of news, calls, central bank text
- Also: extracting risk disclosures, chatbots
- Weaknesses: negation, sarcasm
- General word lists misread financial vocabulary
- Fixes: domain dictionaries, context-aware models

IN THE LESSON

- In finance, N L P is used to score the sentiment of news and earnings calls, to read central bank statements, to extract risk disclosures from filings, and to power chatbots.
- It also has standard weaknesses.
- Negation is one: not profitable contains the word profitable.
- Sarcasm is another.
- A third is that word lists written for general language can misread financial vocabulary.
- In ordinary speech the word liability might sound negative, yet in accounting it is routine, and general lists may score words like tax or cost as negative, though in finance they are usually neutral.
- Domain-specific dictionaries, written for finance, help, and so do context-aware models that read each word in the light of the words around it.

28 Reinforcement learning: the loop

- Agent: observe state, act, get reward, new state
- Goal: maximize discounted cumulative reward
- Q(s, a) = long-run reward of action a in state s
- Explore at random with probability ε, else exploit
- Exploit = take the action with the highest Q value

IN THE LESSON

- Reinforcement learning is a loop.
- An agent observes a state, takes an action, receives a reward and moves to a new state.
- Its aim is to maximize the discounted cumulative reward, meaning rewards today count fully and rewards further away count a bit less.
- The Q value, written Q of state and action, estimates the long-run reward of taking that action in that state and then acting well afterward.
- Learning is a trade-off.
- The agent explores, trying actions at random with a small probability called epsilon, and otherwise exploits, picking the action with the highest Q value so far.
- Too little exploration and it can get stuck with a mediocre habit.
- Too much and it wastes effort.

29 The Q-learning update

Q_new = Q_old + α × [R + y × max Q(next) − Q_old]

Q value	Current estimate, Q_old
Learning rate, alpha	Learning rate, 0 to 1
Reward	Reward just received
Discount factor, gamma	Discount factor, 0 to 1
Best next Q value	Highest Q in the next state

IN THE LESSON

- Q learning updates a Q value a little at a time.
- The new Q equals the old Q plus alpha times a bracket.
- Inside the bracket is the reward received, plus gamma times the best Q value available in the next state, minus the old Q.
- The bracket is the gap between a better estimate and the old one, so the update nudges Q toward the better estimate.
- Alpha is the learning rate between zero and one.
- Gamma is the discount factor, also between zero and one.
- A point to remember: the maximum is taken over the actions in the next state, whether or not the agent actually chooses that action there.

30 Example 9: one Q-value update

Illustration: Q_old 2, reward 1, next-state Q values 3 and 4, α = 0.5, y = 0.9

Better estimate	1 + 0.9 × 4	**4.6**
Gap to old Q	4.6 − 2	**2.6**
Step taken	0.5 × 2.6	**1.3**
New Q value	2 + 1.3	**3.3**

IN THE LESSON

- Illustration.
- The agent is in a state where the current Q value for an action is two.
- It takes that action, and receives a reward of one.
- In the new state, the available actions have Q values of three and four, so the best is four.
- Set alpha to point five and gamma to point nine.
- First, the better estimate: reward one plus point nine times four, which is four point six.
- Second, the gap: four point six minus the old value of two, which is two point six.
- Third, move halfway, since alpha is point five: point five times two point six is one point three.
- The new Q is two plus one point three, which is three point three.
- Notice the max uses four, even if the agent later explores.

31 Five mistakes to avoid

- Scaling on all data: use training parameters only
- Trusting in-sample fit: judge out-of-sample
- Peeking at test set: tune on validation, test once
- Picking K by lowest total distance: use the elbow
- Q target on chosen action: use max over next actions

IN THE LESSON

- Mistake one: rescaling or tuning on the whole sample.
- Minimum, maximum, mean and standard deviation must come from the training set only.
- Mistake two: trusting in-sample fit.
- A model that fits its training data tells you little, so judge by out-of-sample accuracy.
- Mistake three: peeking at the test sample.
- Tune on validation, and touch the test sample once.
- Mistake four: choosing the number of clusters by minimum total distance.
- That always falls as K rises, so use the elbow instead.
- Mistake five, in reinforcement learning: building the Q target from the action the agent actually chose.
- Take the maximum over next-state actions instead.
- Keep these five fixes in your head.

32 Recap: the main ideas

- Econometrics explains; ML predicts, out of sample
- Rescale on training; split train, valid, test
- Underfit = bias; overfit = variance; check the gap
- PCA: eigenvalue ÷ sum. K-means: assign, update
- NLP: text to features. Q: R + γ × max Q next

IN THE LESSON

- Let's pull it together.
- Econometrics starts from theory and explains; machine learning starts from data and predicts, judged out of sample.
- Supervised learning has labels, unsupervised has none, and reinforcement learning learns from rewards.
- Rescale with min-max or standardization, using training parameters only.
- Split into training, validation and test, in time order for time series.
- Underfitting is high bias, overfitting is high variance, and the gap diagnoses it.
- P C A shares are eigenvalue over the sum of eigenvalues.
- K means repeats assignment and updating, with K chosen by the elbow.
- N L P turns text into features.
- Q learning moves the Q value toward reward plus discounted best next value.

33 Try three quick questions

1. Train error 2%, validation error 14%. Problem and a remedy?
   Answer: Overfitting (high variance). Add regularization, data, or simplify.
2. Eigenvalues 2.4, 1.0, 0.4, 0.2 (standardized). Share of PC 1?
   Answer: Sum = 4, so 2.4 ÷ 4 = 60%.
3. Q_old 5, α 0.1, γ 0.9, R −2, best next Q 6. New Q?
   Answer: 5 + 0.1 × (−2 + 5.4 − 5) = 4.84.

IN THE LESSON

- Now try three quick questions yourself.
- Pause after each one.
- Question one.
- A model has a training error of two per cent and a validation error of fourteen per cent.
- What is the problem, and name one remedy?
- That is overfitting, high variance, and a remedy is a stronger penalty.
- Question two.
- Four standardized features have eigenvalues of two point four, one, point four and point two.
- What share does the first component explain?
- Two point four over four, sixty per cent.
- Question three.
- Q old is five, alpha is point one, gamma is point nine, reward is minus two, and the best next Q is six.
- What is the new Q?
- Four point eight four.