UiO : University of Oslo

institutt for informatikk

IN3050/IN4050 - Introduction to Artificial Intelligence and Machine Learning

Lecture 9

Topics in supervised learning

Jan Tore Lønning

overview

Topics in supervised machine learning

- This presentation
- Videos
- Some jupyter notebooks
- scikit-learn

Topics in supervised machine learning

1. This intro
2. ML in scikit-learn:
   - video
   - notebook: sklearn_assignment_2
3. Overfitting and regularization -
   - video
4. Regularization in scikit-learn
   - video
   - notebook: boston_regularization_cv
4. Bias-variance tradeoff
   - video
5. Cross-validation
   - video
   - notebook: boston_regularization_cv
6. Ensemble learning and random forest
   - video

- The slides and videos was mostly made last year (2020)
- Included this slide to match the page numbering in the videos ☺

scikit-learn

Jan Tore Lønning

Why use toolkits

Toolkits like scikit-learn

- More efficient, faster
- Tested code (error-free)
- Comprehensive
  - many different learners
- Flexible
  - many options
- Consistent interface
- Well-integrated with Python, NumPy, etc.

Own implementation

Help us to

- Understand the inner workings of the algorithms
- Make informed choices regarding:
  - Learner
  - Parameter settings

Other toolkits

Why scikit?

- Why not jump directly to deep learning toolkits?
- Many problems can be solved with simpler learners
    - Don't use a jumbo jet when you are going to the store
- A base-line for more advanced learners
- Easy to use

Toolkits for deep learning

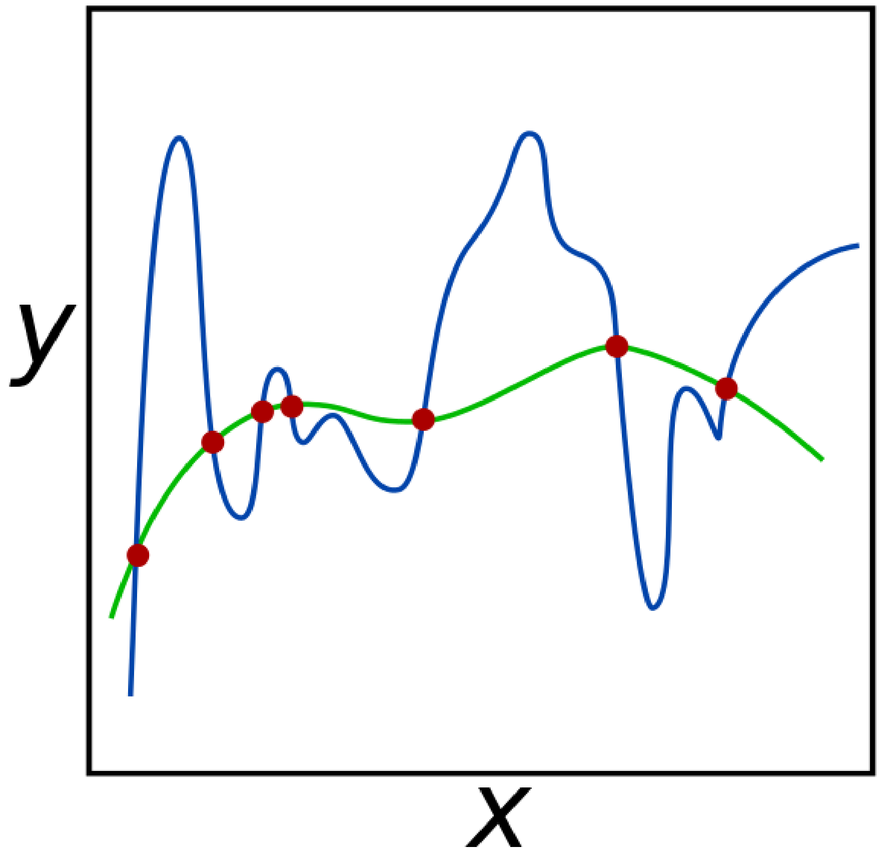
- Tensor Flow
    - with Keras
- PyTorch
- and more

Overfitting and regularization

Jan Tore Lønning

Overfitting

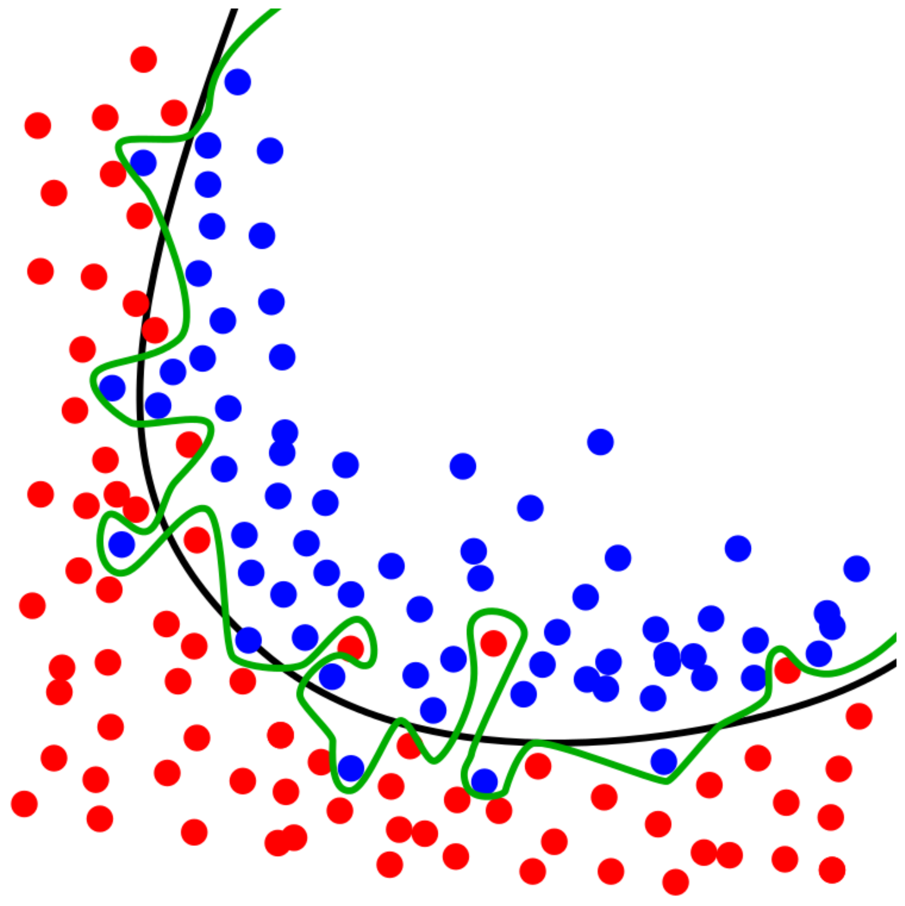
- Goal in supervised ML:
  - Construct models that fit the training set
- Sometimes, we succeed too well:
  - Models fit training data very well
  - Does not generalize to other data



Source: Wikipedia

Overfitting

- Classification



Source: Wikipedia

How come?

- Whenever we choose a random sample of individuals from a larger population,
  there might be attributes that have a different distribution in the sample than in the population at large
- The learning algorithm may focus on these attributes

- Example:
- Pick 100 Swedes and 100 Norwegians at random
- We might find properties shared by the 100 Swedes and not the 100 Norwegians which are not representative for Swedes and Norwegians in general

Gradient descent-based algorithms

Classification:

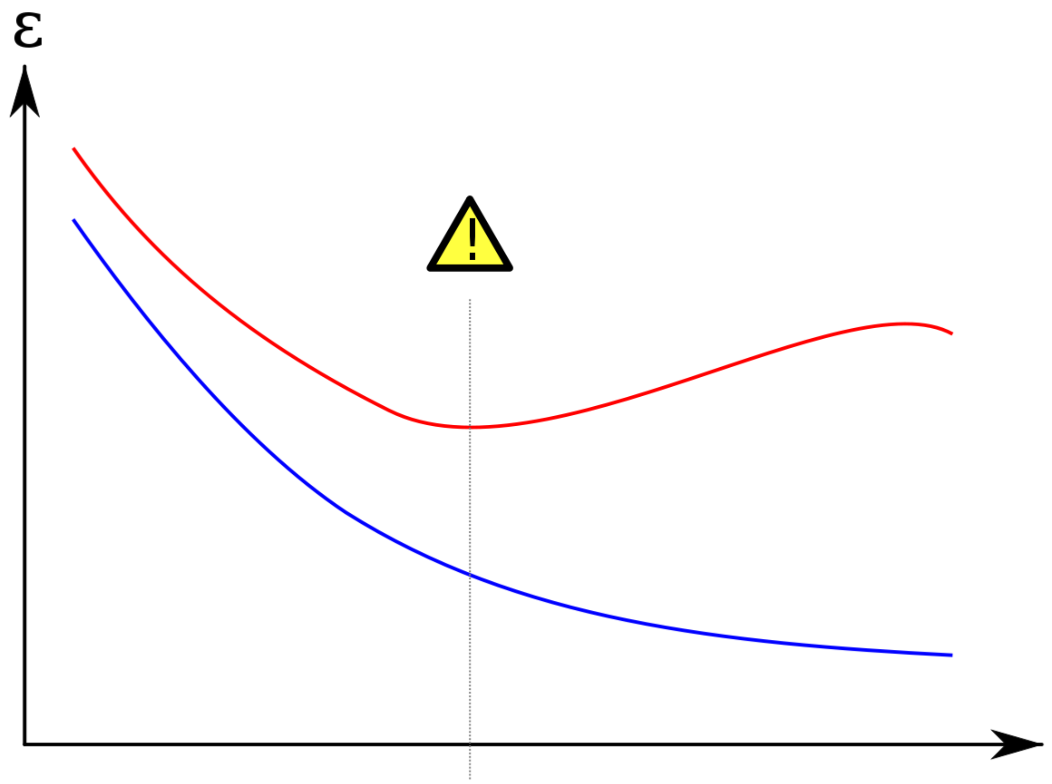
- We replace the true objective:
    - accuracy, recall, precision, etc.
- with a loss function
- The loss function may continue to improve long after we have reached the best accuracy
- It moves weights to the best discriminators
- They are not necessarily representative

- Typically, overfitting with
    - many features
    - relative to the training data
- Advice:
    - At least ten times more training datapoints than features
- Not always possible
    - E.g., NLP:
        - Millions of features
        - A few thousand datapoints.

Remedies

Alt. 1: Early stopping

Alt. 2: Regularization



Source: Wikipedia

Regularization

So far

- Training set $(\boldsymbol{X}, \boldsymbol{t})$, where
  - $\boldsymbol{X}$ is the data (inputs)
  - $\boldsymbol{t}$ the corresponding target values
- A model with weights $\boldsymbol{w}$, s.t. it
  - from $\boldsymbol{X}$ and $\boldsymbol{w}$ predicts values $\boldsymbol{y}$.
- A loss function which tells how well $\boldsymbol{y}$ approximates $\boldsymbol{t}$.
- Learning objective:
  Find the weights that minimize the loss:
- $\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmax}}(-Loss(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{w}))$

Regularization

- Replace
  - $-Loss(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{w})$ with
  - $-Loss(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{w}) - \alpha R(\boldsymbol{w})$
- Where
  - $R(\boldsymbol{w})$ punishes large weights
  - α determines how hard they should be punished
  - α is to be tuned on a validation set

Alternatives for regularization

Ridge regularization

- The most common uses L2-distance
- $R(\boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{w} = \sum_{i=0}^{m} {w_i}^2$
- Example:
  - $\boldsymbol{w}_1 = (0,0,0,1), R(\boldsymbol{w}_1) = 1$
  - $\boldsymbol{w}_2 = (0.1, 0.2, 0.3, 0.4),$ $R(\boldsymbol{w}_2) = 0.3$
  - Preference for $\boldsymbol{w}_2$

Other alternatives

- Lasso regularization:
  - Uses L1
  - $R(\boldsymbol{w}) = \sum_{i=0}^{m} |w_i|$
- Elastic nets:
  - uses both L2 and L1

To the notebook

- In scikit-learn, you find regularization in
  - Ridge, for linear regression
  - RidgeClassifier, for linear regression classifier
  - LogisticRegression
  - and more

Bias-variance tradeoff

Meanings of "bias"

1. Bias term
   - Not today's topic

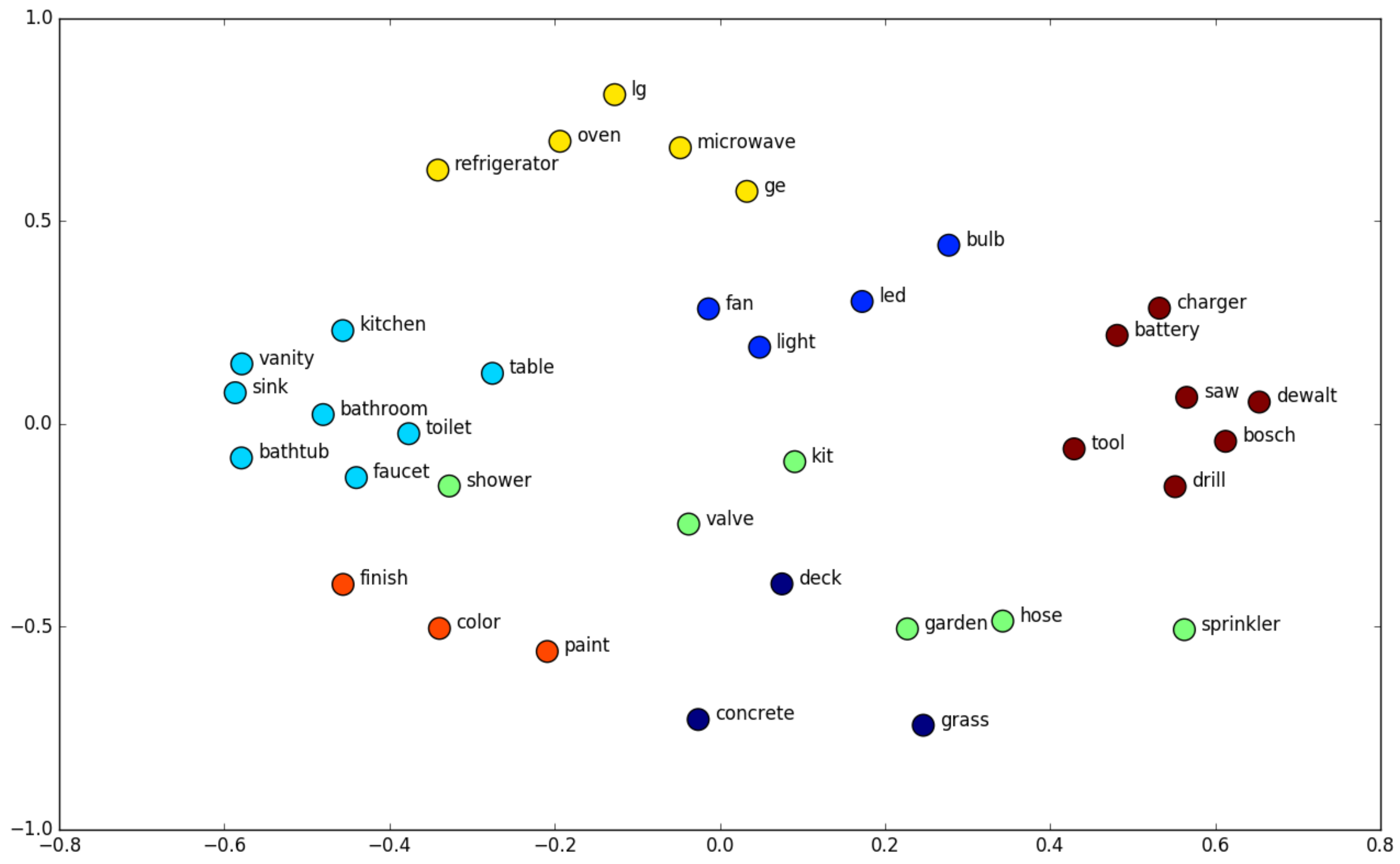
2. About a classifier:
   - *Unfairness*, *prejudice*
   - For ML-systems this might be because of the selection of
     - training examples, or
     - features

3. A statistics is biased
   - (this is what can be traded)

- Examples (of 2)
- Early speech recognition system for cars:
  - Performed much better on males than females
  - Reason: Mostly trained on men
- NLP, *word embeddings:*
  - *Man is to Computer Programmer as Woman is to Homemaker*

Bias (meaning 2) contd.

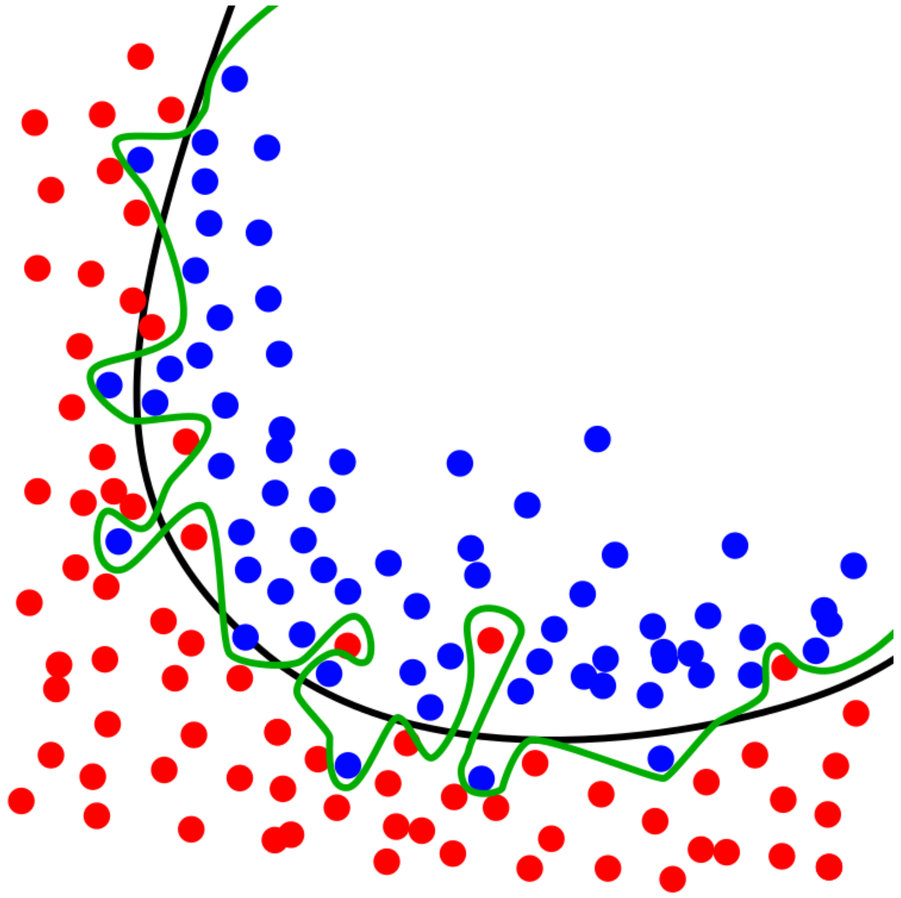
- http://vectors.nlpl.eu/explore/embeddings/en/



https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/

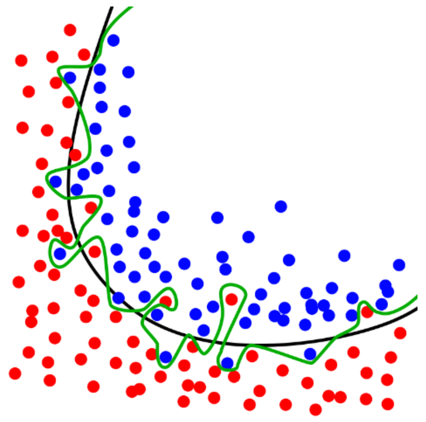
Bias-variance tradeoff

- Underlying idea
  - (from statistics, can be formalized)
- The training data $\mathrm{D} = (\boldsymbol{X}, \boldsymbol{t})$ can be described by an expression
- $t = f(\boldsymbol{x}) + \varepsilon$, where
  - ε is irreducible (noise) centered at 0
- The goal of the ML training is to find (an approximation) g to f, where $(g(x) - f(x))^2$ is minimal for x both from *D*, and from outside of D.



Source: Wikipedia

Bias-variance tradeoff

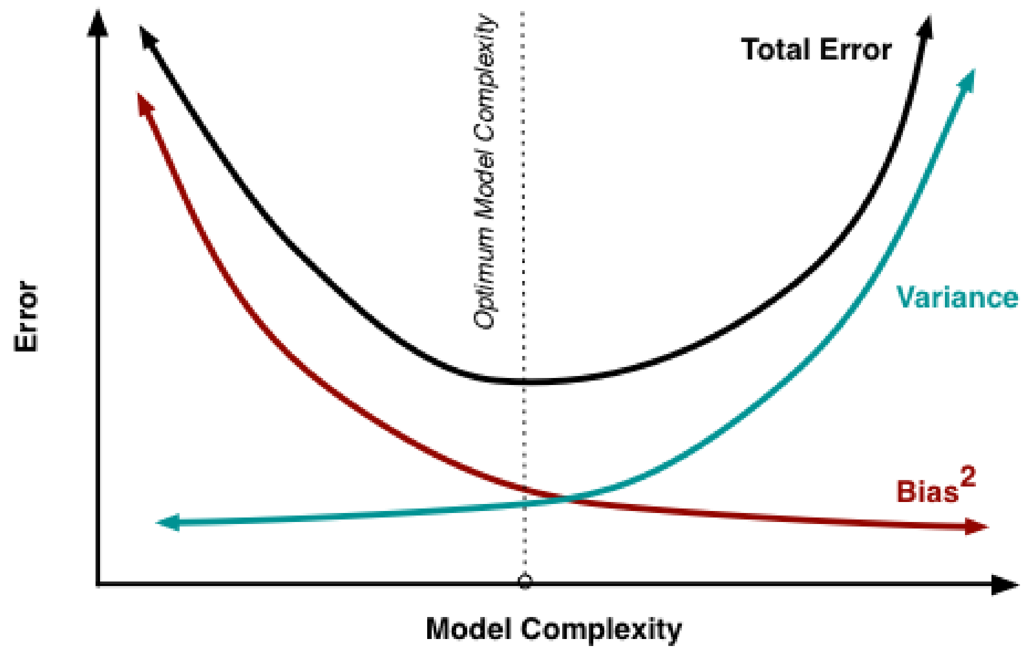


- In general, we will not succeed in finding g identical to f
- The expectation of the error can be split into
    - bias
    - variance

- Bias (meaning 3)
    - the classifiers systematically misses the target,
    - e.g. searching for linear classifiers to a non-linear problem
    - Underfitting
- Variance
    - Picking up some of the noise, being to sensible to small variation in the input data
    - cf. the green dec. boundary
    - Overfitting

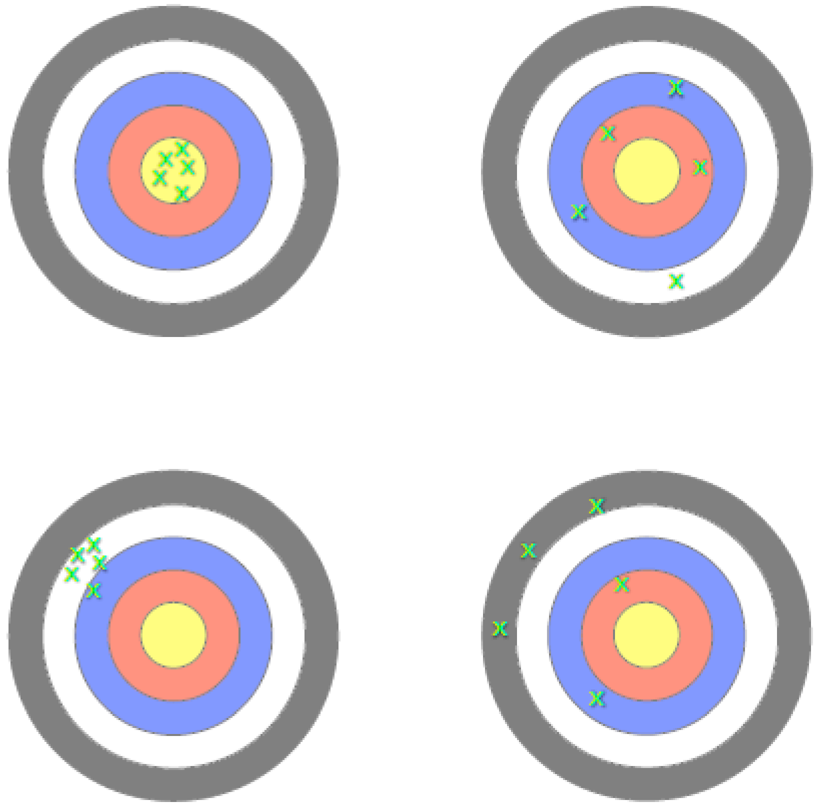
The tradeoff

- Traditional wisdom:
  - Lower bias gives higher variance,
  - and the other way around
- There is a sweet spot in the middle

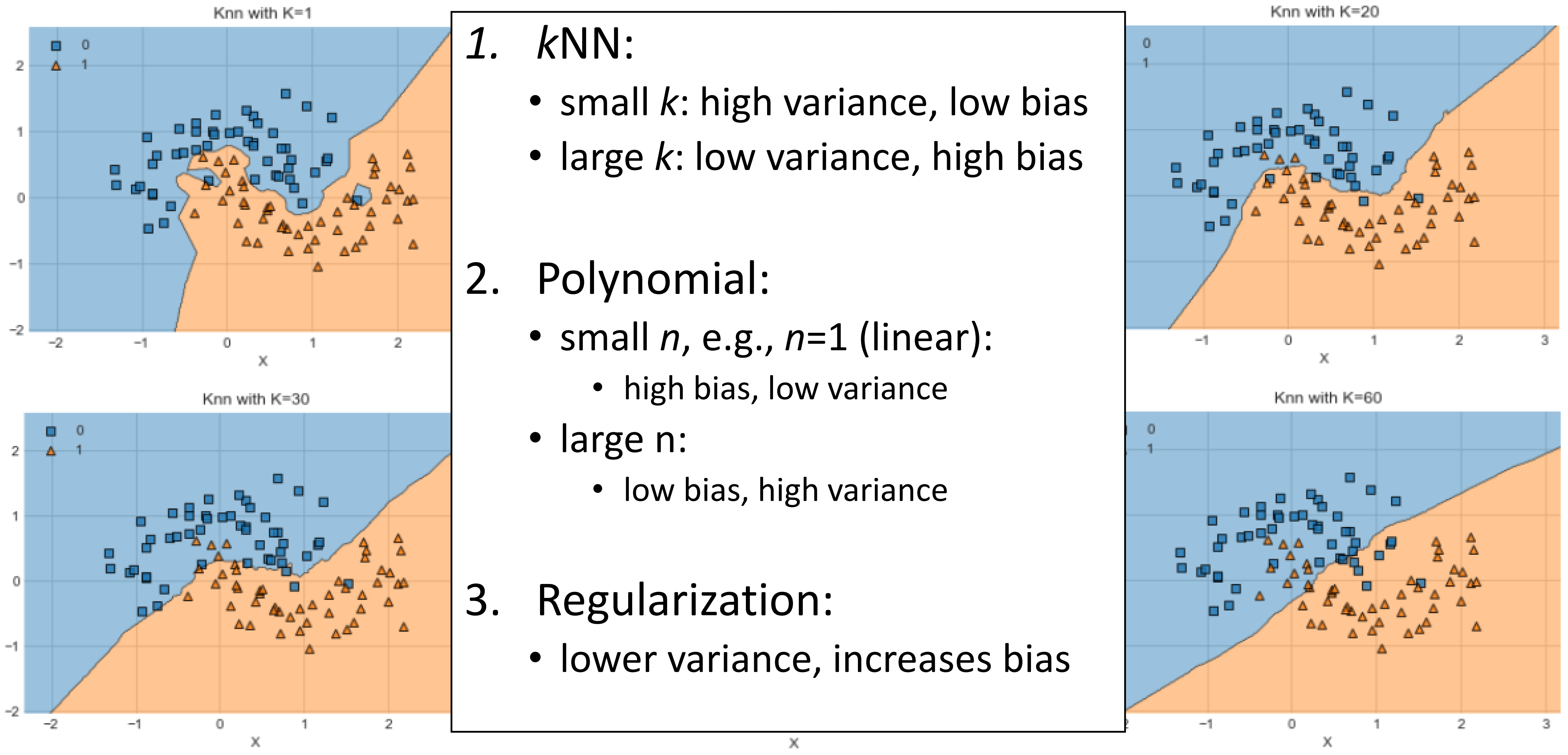


http://scott.fortmann-roe.com/docs/BiasVariance.html

Metaphor of bias vs variance



http://people.cs.bris.ac.uk/~flach/mlbook/materials/



1. *k*NN:
   - small *k*: high variance, low bias
   - large *k*: low variance, high bias

2. Polynomial:
   - small *n*, e.g., *n*=1 (linear):
     - high bias, low variance
   - large n:
     - low bias, high variance

3. Regularization:
   - lower variance, increases bias

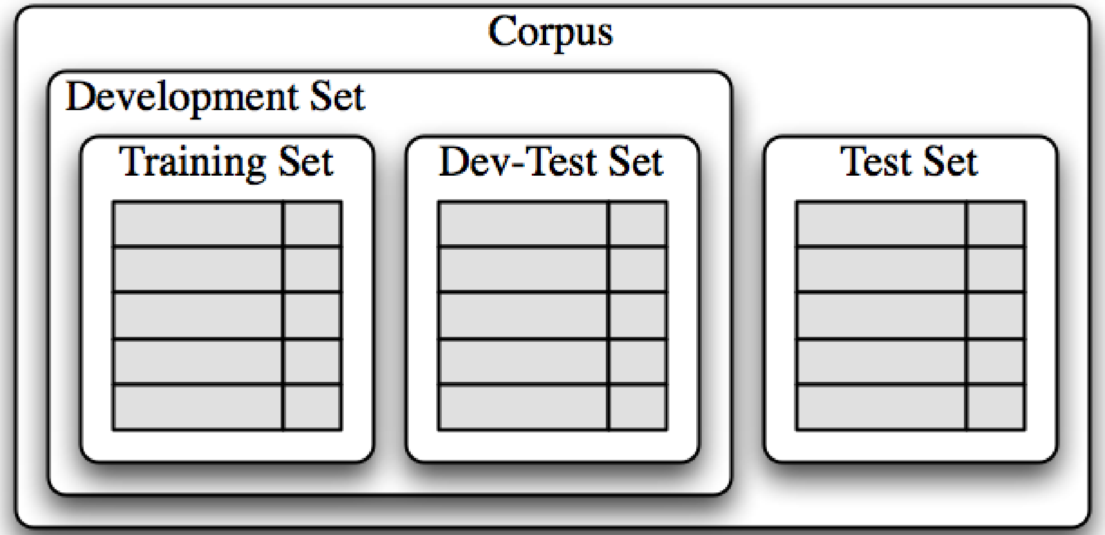
KNN visualization for the U-shaped dataset

https://towardsdatascience.com/knn-visualization-in-just-13-lines-of-code-32820d72c6b6

Cross-validation

Training and test sets (lecture 5)

- To measure improvement, we need (at least) two disjoint labeled sets:
  - Training set
  - Test set
- Train on the training set.
- Predict labels on the test set (after removing the labels)
- Compare the prediction to the given labels



https://www.nltk.org/book/ch06.html

- For repeated development we need (at least) two test sets.
  - One for repeated testing during development
  - One for final testing

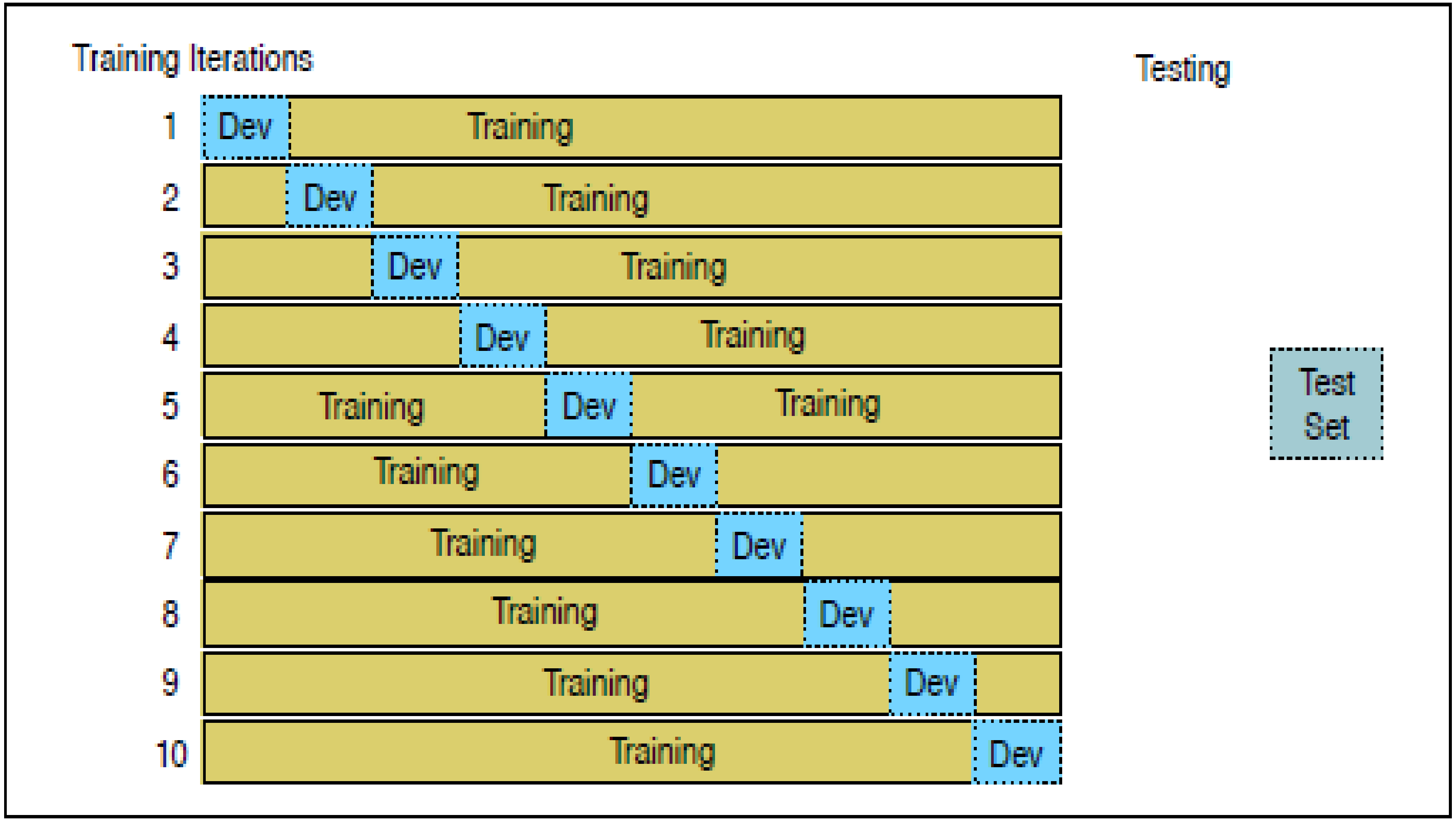


Figure 6.7 10-fold crossvalidation

https://web.stanford.edu/~jurafsky/slp3/

Cross-validation

- Small test sets ➔ Large variation in results
- N-fold cross-validation:
  - Split-off a final test set
  - Split the development set into n equally sized bins (e.g. n = 10)
  - Conduct n many experiments:
    - In experiment m, use part m as test set and the n-1 other parts as training set.
  - This yields n many results:
    - We can consider the mean of the results
    - We can consider the variation between the results.
      - Statistics!

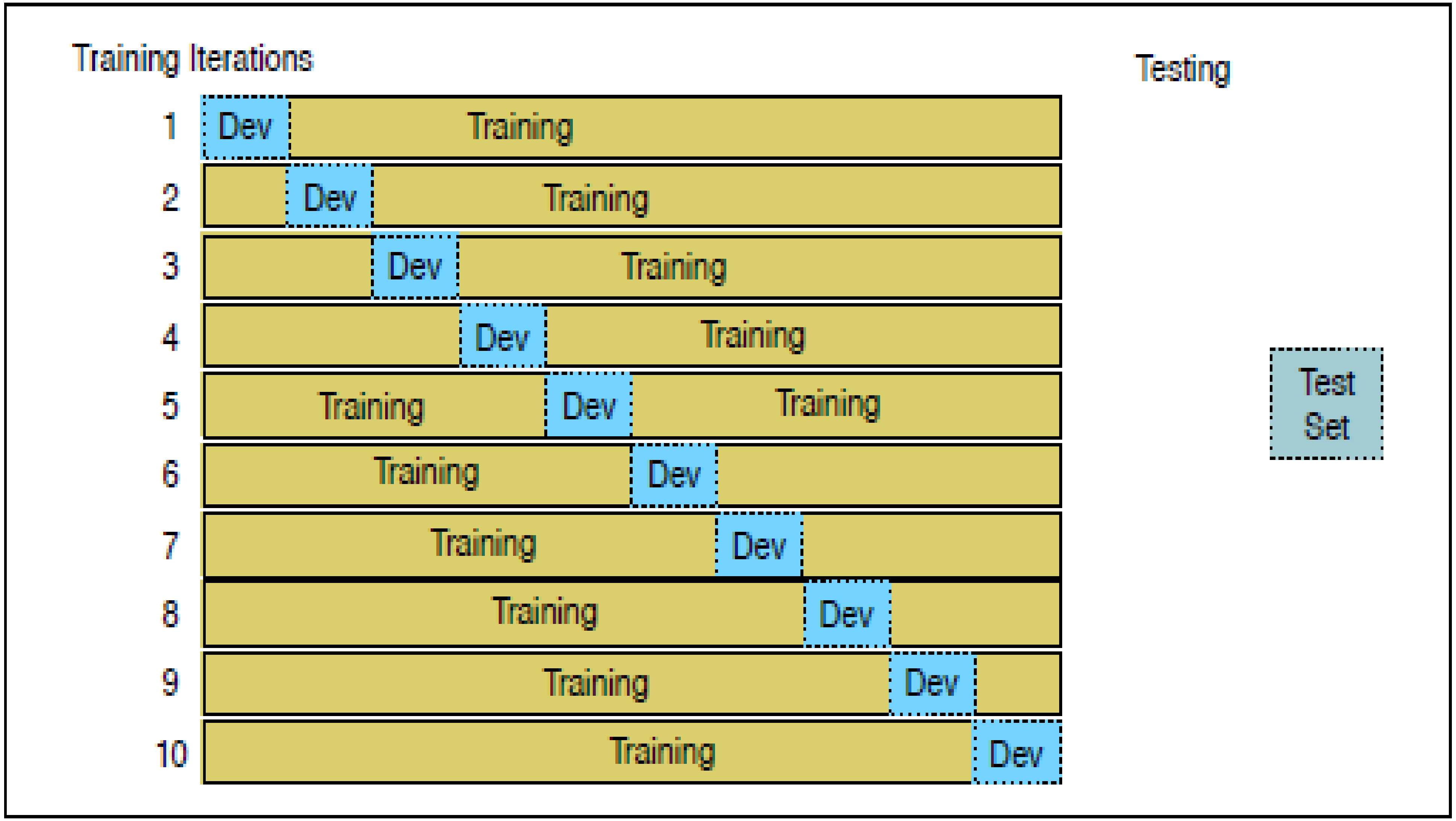
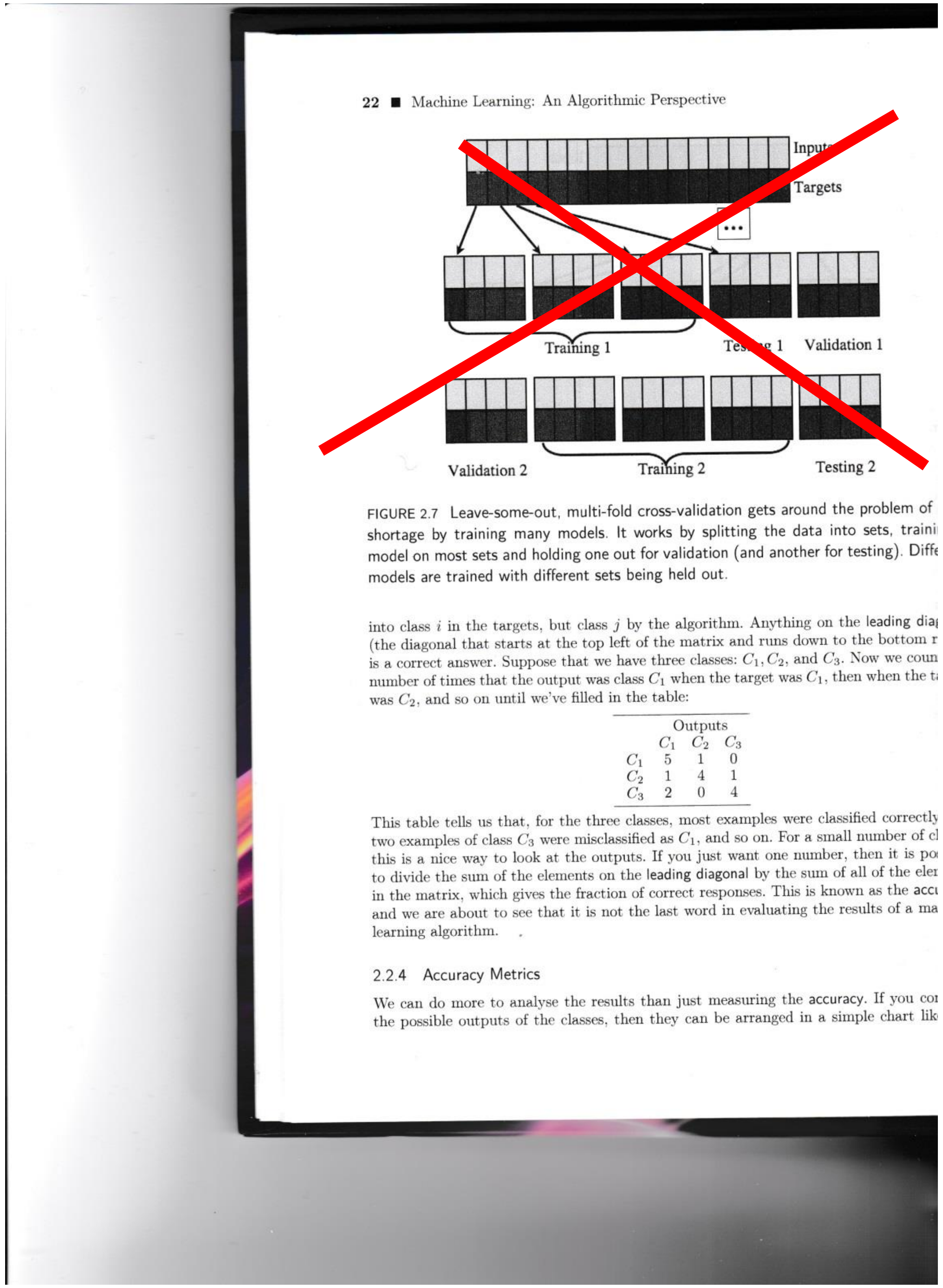


Figure 6.7 10-fold crossvalidation

https://web.stanford.edu/~jurafsky/slp3/

Marsland Ch. 2

- First edition 2009
- Second edition 2015:
  - Removed mistakes
  - Added chapter 2
    - With mistakes
- We use the same final test set. That is not changed with the folds.
- We might use several val-sets, say
  - One for testing and
  - one for tuning



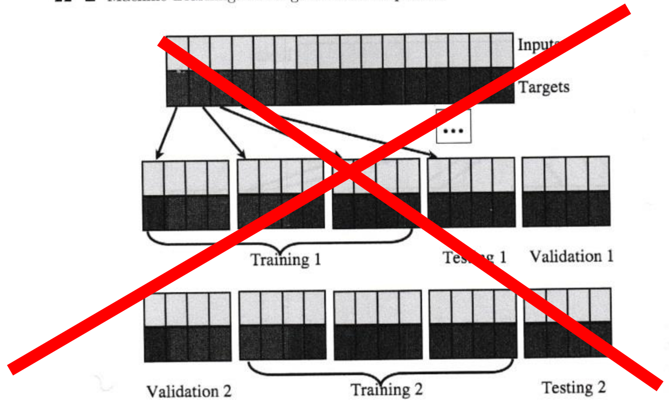


FIGURE 2.7 Leave-some-out, multi-fold cross-validation gets around the problem of shortage by training many models. It works by splitting the data into sets, traini model on most sets and holding one out for validation (and another for testing). Diffe models are trained with different sets being held out.

into class i in the targets, but class j by the algorithm. Anything on the leading dia (the diagonal that starts at the top left of the matrix and runs down to the bottom r is a correct answer. Suppose that we have three classes: C_1, C_2, and C_3. Now we coun number of times that the output was class C_1 when the target was C_1, then when the t was C_2, and so on until we've filled in the table:

	Outputs		
	C_1	C_2	C_3
C_1	5	1	0
C_2	1	4	1
C_3	2	0	4

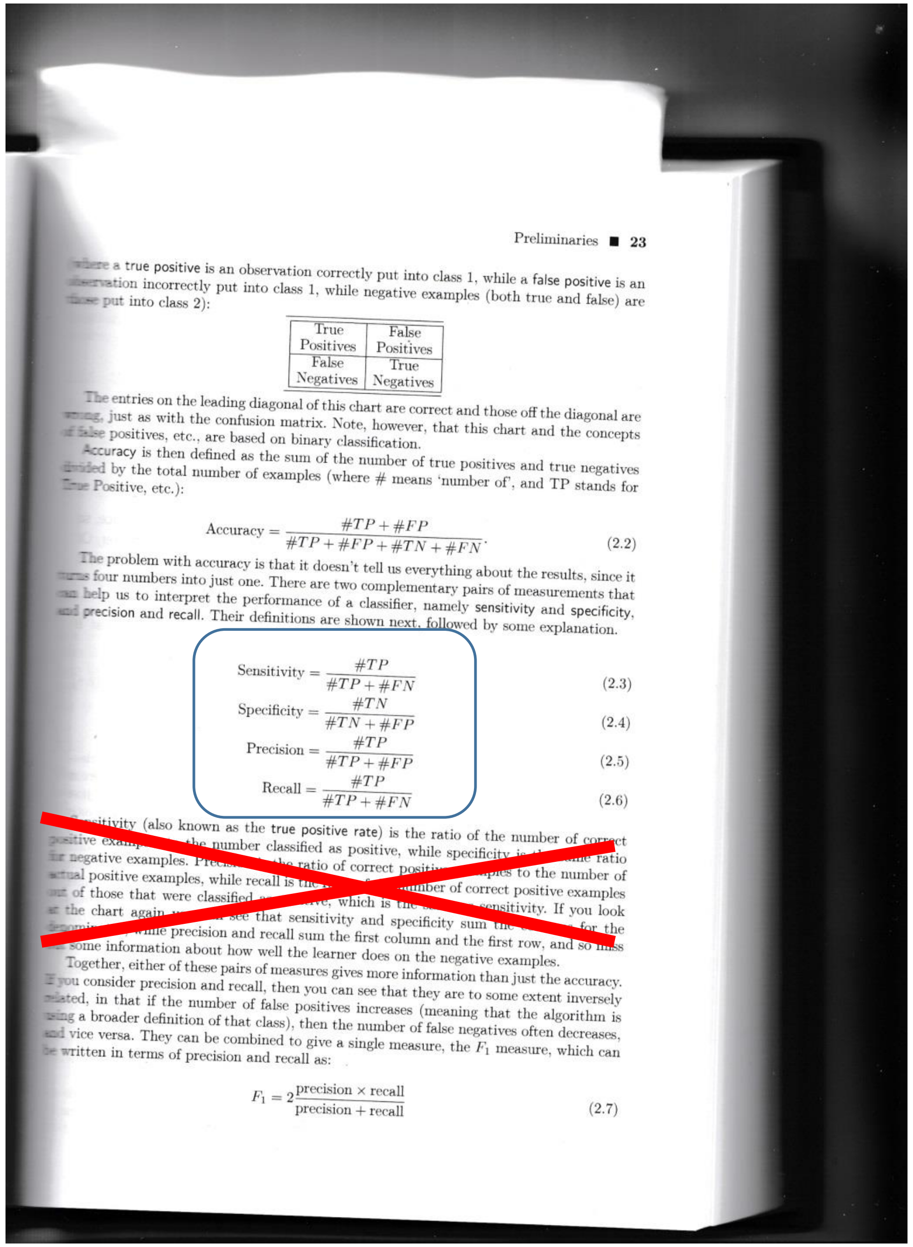
This table tells us that, for the three classes, most examples were classified correctly two examples of class C_3 were misclassified as C_1, and so on. For a small number of cl this is a nice way to look at the outputs. If you just want one number, then it is po to divide the sum of the elements on the leading diagonal by the sum of all of the eler in the matrix, which gives the fraction of correct responses. This is known as the accu and we are about to see that it is not the last word in evaluating the results of a ma learning algorithm.

2.2.4 Accuracy Metrics

We can do more to analyse the results than just measuring the accuracy. If you co the possible outputs of the classes, then they can be arranged in a simple chart lik

Precision and recall

- The formulas for Precision and Recall are correct
- The explanation mixes things up



where a true positive is an observation correctly put into class 1, while a false positive is an observation incorrectly put into class 1, while negative examples (both true and false) are those put into class 2):

True Positives	False Positives
False Negatives	True Negatives

The entries on the leading diagonal of this chart are correct and those off the diagonal are wrong, just as with the confusion matrix. Note, however, that this chart and the concepts of false positives, etc., are based on binary classification.

Accuracy is then defined as the sum of the number of true positives and true negatives divided by the total number of examples (where # means 'number of', and TP stands for True Positive, etc.):

$$\text{Accuracy} = \frac{\#TP + \#FP}{\#TP + \#FP + \#TN + \#FN}. \quad (2.2)$$

The problem with accuracy is that it doesn't tell us everything about the results, since it turns four numbers into just one. There are two complementary pairs of measurements that can help us to interpret the performance of a classifier, namely sensitivity and specificity, and precision and recall. Their definitions are shown next, followed by some explanation.

$$\text{Sensitivity} = \frac{\#TP}{\#TP + \#FN} \quad (2.3)$$

$$\text{Specificity} = \frac{\#TN}{\#TN + \#FP} \quad (2.4)$$

$$\text{Precision} = \frac{\#TP}{\#TP + \#FP} \quad (2.5)$$

$$\text{Recall} = \frac{\#TP}{\#TP + \#FN} \quad (2.6)$$

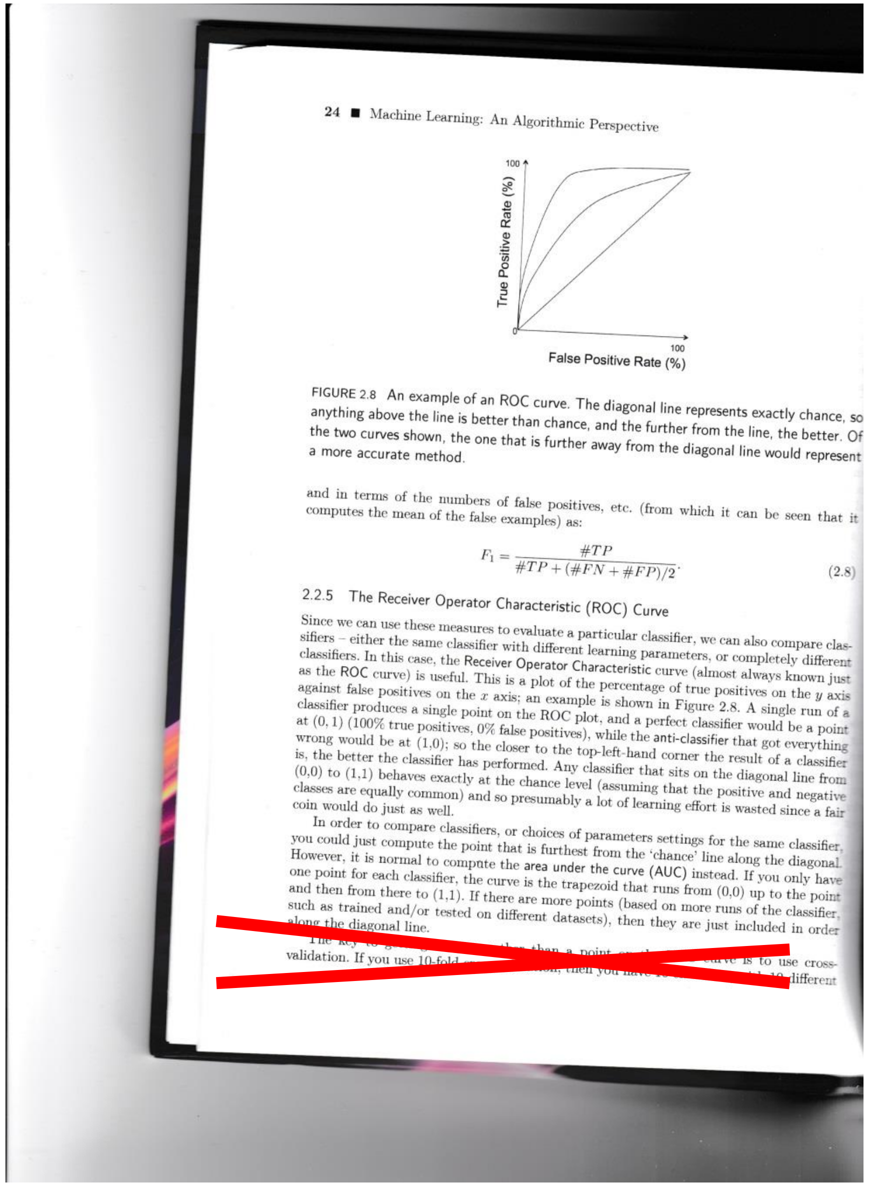
~~Sensitivity (also known as the true positive rate) is the ratio of the number of correct positive examples [illegible] the number classified as positive, while specificity is [illegible] ratio for negative examples. Precision is the ratio of correct positive [illegible] examples to the number of actual positive examples, while recall is the [illegible] number of correct positive examples out of those that were classified [illegible] positive, which is the [illegible] sensitivity. If you look at the chart again [illegible] see that sensitivity and specificity sum the [illegible] for the denominator, while precision and recall sum the first column and the first row, and so miss~~ some information about how well the learner does on the negative examples.

Together, either of these pairs of measures gives more information than just the accuracy. If you consider precision and recall, then you can see that they are to some extent inversely related, in that if the number of false positives increases (meaning that the algorithm is using a broader definition of that class), then the number of false negatives often decreases, and vice versa. They can be combined to give a single measure, the F_1 measure, which can be written in terms of precision and recall as:

$$F_1 = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (2.7)$$

2.2.5 ROC curves

- We will not consider this section
- But in case you are interested in
  - Precision-recall curves, or
  - ROC-curves
- Look somewhere else.
- You do not use cross-validation for constructing them



100
True Positive Rate (%)
0
100
False Positive Rate (%)

24 ■ Machine Learning: An Algorithmic Perspective

FIGURE 2.8 An example of an ROC curve. The diagonal line represents exactly chance, so anything above the line is better than chance, and the further from the line, the better. Of the two curves shown, the one that is further away from the diagonal line would represent a more accurate method.

and in terms of the numbers of false positives, etc. (from which it can be seen that it computes the mean of the false examples) as:

$$F_1 = \frac{\#TP}{\#TP + (\#FN + \#FP)/2}. \tag{2.8}$$

2.2.5 The Receiver Operator Characteristic (ROC) Curve

Since we can use these measures to evaluate a particular classifier, we can also compare classifiers – either the same classifier with different learning parameters, or completely different classifiers. In this case, the Receiver Operator Characteristic curve (almost always known just as the ROC curve) is useful. This is a plot of the percentage of true positives on the y axis against false positives on the x axis; an example is shown in Figure 2.8. A single run of a classifier produces a single point on the ROC plot, and a perfect classifier would be a point at (0, 1) (100% true positives, 0% false positives), while the anti-classifier that got everything wrong would be at (1,0); so the closer to the top-left-hand corner the result of a classifier is, the better the classifier has performed. Any classifier that sits on the diagonal line from (0,0) to (1,1) behaves exactly at the chance level (assuming that the positive and negative classes are equally common) and so presumably a lot of learning effort is wasted since a fair coin would do just as well.

In order to compare classifiers, or choices of parameters settings for the same classifier, you could just compute the point that is furthest from the 'chance' line along the diagonal. However, it is normal to compute the area under the curve (AUC) instead. If you only have one point for each classifier, the curve is the trapezoid that runs from (0,0) up to the point and then from there to (1,1). If there are more points (based on more runs of the classifier, such as trained and/or tested on different datasets), then they are just included in order along the diagonal line.

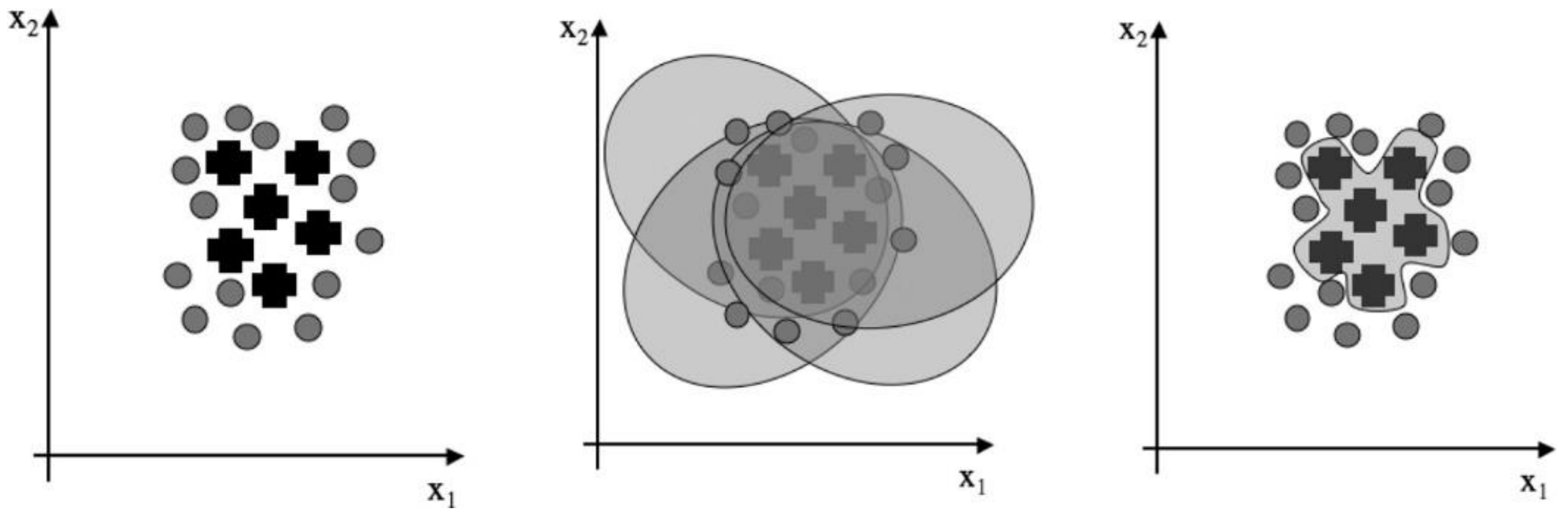
~~The key to~~ [illegible] ~~than a point~~ [illegible] ~~curve is to use cross-validation. If you use 10-fold~~ [illegible] ~~then you have~~ [illegible] ~~different~~

Example

- Notebook: boston_regularization_cv

Ensemble Learning, Random Forest

Voting classifiers



- Train several different classifiers
  - E.g. A LogReg
  - One or more *k*NNs
  - An SVM, etc.
- For prediction:
  - Run all the classifiers
  - Pick the majority vote (Hard vote)
  - Or the average if they provide probabilities (Soft vote)

Observation

- The ensemble classifier may perform better than any of the individual classifiers.
- Even weak classifiers (accuracies just above 0.5) may perform well together, provided
  - They are independent
  - There are sufficiently many of them

Bagging (Bootstrap Aggregating)

- An alternative to different learners:
- Train the same learner on slightly different data
- With much training data, you may train on subsets of the data
- One way to produce different datasets is called *bootstrap.*

Bootstrap:

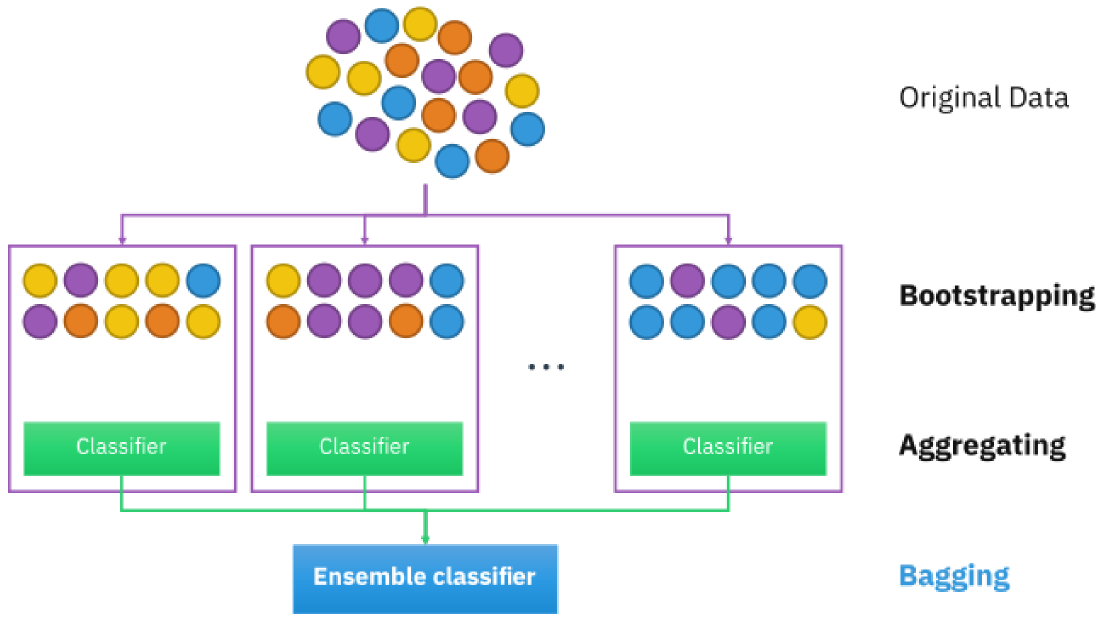
- Given a set of training data $D = (\boldsymbol{X}, \boldsymbol{t})$ of size *n*.
- Produce a new set by picking *n* samples from *D* with replacement.
- Produce several such sets, at least 50.
- Fit a model to each set.
- Prediction: use hard or soft vote.
- That's it!

More on bagging

- Subagging:
  - Similar, but pick sets of smaller size than *D* (all of same size)
- Pasting:
  - Similar but sampling without replacement.

=========================

- By the way:
  - Bootstrap may also be useful in evaluation.



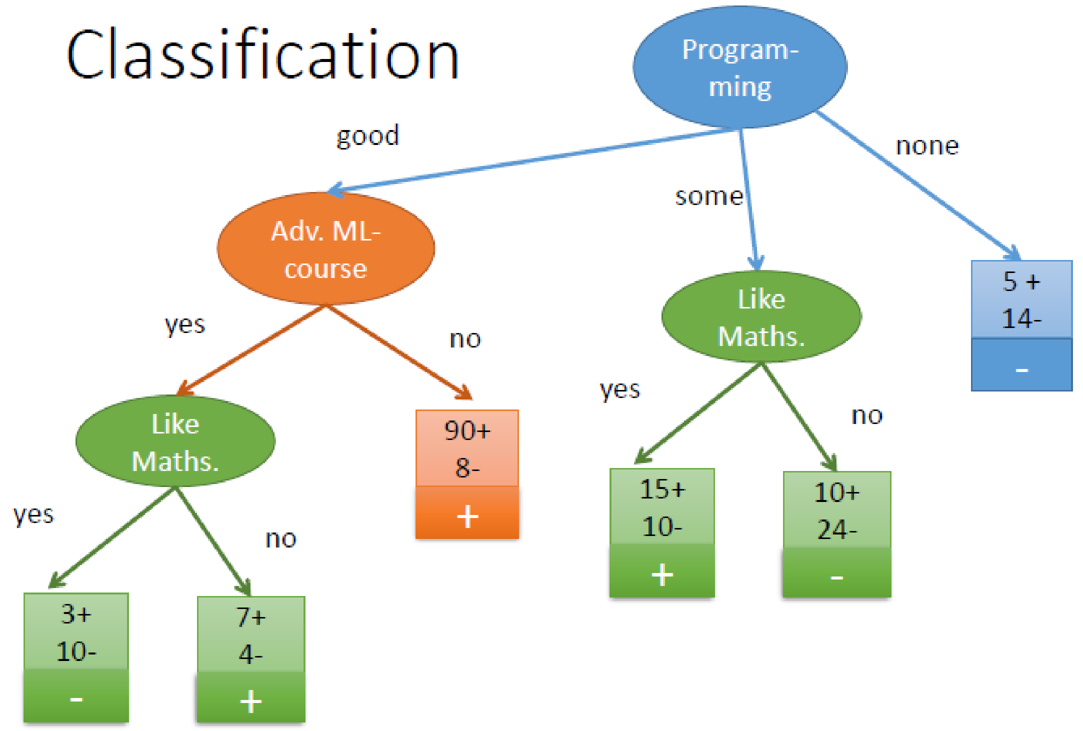
https://en.wikipedia.org/wiki/Bootstrap_aggregating

Back to decision trees

- Remember? (lecture 1)
- We used simplified criterion for selecting the stumps
  - Does not matter for the rest of the construction

==================

- Bias – variance:
- Stop the trees from growing:
  - More bias
  - Less variance



Random forest

Randomness, step 1

- Given training data $D = (\boldsymbol{X}, \boldsymbol{t})$:
- Use Bootstrap to construct sets of training data

=====================

- Train a decision tree on each bootstrap sample

Randomness, step 2

- At each step in the construction of the tree, consider only a subset of the available features
- In the algorithm:
  - a parameter *m*
  - *= the number of features to consider at each step*

Random forest - properties

- Good results both on big and small datasets
- Embarrassingly parallel

The 5+ lectures on supervised learning

Understanding

Hopefully

- You understand basic algorithms including neural networks, s.t.
  - You can choose between them
  - You can choose hyper-parameter values and tune them
- You have got a foundation for further studies in deep learning

Practical skills

- You can set up and run supervised ML experiments
- You can evaluate them