Model selection

(and intro to learning theory)

15-381 lecture notes, 16 March 2000

David Cohn <www.cs.cmu.edu/~cohn>

After this lecture, you should:

- understand test set pruning of decision trees
- understand χ^2 pruning of decision trees
- be able to say "probably approximately correct" without giggling
  - understand what it means
- be able to say "Vapnik Chervonenkis dimension" without stuttering
  - understand what it measures
- understand what a nested concept class is
- be able to recite the differences and advantages/disadvantages of
  - test set validation
  - k-fold crossvalidation
  - leave-one-out crossvalidation

How good is our predictor?

- How well does it explain data we've already seen?
- How well does it predict future (test) data?
  - These two are not always closely related
- Example
  - each day, look at weather, pick up one of a couple of dice, roll it and predict the result

x1	mon	sun	6-side	1
x2	tues	rain	10-side	9
x3	weds	sun	6-side	1
x4	thurs	tornado	6-side	5
x5	fri	sun	10-side	4
x6	sat	rain	6-side	2
x7	sun	rain	10-side	8
x8	mon	sun	10-side	6
x9	tues	tornado	6-side	5
x10	weds	sun	6-side	3

  - can build decision tree that perfectly explain data
  - how well will it predict future rolls of the die?

- What *would* we like the decision tree to look like?

Improving decision tree performance

- Assessing the performance of our decision tree*
  - split available examples into a training set and a test set
  - build a decision tree based on the training set
  - evaluate performance on the test set

*Based on some common assumptions of machine learning
  - data are drawn at random from an "independent, identically-distributed" (IID) distribution
  - test data come from same distribution as training data

- Pruning with a test set (also called "early stopping")
  - 0) split data at random into a training set and test set
  - 1) begin with decision tree with no splits
  - 2) identify Bayes optimal ("majority") prediction at current node
  - 3) compute accuracy of Bayes optimal prediction on training set and on testing set
  - 4) identify best split, according to information gain *on training set*
  - 5) using the Bayes optimal prediction at each leaf node, compute accuracy on training sets and test sets
    - note that training accuracy should always go up
  - 6) if test accuracy goes up, go to (4) and repeat

- [Need example for both this and stats pruning!]

Improving decision tree performance

- Pruning with statistical tests
  - the *null hypothesis*: what if there's no real pattern underlying the data?
    - what distribution of positive and negative examples would we expect to see?
    - have we seen enough data, and enough of a pattern to have confidence that the null hypothesis is wrong?
  - in parent node, we've seen p positive and n negative examples
    - if split is made on attribute with **no predictive power**, expected number of positive and negative examples in child node i are

$$\hat{p}_i = p \cdot \frac{p_i + n_i}{p+n} \qquad \hat{n}_i = n \cdot \frac{p_i + n_i}{p+n}$$

    - in practice, we get p_i and n_i - measure the deviation of observed p_i and n_i from their expected values:

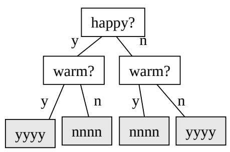
$$D = \sum_i \frac{(p_i - \hat{p}_i)^2}{\hat{p}_i} + \frac{(n_i - \hat{n}_i)^2}{\hat{n}_i}$$

    - what's the probability that we'd see this deviation given v samples if the null hypothesis were true?

- Under null hypothesis, D is distributed according to χ^2 statistic with $n+p-1$ degrees of freedom

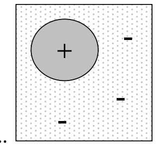
Improving decision tree performance

- Under null hypothesis, *D* is distributed according to χ^2 statistic with *v-1* degrees of freedom
  - what's the probability that we'd get a deviation as large as observed *D* if the null hypothesis were true?
  - does this give us confidence that we're splitting on something other than random fluctuations?
- Decision tree pruning with χ^2 pruning:
  - 1) build full decision tree out to leaves
  - 2) pick a bottom level split at random and compute its χ^2 statistic. If there is less than confidence X (e.g. 90%) that the null hypothesis is false, prune the split and merge leaves
  - 3) go to (2)
- Why build out then prune back instead of just stopping when no confidence?
  - an example of when early stopping does the wrong thing:



What's going on? - a little learning theory

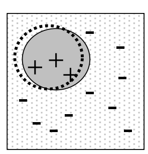
- Concept classes
  - **concept**: a subset of points *c* in input space
    - if a point is *in* the concept, it is a positive example (+), otherwise it is a negative example (-)
  - **concept class**: a set of concepts, e.g.
    - all sets separable by circles
    - all sets separable by hyperplanes
  - **nested concept class**: a set of concept classes parameterized by *i* such that $C_1 \subseteq C_2 \subseteq C_3 \subseteq ...$



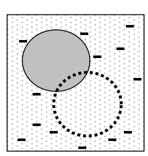
- Example concept classes
  - a hyperplane
    - nested classes: a hyperplane in *i* dimensions, intersection/union of *i* hyperplanes...
  - a decision tree
    - nested classes: decision trees with *i* splits
- Learning theory poses problem as follows:
  - given a concept class *C* and examples of randomly drawn data labeled according to some unknown $c \in C$, can we guess the correct *c*, or at least one that's reasonably close?
  - how many examples do we need to see to have confidence that we've guessed the correct *c* (or one that's reasonably close)?
  - *assume that there is no noise, so it is always possible to find a c' that is consistent with all data we've seen*

Learning theory - PAC learning

- Can we find the "right" concept?
  - in general, no
  - how close is close enough?
  - measure in terms of expected performance:
    - ε-**approximation:** given a randomly drawn future example, probability of disagreement between *c* and our guess *c'* is less than ε (written $error(c,c')<\varepsilon$)



- Can we always get close?
  - in general, no - if data are drawn at random there's always some chance that we'll get a horribly useless training set
  - measure in terms of **probability** δ that we'll get an unfortunate set of examples that doesn't let us find a good (within ε) approximation



- The more examples we have, the more likely we are to get a good approximation - how many do we need to be "probably approximately correct"?
  - i.e., how many do we need such that with probability $1-\delta$, $error(c,c')<\varepsilon$?

PAC learning

- Given unknown $c \in C$, assume existence of a "bad" concept c_{bad}
  - c_{bad} is consistent with all *m* training examples we've seen, but
  - c_{bad} is differs from *c* by more than ε
- The probability c_{bad} agrees with *c*
  - on some random example is $\leq (1-\varepsilon)$
  - on *m* random examples is $\leq (1-\varepsilon)^m$
- Can be no more than $|C|$ bad hypotheses in *C*
  - so probability that there exists a bad hypothesis in *C* consistent with all *m* training examples is $\leq |C|(1-\varepsilon)^m$
- Want probability of a bad hypothesis surviving to be $\leq \delta$
  - set $|C|(1-\varepsilon)^m \leq \delta$, and solve for *m:*

$$m \geq \frac{1}{\varepsilon}\left(\ln\frac{1}{\delta} + \ln|C|\right)$$

- What does this tell us?
  - the greater our desired confidence, and the less our tolerance for error, the more examples we need
  - the bigger our hypothesis class, the more examples we need

PAC learning

- In order to PAC learn, we need at least

$$m \geq \frac{1}{\varepsilon}\left(\ln\frac{1}{\delta} + \ln|C|\right)$$

- Let's say we have a decision tree with n binary attributes
  - is it a concept class? (sure!)
  - what's $|C|$?
  - how many examples will we need to PAC learn?
- Now limit the decision tree to having no more than i splits
  - what's $|C|$?
  - how many examples will we need to PAC learn?
  - *what about if we have i+1 splits?*
- These are nested concept classes
  - if we can fit the training data equally well with C_i and C_{i+1}, learning theory tells us that we will have greater confidence in predictions from the smaller class
  - mathematical justification for Ockham's razor!

PAC learning - "But what if...?"

- In order to PAC learn, we need at least

$$m \geq \frac{1}{\varepsilon}\left(\ln\frac{1}{\delta} + \ln|C|\right)$$

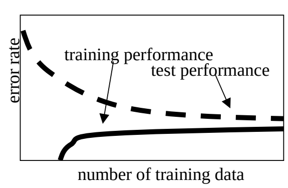
- *But* - what if we're fitting a hyperplane in i continuous dimensions - *what's $|C|$?*
  - urk. ummm....
  - fortunately a version of $|C|$ for continuous concept classes
  - big fancy name: the *Vapnik-Chervonenkis dimension "d"*
  - indicates the expressiveness of an arbitrary concept class

$$m \geq \max\left(\frac{1-\varepsilon}{\varepsilon}\ln\frac{1}{\delta}, \frac{8d}{\varepsilon}\log\frac{13}{\varepsilon}\right) \quad \text{(Blumer et al '89)}$$

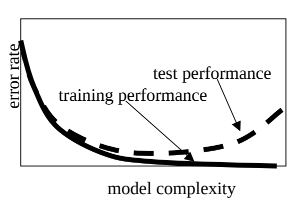
- *But* - what if we've got noise, if there are no concepts consistent with the data?
  - version of the computation that bounds difference between training and test performance: given VC-dimension d, sample size m, and training error $error_{train}$, can compute probability δ that $error_{test} > error_{train} + \varepsilon$
  - *messy, messy formulation*
  - *"worst case" bounds - usually orders of magnitude off from actually observed performance*

Learning theory and model selection

- For a given concept class, more training data generally
  - increases training error
  - decreases test error

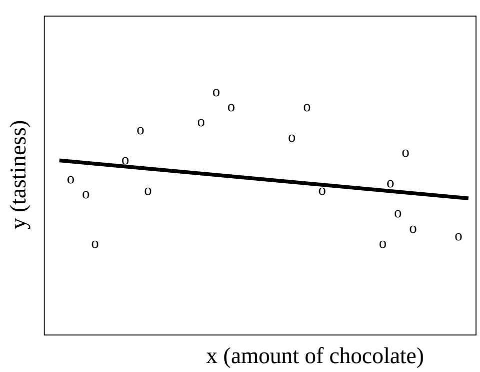


- Often get to choose between concept classes ("models") with different expressive power
  - decision trees with *1,2,3,...,n* splits
  - hyperplanes in *1,2,3...,n* dimensions
  - neural networks with *1,2,3...,n* units
- Selecting larger concept with more expressive power does two things
  - allows us to fit training data better
    - (decrease training error)
  - decreases confidence that training performance will reflect test performance



Another model selection example: regression

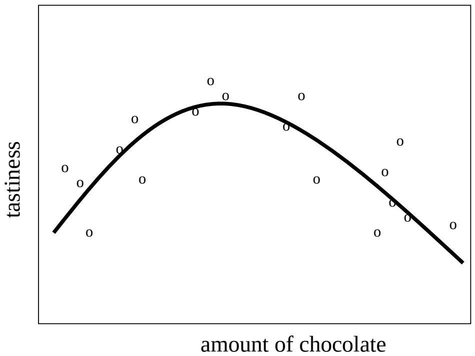
- Given real-valued inputs x and outputs y, predict y from x
  - also called "function fitting" and "regression"
- One approach - linear regression:



- Error on training set is pretty high
- Simple (low VC dimension)
- How well do we think it will predict?

Another model selection example: regression

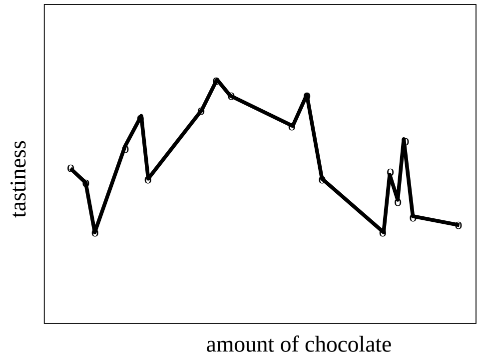
- Another approach - 2nd order (quadratic) regression



- Error on training set is a lot lower
- More complex model (higher VC dimension)
- How well do we think it will predict?

Another model selection example: regression

- Yet another approach - piecewise linear fitting
  - aka - "connect the dots"

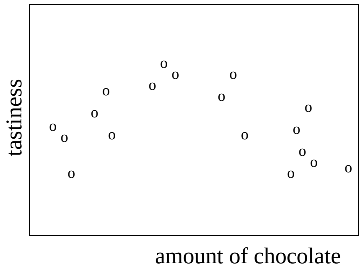


- Error on training set is zero!
- Much more complex model (higher VC dimension)
- How well do we think it will predict?

- Of these three models - which is best?
  - How do we get a computer to decide?

What we really want

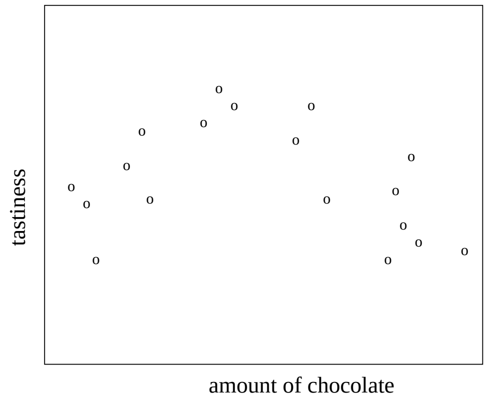
- How well are we going to do at predicting future data drawn from the same distribution?



- Same as model selection problem for classification!
- A few different approaches
  - pure theory
    - VC-dimension
    - Akaike's Information Criterion (AIC)
    - Bayesian Information Criterion (BIC)
    - Minimum Description Length (MDL)
  - empirical validation
    - test set validation
    - k-fold crossvalidation
    - leave-one-out crossvalidation

Test set validation

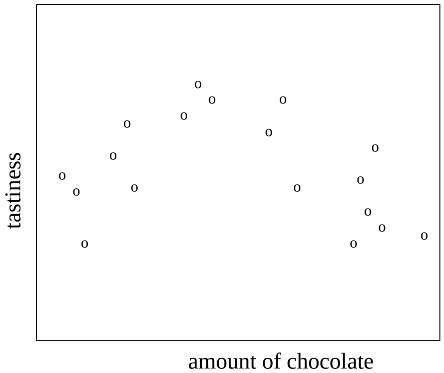
- Randomly choose 25% of data to be used as test set
- Fit model to remaining 75% of data
- Evaluate error of model on test set
- Repeat for each candidate model



- How did we choose 25%?
  - What if we used 90%?
  - What if we used 0.01%?

K-fold crossvalidation

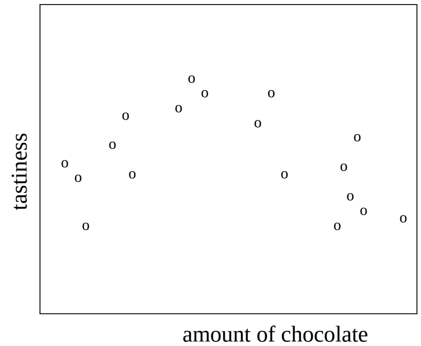
- Divide data into k equal-sized partitions
- For each partition, train on k-1 partitions, test on remaining partition
- Average error from k test partitions
- Repeat for each candidate model



- Advantages over test set validation
  - actually using more of our data for training
  - less dependence on getting a single bad split
- Disadvantages
  - requires k times as much effort

The logical limit: leave-one-out crossvalidation

- Remove first point from training set, train on all *m*-1 points
- Evaluate error on first point
- Put first point back, remove second point...
- Repeat for all *m* points
- Average error from all *m* points
- Repeat for each candidate model



- Advantages over test set validation
  - using almost all of our data for training
- Disadvantages
  - requires *m* times as much effort
  - can run into sneaky problems when data aren't truly IID