COMP 551 - Applied Machine Learning
Lecture 8 – Feature construction

William L. Hamilton

(with slides and content from Joelle Pineau and Jackie Cheung)

Upcoming tutorials

- Tutorial 2 on SciKit learn + Intelligent tutor
  - Session 1: Feb 6th (i.e., tomorrow)
    - STBIO S1/4, 6-8pm
  - Session 2: Feb 8th (i.e., this Friday)
    - Adams Aud, 6-8pm
- First half will be a standard tutorial, but the second half will be a demo of a AI-based tutoring system developed by a Mila student, Iulian Serban (www.iulianserban.com)

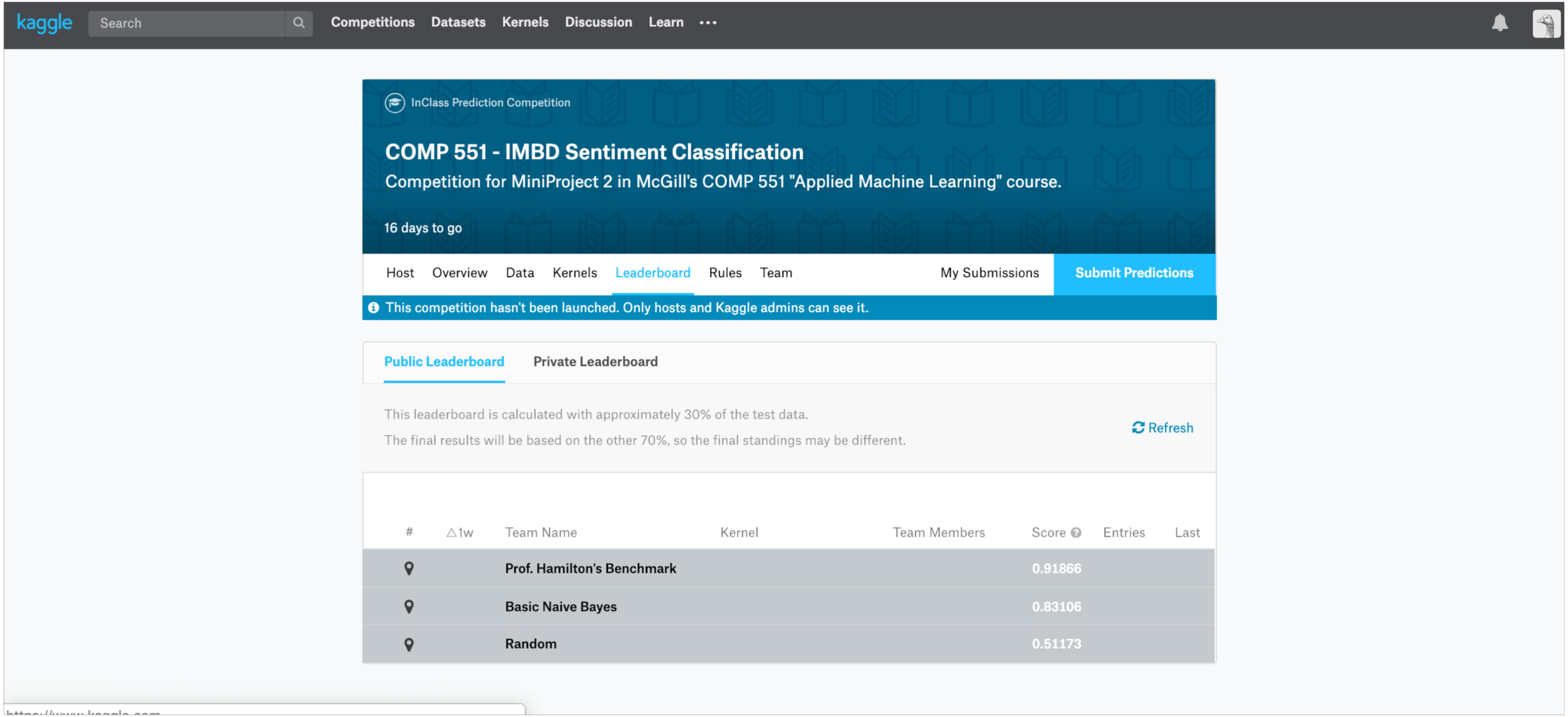
Office hours

- This week (Feb 4-8) I will have my office hours **Wednesday (Feb 6) from 9-11am.**
- Usual location: MC 309
- Apologies for the inconvience!

MiniProject 2

- Details are now available: https://cs.mcgill.ca/~wlh/comp551/files/miniproject2_spec.pdf
- Due Feb. 22nd at 11:59pm.

MiniProject 2



Steps to solving a supervised learning problem

1. Decide what the input-output pairs are.

With a focus on feature extraction for language problems!

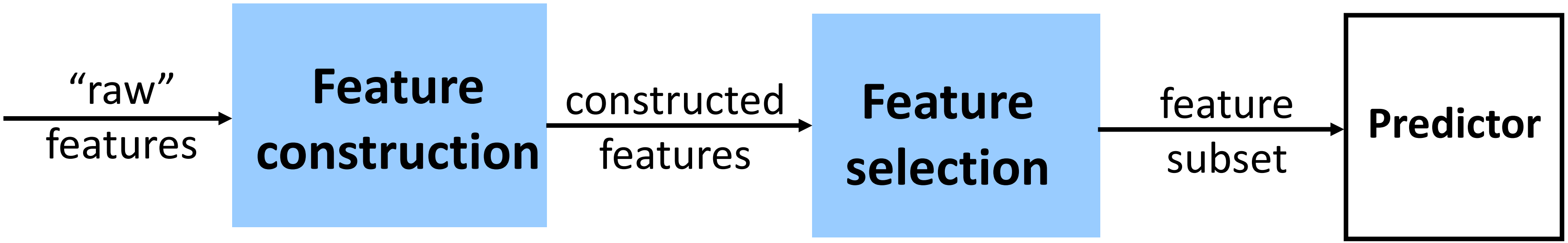
2. Decide how to encode inputs and outputs.
   - This defines the input space X and output space Y.

Today: deciding on what the inputs are

3. Choose a class of hypotheses / representations H.
   - E.g. linear functions.
4. Choose an error function (cost function) to define best hypothesis.
   - E.g. Least-mean squares.
5. Choose an algorithm for searching through space of hypotheses.

So far: we have been focusing on this

Feature extraction steps



A few strategies

- Use domain knowledge to construct "ad hoc" features.
- Normalization across different features, e.g. centering and scaling with $x_i = (x'_i - \mu_i) / \sigma_i$.
- Normalization across different data instances, e.g. counts/histogram of pixel colors.
- Non-linear expansions when first order interactions are not enough for good results, e.g. products x_1x_2, x_1x_3, etc.
- Other functions of features (e.g. sin, cos, log, exponential etc.)
- Regularization (lasso, ridge).

Feature construction

Why do we do feature construction?

- Increase predictor performance.
- Reduce time / memory requirements.
- Improve interpretability.

But: Don’t lose important information!

Problem: we may end up with lots of possibly irrelevant, noisy, redundant features. (Here, “noisy” is in the sense that it can lead the predictor astray.)

Applications with lots of features

- Any kind of task involving images or videos - object recognition, face recognition. Lots of pixels!

- Classifying from gene expression data. Lots of different genes!
  - Number of data examples: 100
  - Number of variables: 6000 to 60,000

- Natural language processing tasks. Lots of possible words!

Features for modelling natural language

- Words
- TF-IDF
- N-grams
- Syntactic features
- Word embeddings (not in this lecture…)
- Useful Python package for implementing these: http://www.nltk.org/

Words

- Binary (present or absent)
- Absolute frequency
  - i.e., raw count
- Relative frequency
  - i.e., proportion
  - document length

Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.

Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

Stopword List

for
is
of
the
to

Multinomial vs Bernoulli Naïve Bayes

- Note that using binary vs. count features can have implications for your model!
- In Lecture 5 we discussed the Bernoulli (i.e., binary) Naïve Bayes:

$$P(\mathbf{x}|y=k) = \prod_{j=1}^{m} \theta_{j,k}^{x_j}(1-\theta_{j,k})^{(1-x_j)} \longleftarrow \text{Bernoulli distribution}$$

  - Assumes the features are binary counts.
- But we can also have a Multinomial Naïve Bayes:

$$P(\mathbf{x}|y=k) = \frac{(\sum_{j=1}^{m} x_j)!}{\prod_{j=1}^{m} x_j!} \prod_{j=1}^{m} \theta_{j,k}^{x_j} \longleftarrow \text{Multinomial distribution}$$

  - Assumes the features are **integer** counts.

Homework: what is the maximum likelihood estimate for the multinomial Naïve Bayes?

More options for words

- Stopwords
  - Common words like “the”, “of”, “about” are unlikely to be informative about the contents of a document.
  - Standard practice to remove them, though they can be useful (e.g., for authorship identification)
- Lemmatization
  - Inflectional morphology: changes to a word required by the grammar of a language
    - e.g., “perplexing” “perplexed” “perplexes”
    - (Much worse in languages other than English, Chinese, Vietnamese)
  - **Lemmatize** to recover the canonical form; e.g., “perplex”

Term weighting

- Not all words are equally important.
- What do you know about an article if it contains the word

the?

penguin?

TF*IDF (Salton, 1988)

- Term Frequency Times Inverse Document Frequency
- A term is important/indicative of a document if it:
    1. Appears many times in the document
    2. Is a relative rare word overall
- TF is usually just the count of the word
- IDF is a little more complicated:
    - $IDF(t, Corpus) = \log \frac{\#(\text{Docs in Corpus})}{\#(\text{Docs with term t}) + 1}$
  - Can use a separate large training corpus for this
- Originally designed for document retrieval

N-grams

- Use sequences of words, instead of individual words
- e.g., *… quick brown fox jumped …*
  - Unigrams (i.e. words)
    - quick, brown, fox, jumped
  - Bigrams
    - quick_brown, brown_fox, fox_jumped
  - Trigrams
    - quick_brown_fox, brown_fox_jumped
- Usually stop at N <= 3, unless you have lots and lots of data

Rich linguistic features

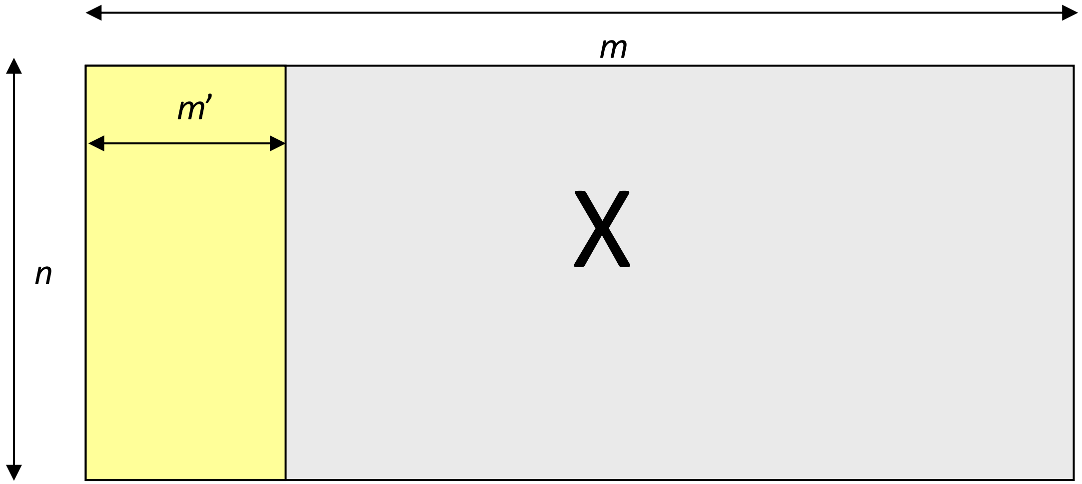
- Syntactic
  - Extract features from a parse tree of a sentence
  - [SUBJ The chicken] [VERB crossed] [OBJ the road].
- Semantic
  - e.g., Extract the semantic roles in a sentence
  - [AGENT The chicken] [VERB crossed] [LOC the road].
  - e.g., Features are synonym clusters ("chicken" and "fowl" are the same feature) → WordNet
- **Trade-off**: Rich, descriptive features might be more discriminative, but are hard (expensive, noisy) to get!

Sentiment and affective lexicons

- Words have different connotations and associated meanings.
  - “hate” vs. “love”
- There are collections of “sentiment lexicons”, which map words to different sentiment values. E.g.,
  - Bing Liu’s “Opinion Lexicon” (6000 positive/negative words)
  - Warriner et al’s “Affective Rating Dataset” (continuous scores for 14,000 words)
    - Ratings for **valence** (”how positive vs. negative”) and **arousal** (“how “strong”).
- Other types of lexicons exist. E.g.,
  - LIWC contains various categories (“anger”, “cognitive processes”)
  - Brysbaert et al’s “concreteness lexicon”

Feature selection

- Thousands to millions of low level features: select the most relevant one to build better, faster, and easier to understand learning machines.



Feature selection techniques

- Principal Component Analysis (PCA)
  - Also called (Truncated) Singular Value Decomposition, or Latent Semantic Indexing in NLP
- Variable Ranking
  - Think of features as random variables
  - Find how strong they are associated with the output prediction, remove the ones that are not highly associated, either before training and during training
- Representation learning techniques [future lectures]

Principal Component Analysis (PCA)

- **Idea:** Project data into a lower-dimensional sub-space, $R^m \dashrightarrow R^{m'}$, where $m' < m$.

Principal Component Analysis (PCA)

- **Idea:** Project data into a lower-dimensional sub-space, $R^m \dashrightarrow R^{m'}$, where $m' < m$.

- Consider a linear mapping, $x_i \dashrightarrow W^T x_i$
  - W is the compression matrix with dimension $R^{m \times m'}$.
  - Assume there is a decompression matrix $U^{m' \times m}$.

Principal Component Analysis (PCA)

- **Idea:** Project data into a lower-dimensional sub-space, $R^m \dashrightarrow R^{m'}$, where $m' < m$.

- Consider a linear mapping, $x_i \dashrightarrow W^T x_i$
  - W is the compression matrix with dimension $R^{m \times m'}$.
  - Assume there is a decompression matrix $U^{m' \times m}$.

- Solve the following problem: $\text{argmin}_{W,U} \Sigma_{i=1:n} || x_i - UW^T x_i ||^2$

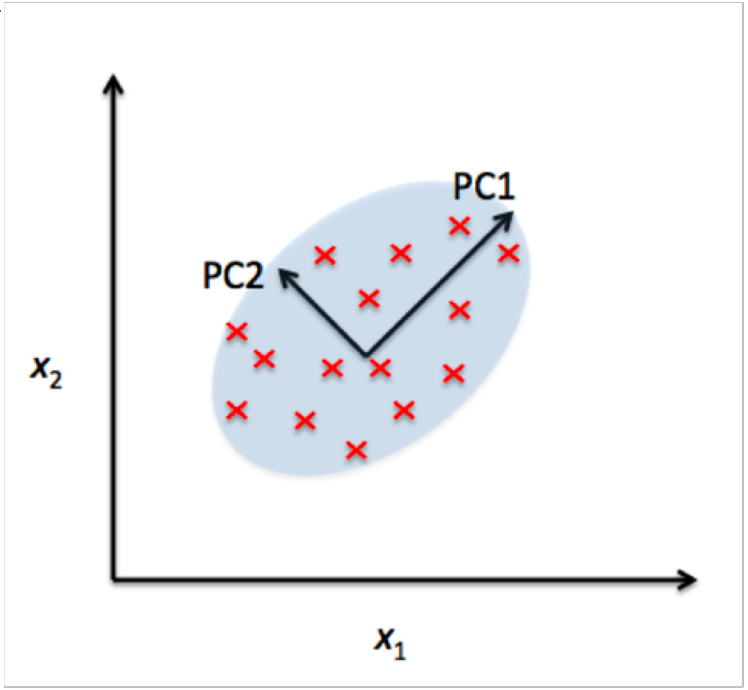
Principal Component Analysis (PCA)

- Solve the following problem: $\text{argmin}_{W,U} \Sigma_{i=1:n} || x_i - UW^T x_i ||^2$
- Equivalently: $\text{argmin}_{W,U} || X - XWU^T ||^2$

- Solution is given by eigen-decomposition of X^TX.
  - W is mxm' matrix corresponding to the first m' eigenvectors of X^TX (sorted in descending order by the magnitude of the eigenvalue).
  - Equivalently: W is mxm' matrix containing the first m' left singular vectors of X
  - **Note:** The columns of W are orthogonal!

Principal Component Analysis (PCA)

- $\mathbf{W}$ is given by eigen-decomposition of $\mathbf{X}^T\mathbf{X}$
- Select the project dimension, m', using cross-validation.
- Typically "center" the examples before applying PCA (i.e., subtract the mean).
- **Interpretation:** First column of $\mathbf{W}$ is direction with maximal variance in the data; second column is the orthogonal direction with second-most variance, etc.



Eigenfaces

- Turk & Pentland (1991) used PCA method to capture face images.
- Assume all faces are about the same size
- Represent each face image as a data vector.
- Each eigenvector is an image, called an **Eigenface**.

Average image

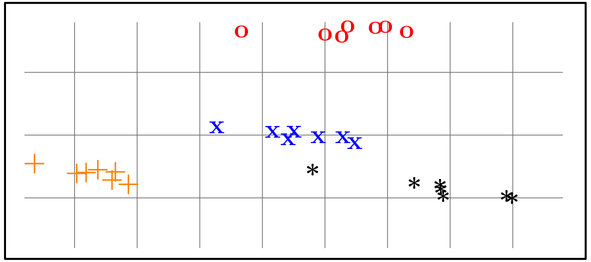
Eigenfaces

Training images

Original images in R^{50x50}.

Projection to R^{10} and reconstruction

Projection to R^2. Different marks Indicate different individuals.



Other feature selection methods

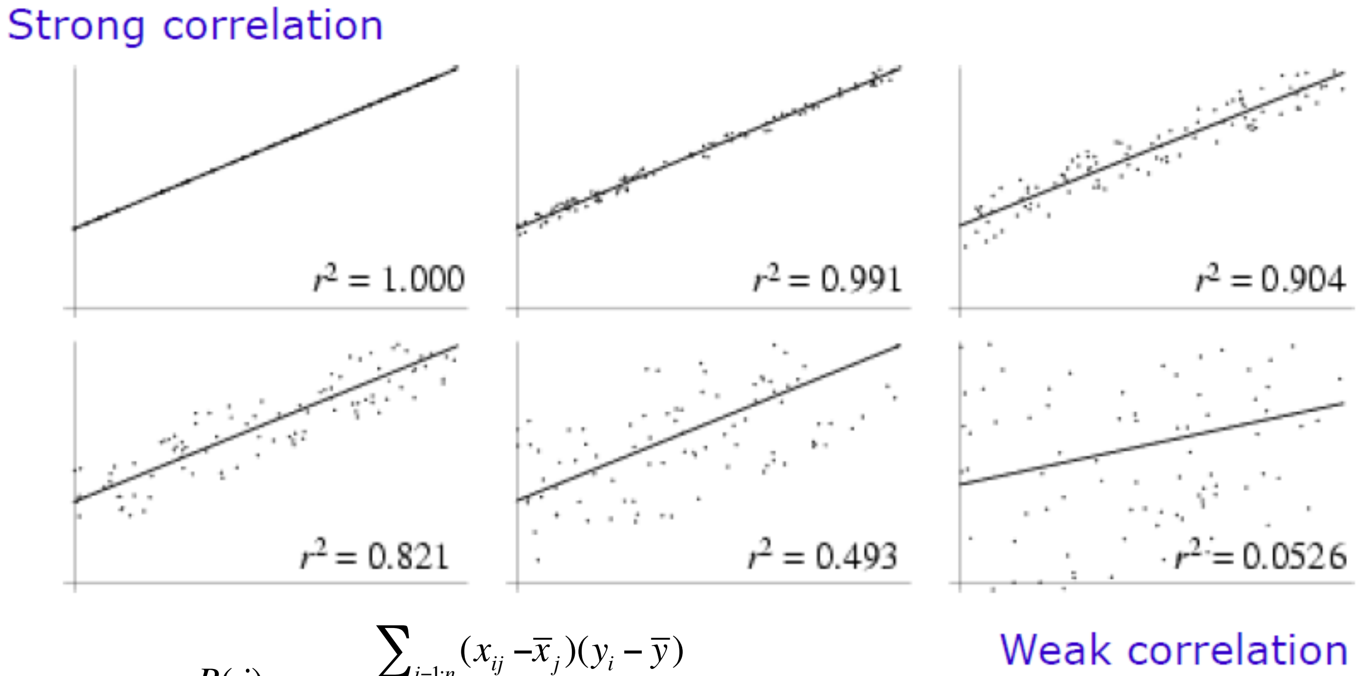
- The goal: Find the input representation that produces the best generalization error.

- Two classes of approaches:
  - Wrapper & Filter methods: Feature selection is applied as a pre-processing step.
  - Embedded methods: Feature selection is integrated in the learning (optimization) method, e.g. regularization.

Variable Ranking

- Idea: Rank features by a scoring function defined for individual features, independently of the context of others. Choose the m' highest ranked features.
- Pros / cons:
  - Need to select a scoring function.
  - Must select subset size (m'): cross-validation
  - Simple and fast – just need to compute a scoring function m times and sort m scores.

Scoring function: Correlation Criteria



$$R(j) = \frac{\sum_{i=1:n} (x_{ij} - \bar{x}_j)(y_i - \bar{y})}{\sqrt{\sum_{i=1:n} (x_{ij} - \bar{x}_j)^2 \sum_{i=1:n} (y_i - \bar{y})^2}}$$

slide by Michel Verleysen

Scoring function: Mutual information

- Think of $\mathbf{X_j}$ and Y as random variables.

- Mutual information between variable $\mathbf{X_j}$ and target Y:

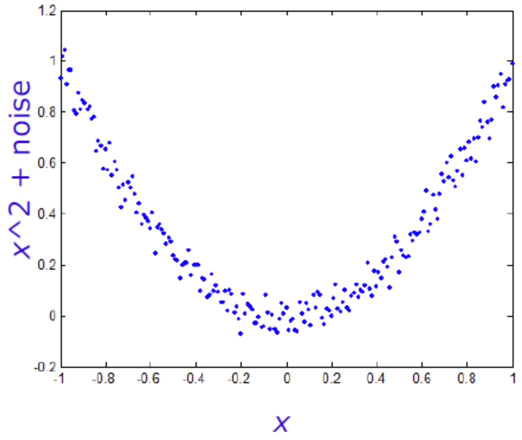
$$I(j) = \int_{X_j} \int_{Y} p(x_j, y) \log \frac{p(x_j, y)}{p(x_j) p(y)} dx\, dy$$

$$I(j) = \sum_{X_j} \sum_{Y} P(X_j = x_j, Y = y) \log \frac{p(X_j = x_j, Y = y)}{p(X_j = x_j) p(Y = y)}$$

- Empirical estimate from data (assume discretized variables):

Nonlinear dependencies with MI

- Mutual information identifies nonlinear relationships between variables.
- Example:
  - x uniformly distributed over [-1 1]
  - $y = x^2$ + noise
  - z uniformly distributed over [-1 1]
  - z and x are independent



1000 samples	y,y	x,y	z,y
Correlation	1	0.0460	0.0522
Mutual information	2.2582	1.1996	0.0030

Variable Ranking

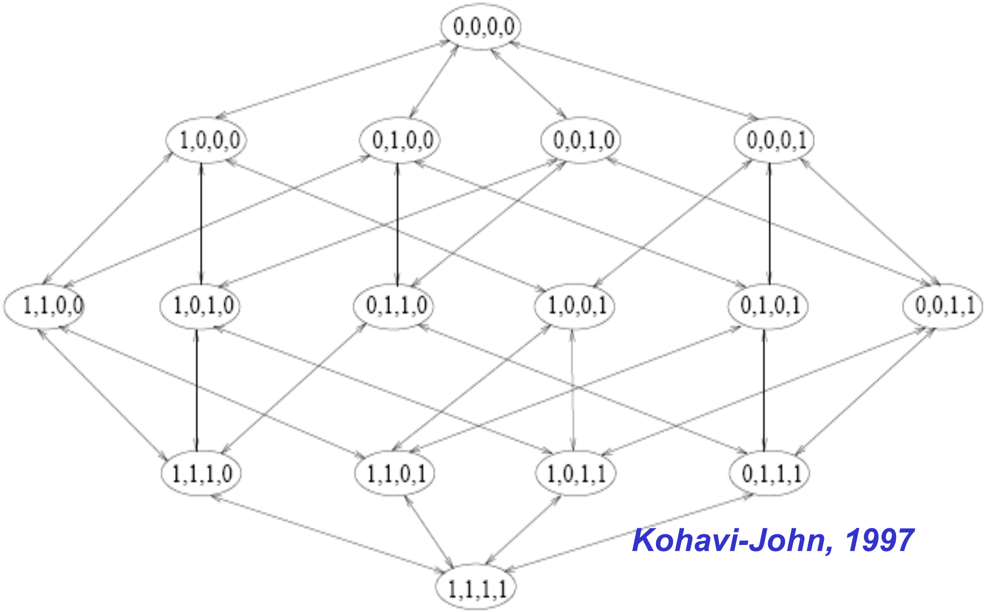
- Idea: Rank features by a scoring function defined for individual features, independently of the context of others. Choose the m' highest ranked features.
- Pros / cons:
  - Need to select a scoring function.
  - Must select subset size (m'): cross-validation
  - Simple and fast – just need to compute a scoring function m times and sort m scores.
  - Scoring function is defined for individual features (not subsets).

Best-Subset selection

- Idea: Consider all possible subsets of the features, measure performance on a validation set, and keep the subset with the best performance.

- Pros / cons?
  - We get the best model!
  - Very expensive to compute, since there is a combinatorial number of subsets.

Search space of subsets



Kohavi-John, 1997

n features, $2^n - 1$ possible feature subsets!

Subset selection in practice

- Formulate as a search problem, where the state is the feature set that is used, and search operators involve adding or removing feature set
  - Constructive methods like forward/backward search
  - Local search methods, genetic algorithms
- Use domain knowledge to help you group features together, to reduce size of search space
  - e.g., In NLP, group syntactic features together, semantic features, etc.

Regularization

- Idea: Modify the objective function to constrain the model choice. Typically adding term $(\sum_{j=1:m} w_j^{\,p})^{1/p}$.
  - Linear regression -> Ridge regression, Lasso
- Challenge: Need to adapt the optimization procedure (e.g. handle non-convex objective).
- Regularization can be viewed as a form of feature selection.

- This approach is often used for very large natural (non-constructed) feature sets, e.g. images, speech, text, video.

Steps to solving a supervised learning problem

1. Decide what the input-output pairs are.
2. Decide how to encode inputs and outputs.
   - This defines the input space X and output space Y.
3. Choose a class of hypotheses / representations H.
   - E.g. linear functions.
4. Choose an error function (cost function) to define best hypothesis.
   - E.g. Least-mean squares.
5. Choose an algorithm for searching through space of hypotheses.

Evaluate on validation set

(If doing k-fold cross-validation, re-do feature selection for each fold.)

Evaluate final model/hypothesis on test set

Final comments

Classic paper on this:

I. Guyon, A. Elisseeff, An introduction to Variable and Feature Selection. Journal of Machine Learning Research 3 (2003) pp.1157-1182

http://machinelearning.wustl.edu/mlpapers/paper_files/GuyonE03.pdf

(and references therein.)

More recently, move towards learning the features end-to-end, using neural network architectures (more on this later in the course).