COMP 551 – Applied Machine Learning
Lecture 9: A brief survey of methods for feature construction and selection

Instructor: Joelle Pineau (*jpineau@cs.mcgill.ca)*

Guest Lecturer: Ali Emami (*ali.emami@mail.mcgill.ca*)

Class web page: *www.cs.mcgill.ca/~jpineau/comp551*

Who am I?

- PhD Student of Jackie Cheung (CS Department)
- My research area: **natural language processing**
  - Computational semantics
    - Understanding how passages of text relate to each other; how they denote entities and events in the real world
  - Commonsense reasoning
    - NLP tasks that require a fair amount of commonsense reasoning
    - "John yelled at Kevin because he was so upset. Who was upset?"

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 hypot
   - 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.

Today: deciding on what the inputs are

With a focus on feature extraction for language problems!

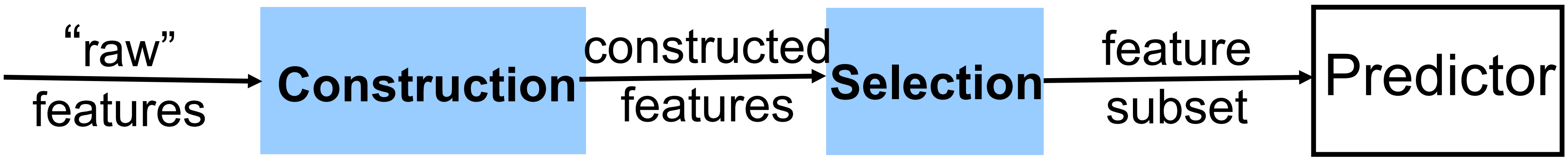
So far: we have been focusing on this

Examples of feature analysis?

- What do you know about feature selection?

    Methods, insights, creative ideas...

Feature Extraction Steps



Ideas for feature construction?

A few strategies we discussed

- 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
- Useful Python package for implementing these:
  - Natural Language toolkit: 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

More options for words

- Stopwords
  – Common words like “the”, “of”, “about” are unlikely to be informative about the contents of a document. Remove!
- Lemmatization
  – Inflectional morphology: changes to a word required by the grammar of a language
    - e.g., “perplex<u>ing</u>” “perplex<u>ed</u>” “perplex<u>es</u>”
    - (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:
  - $IDF(t, Corpus) = \log \frac{\#(\text{Docs in } Corpus)}{\#(\text{Docs with term } t) + 1}$
    - Need 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!

Word embedding models

- Problems with above:
  - Number of features scales with size of vocabulary!
  - Many words are semantically related and behave similarly (e.g., *freedom* vs *liberty*)
- Word embedding models can help us:
  - Embed each word into a fixed-dimension space
  - Learn correlations between words with similar meanings

word2vec (Mikolov et al., 2013)

- Intuition:
  - Words that appear in similar contexts should be semantically related, so they should have similar word vector representations
- Actually two models:
  - **Continuous bag of words (CBOW)** – use context words to predict a target word
  - **Skip-gram** – use target word to predict context words
- In both cases, the representation that is associated with the target word is the embedding that is learned.

word2vec Architectures

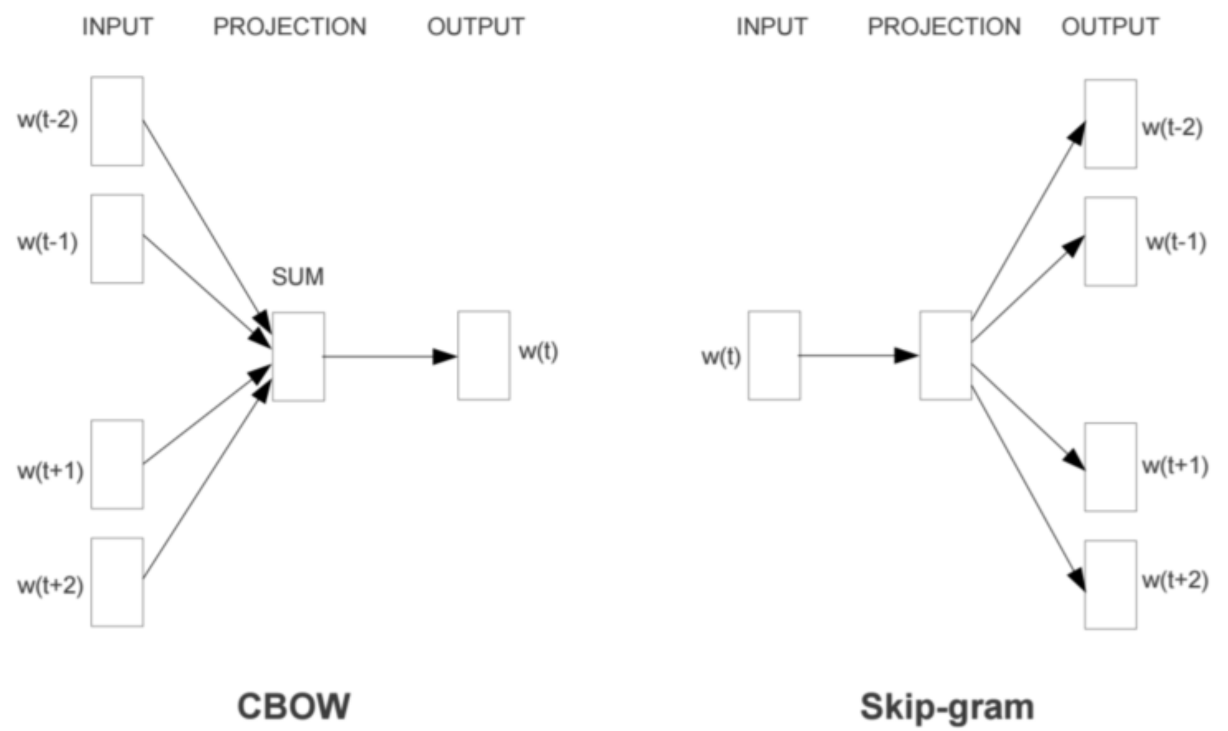


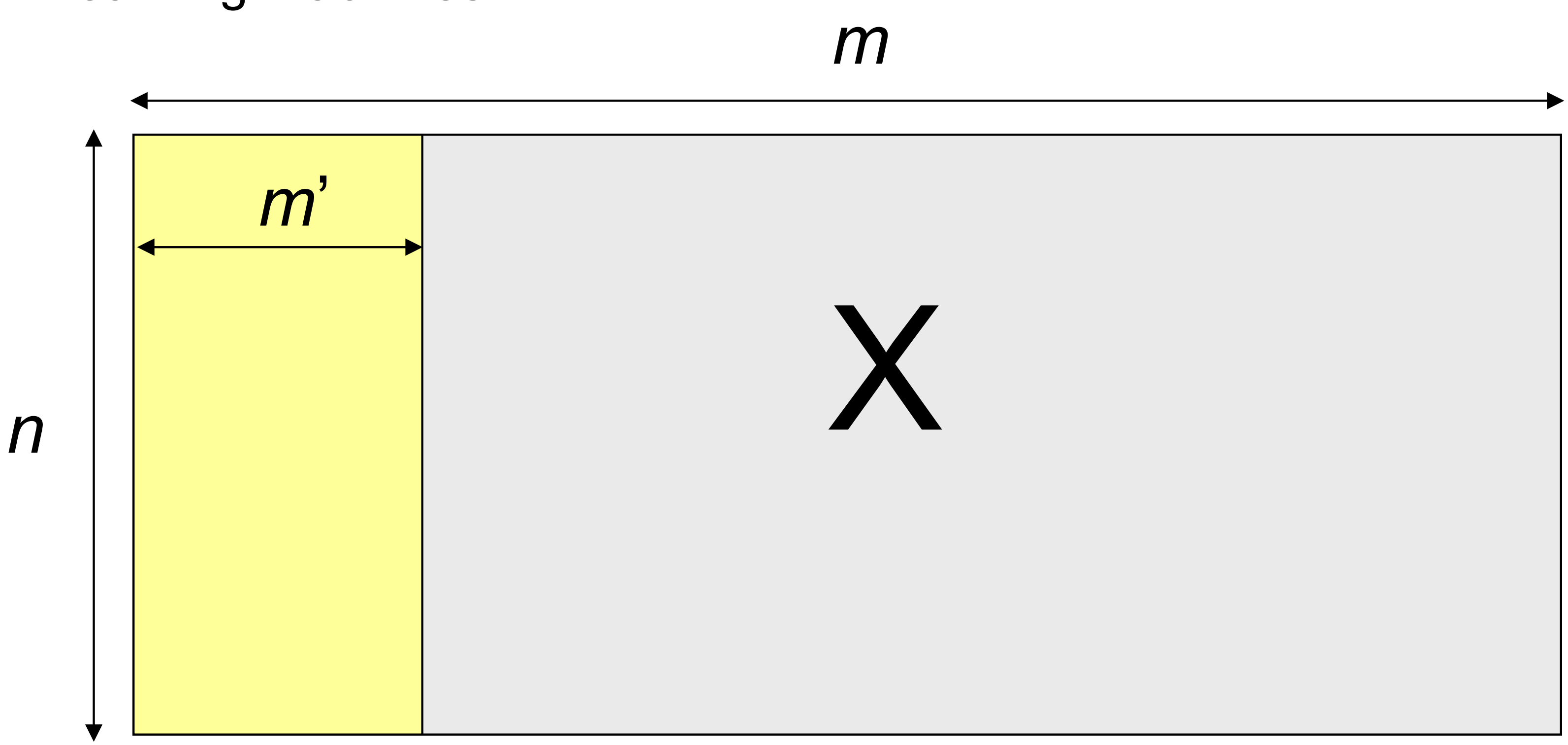
Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

Practical word2vec

- Pre-trained word embeddings are available for download online
  - Google News corpus
  - Freebase entities
- Can also train your own word2vec model, if you have more specialized data
- https://www.tensorflow.org/versions/master/tutorials/word2vec
- Another popular option:
  - GloVe (Pennington et al., 2014)

Feature selection

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



slide by Isabelle Guyon

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 like word2vec also count

Principal Component Analysis (PCA)

- **Idea**: Project data into a lower-dimensional sub-space,

  $R^m \rightarrow R^{m'}$, where $m' < m$.

Principal Component Analysis (PCA)

- **Idea**: Project data into a lower-dimensional sub-space,

  $R^m \rightarrow R^{m'}$, where $m' < m$.

- Consider a linear mapping, $x_i \mapsto Wx_i$
  - 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,

  $R^m \rightarrow R^{m'}$, where $m' < m$.

- Consider a linear mapping, $x_i \mapsto Wx_i$
  - 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: $\mathrm{argmin}_{W,U} \; \Sigma_{i=1:n} \| x_i - UWx_i \|^2$

Principal Component Analysis (PCA)

- **Idea**: Project data into a lower-dimensional sub-space,

  $R^m \rightarrow R^{m'}$, where $m'<m$.

- Consider a linear mapping, $x_i \mapsto Wx_i$
  - W is the compression matrix with dimension $R^{m'xm}$.
  - Assume there is a decompression matrix $U^{mxm'}$.
- Solve the following problem: $argmin_{W,U} \Sigma_{i=1:n} || x_i - UWx_i ||^2$

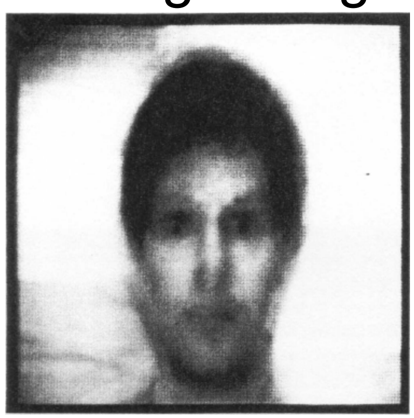
- Select the project dimension, m', using cross-validation.
- Typically "center" the examples before applying PCA (subtract the mean).

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 Eigen vector is an image,
  called an **Eigenface**.

Eigenfaces

Average image



Training images

Original images in R^{50x50}.

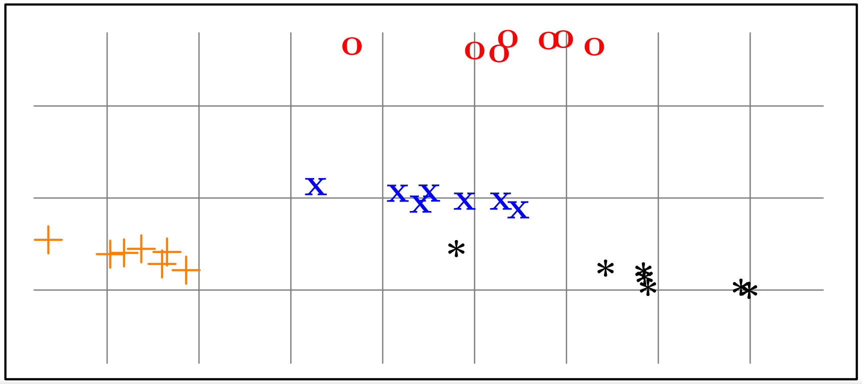
Projection to R^{10} and reconstruction

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:

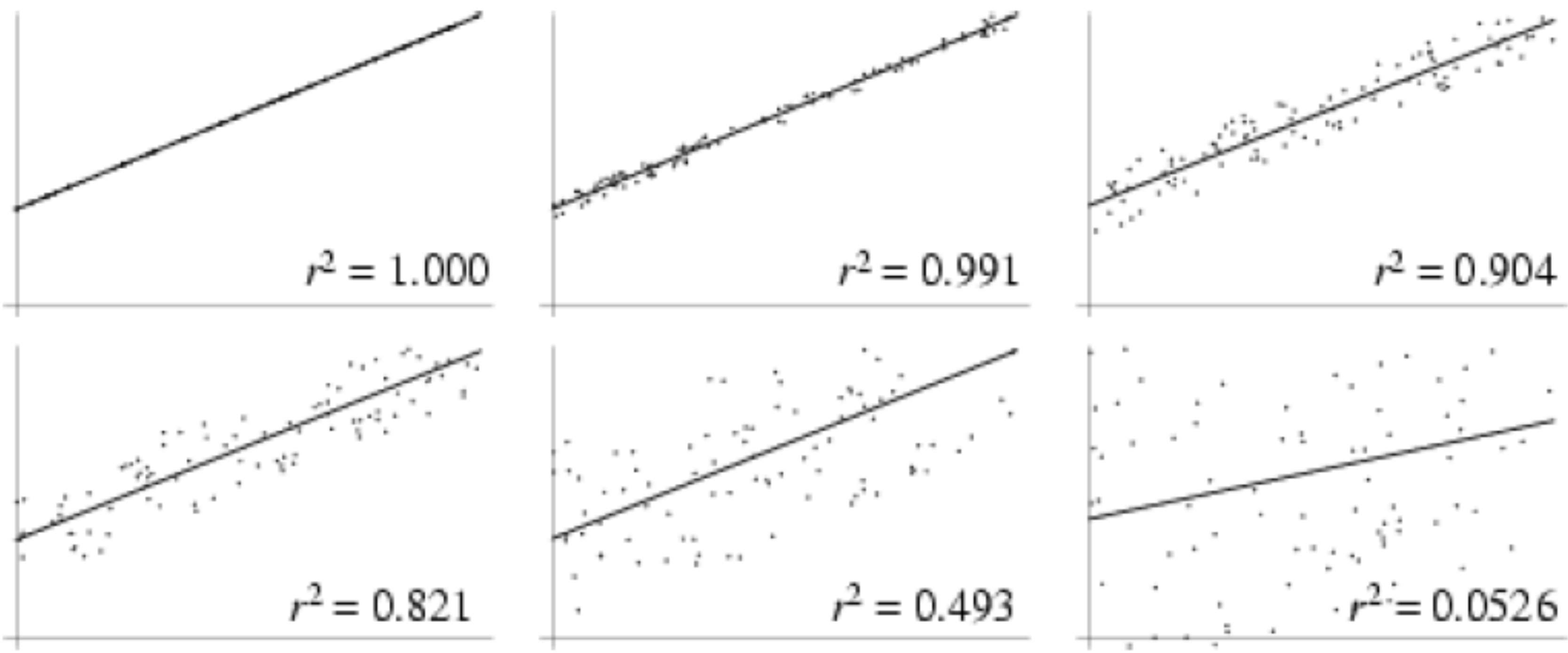
Variable Ranking

- **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

Strong correlation



Weak correlation

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

slide by Michel Verleysen

Scoring function: Mutual information

- Think of X_j and Y as random variables.

- Mutual information between variable X_j and target Y:

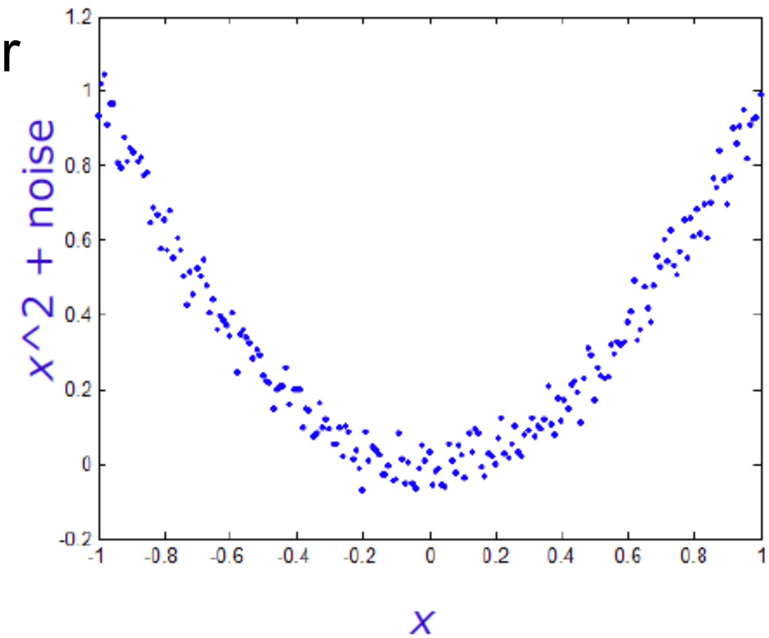
$$I(j) = \int_{X_j} \int_Y p(x_j, y) \log \frac{p(x_j, y)}{p(x_j) p(y)} \, dx \, dy$$

- Empirical estimate from data (assume discretized variables):

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

Nonlinear dependencies with MI

- Mutual information identifies nonlinear relationships between variables.
- Example:
  - x uniformly distributed over [-1 1]
  - $y = x^2 + \text{noise}$
  - 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

slide by Michel Verleysen

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?

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.

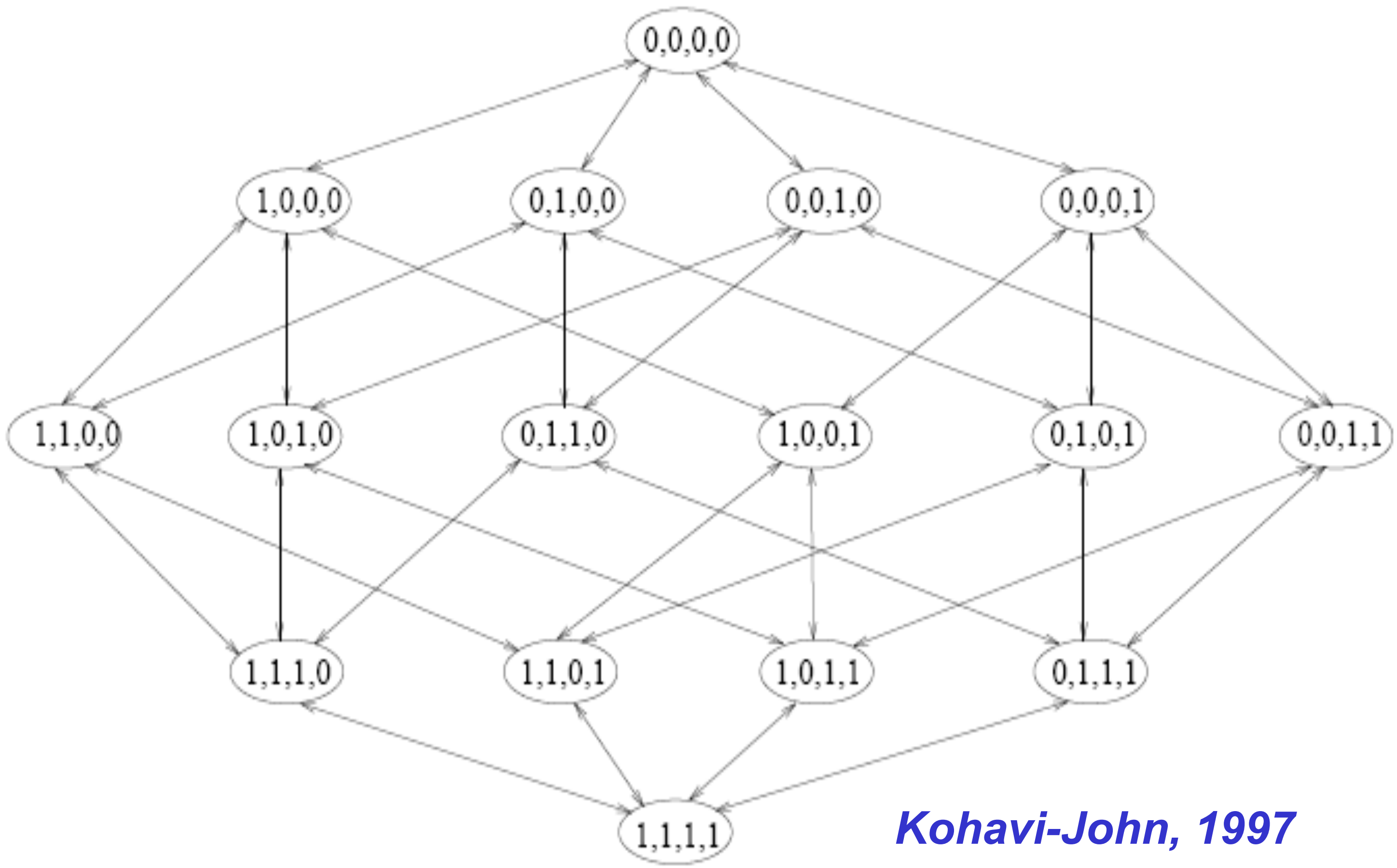
- Pros / cons?
  - Need select a scoring function.
  - Must select subset size (*m'*): cross-validation
  - Simple and fast – just need to compute a scoring function *m* times and sort *m* scores.
  - Scoring function is defined for individual features (not subsets).

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!

slide by Isabelle Guyon

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.

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

Evaluate on test set

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

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).

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

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 architecture (more on this next week.)