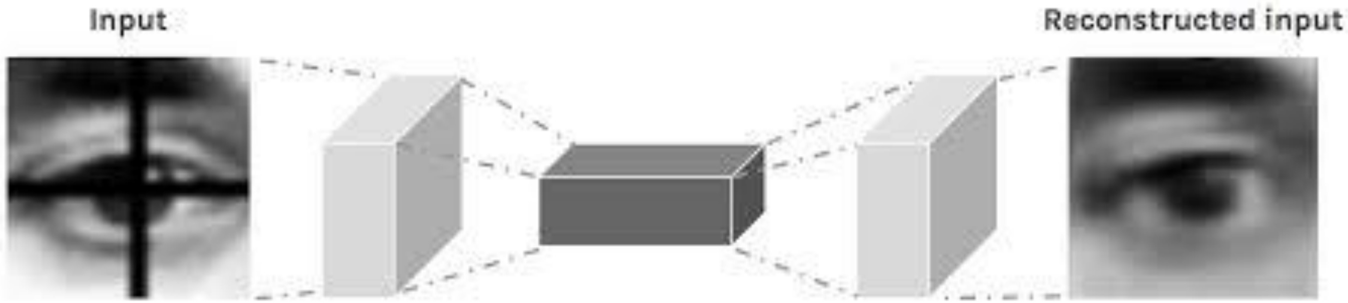
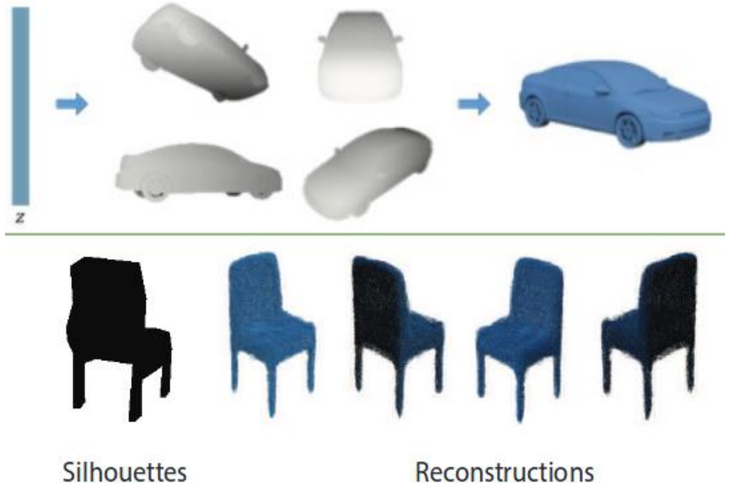
Feature Engineering



“Sense” of Human

- Definition of "**sense**”:
  - "A system that consists of a group of sensory cell types that responds to a specific physical phenomenon, and that corresponds to a particular group of regions within the brain where the signals are received and interpreted.”
  - “A physiological capacity of organisms that provides data for perception.”

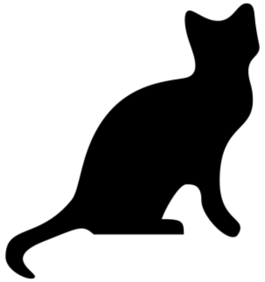


By Arsi Warrior - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=68328908

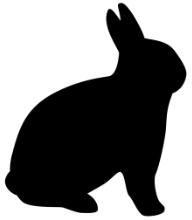
Five "Traditional" Senses of a Human

- Sight
- Hearing
- Taste
- Smell
- Touch

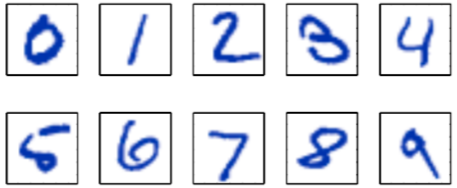
Cat?



Yes!



No!



0 1 2 3 4
5 6 7 8 9

Hot

Hot

Hot?

Yes!

No!

What is the Perception of Machines?

- How to represent the world to a machine?
  - How can you interpret “hot”/"soft”?
  - How can you represent the chair looked from different directions?

  =>Mapping raw data to features.
- How to select the sensory information to be sent to a machine?
  - How to decide which information is more important?
  - How to extract this information and transform the data correctly?
- How to learn from experience/examples?
  - Again, **generalization**: the ability to categorize correctly new examples that differ from those used for training.

Why Feature Engineering?

- Help the model to understand the data set as the same or similar way the human beings do.
- If the quality and size of the data are terrible, training longer or using a deeper network won’t help.
- The pre-processing of data and feature engineering are the foundation of the pyramid.
- Preparing a better dataset can be more important than tuning the parameters for your model.

Representation of The Real World

- AI provides human with powerful tools for ‘better’ decision making.
- To accomplish a task, AI needs human to :
  - Formulate the real-world problem to those that can be read by computers.
  - Choose a model of the task & choose learning algorithms for the model.
  - Find useful raw data of the task.
  - Convert raw data to the formats that the computer can read → e.g. features.

Outline

- Features and Feature Engineering
- Tackling Feature Explosion

Features and Feature Engineering

Introduction

Transforming Data

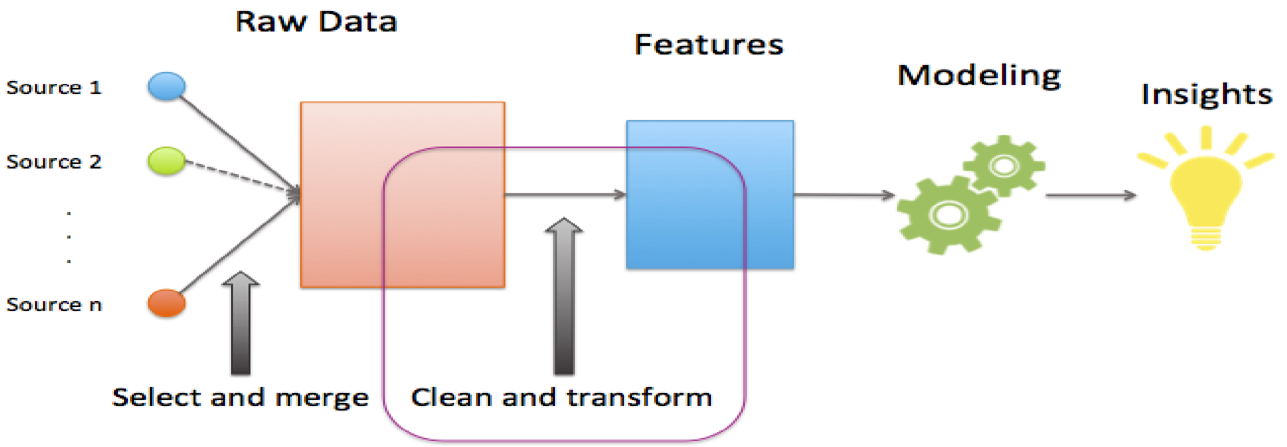
Examples

Feature

- Feature: information that describes a problem at hand and is potentially useful for prediction / problem-solving.

Feature Engineering

- Feature engineering: the process of determining which features might be useful in training a model, then creating these features by transforming raw data.
- In short: design and process features for AI applications.
  - An informal terminology, but is considered essential in applied AI.



Feature Learning / Extraction

- Feature learning process
  1) understand the properties of the task and how they may interact with the strengths and limitations of the chosen model
  2) design a set of features
  3) run experiments and analyze the results on a validation dataset
  4) change the feature set
  5) go to 2).
- Difficult and expensive.
- Automated feature learning is preferred.

Features and Feature Engineering

Introduction

Transforming Data

Examples

Feature Types

- Numerical features
  - Floats
  - Integers
- Categorical features
  - Discrete set of possible values (e.g., names of students of AAI): One-/Multi-hot encoding to map the categorical data to binary vectors.
- Image features.
- …

Need to transform data!

Reasons for Data Transformation [8]

- **Mandatory transformations** for data compatibility.
  - Converting non-numeric features into numeric.
  - Resize inputs to a fixed size.

- **Optional quality transformations:** may help the model perform better.
  - Tokenization or lower-casing of text features.
  - Normalized numeric features.
  - Allowing linear models to introduce non-linearities into the feature space.

Why Normalize Numeric Features?

- Normalization is necessary
  - If you have very different values within the same feature.
    - Without normalization, your training could blow up with *NaNs* if the gradient update is too large..
  - If you have two different features with widely different ranges.
    - This may cause the gradient descent to "bounce" and slow down convergence.
    - A possible solution: using heterogeneous learning rate.

Normalization Techniques

Normalization Technique	Formula	When to Use
Linear Scaling	$x' = (x - x_{min})/(x_{max} - x_{min})$	When the feature is more-or-less uniformly distributed across a fixed range.
Clipping	if x > max, then x' = max. if x < min, then x' = min	When the feature contains some extreme outliers.
Log Scaling	$x' = \log(x)$	When the feature conforms to the power law.
Z-score	$x' = (x - \mu) / \sigma$	When the feature distribution does not contain extreme outliers.

Table source: https://developers.google.com/machine-learning/data-prep/transform/normalization

Bucketing

- Sometimes, you need to transform numeric features into categorical features, using a set of thresholds.

  =>Bucketing.

- Quantization.

Transforming Categorical Data

- One-/Multi-hot encoding
- Hashing
- Embeddings: A categorical feature represented as a continuous-valued feature (high-dimensional vector -> low-dimensional space).

One-/Multi-hot encoding: Example

1. “I have a cat.”
2. “Cats have fur.”

One-hot encoding

	I	cat	a	have	fur
I	1	0	0	0	0
have	0	0	0	1	0
a	0	0	1	0	0
cat	0	1	0	0	0

	I	cat	a	have	fur
cat	0	1	0	0	0
have	0	0	0	1	0
fur	0	0	0	0	1

	I	cat	a	have	fur
1	1	1	1	1	0
2	0	1	0	1	1

Multi-hot encoding

Features and Feature Engineering

Introduction

Transforming Data

Examples

(1) Numerical Features

What you see

What a computer see (raw data)

StateObservation{
gameScore=0,
gameTick=0,
gameWinner=NO_WINNER,
isGameOver=false,
worldDimension=[250.0, 200.0],
blockSize=10,
noOfPlayers=1,
...
}

Feature Engineering

$$[0.0, 0.0, 0.0, -1.0, 250.0, \ldots]$$

Feature vector

(2) Text Features

Raw text → Bag of words → Classify: Puppy or not?

It is a puppy and it is extremely cute.

{"it": 2,
"is": 2,
"a":1,
"puppy":1,
"and":1,
"extremely":1,
"cute":1}

(2) Text Features (*Continued*)

Raw text → Bag of words → Classify: Puppy or not?

It is a puppy and it is extremely cute.

it	2
they	0
I	0
am	0
how	0
puppy	1
and	1
cat	0
aardvark	0
cute	1
extremely	1
…	…

Sparse representation

(3) Image Features

Raw image → Bag of pixels → Classify: Person or object?

Image source: "Recognizing and learning object categories," Li Fei-Fei, Rob Fergus, Anthony Torralba, ICCV 2005—2009.

(3) Image Features

Raw image → Bag of pixels → Classify: Person or object?

3.29
-15
-5.24
46.3
1.36
47.1
.
.
.
2.83
95.4
-19
-89
5.09
37.8

Dense representation

(4) Signal Features

Raw signal → Bag of samples → Classify: Person or not?

-6.10
-9.15
-6.10
-6.10
-6.10
-6.10
.
.
.
-3.05
-3.05
-3.15
0
0
0

Dense representation

Tackling Feature Explosion

- ❖ Introduction to Feature Explosion
- ❖ Feature Selection
- ❖ Regularization (more detailed)

Feature Explosion

- Initial features are always an expression of prior knowledge.
  - text: words, grammatical classes and relations, etc.
  - image: pixels, contours, textures, etc.
  - signal: samples, spectrograms, etc.
- Feature combinations might work better.
- Both lead to (extremely) large number of features.
- Too many features become a problem given the limited size of training data. → overfitting.

Problems of Feature Explosion

- Storage cost
- Irrelevant, redundant or even harmful features
- Large number of required training samples
  - Adding another feature need exponential increase in training samples.
- Dysfunctional distance functions
  - When a measure such as Euclidean distance is used, there is little difference in distance between different pairs of samples.

Benefits of Small Feature Set

- Lead to simpler models.
- Easier to interpret by researchers/users.
- Shorter training times.
- Less computational burden.
- Enhanced generalization by reducing overfitting.
- Reduced feature measurement cost.
- …

Dealing with Feature Explosion

- **Feature selection**: could use a greedy method.
  - Select some of the features that can reach some best ‘criterion’.

- **Regularization**:
  - Include all possible features.
  - Penalize ‘complex’ hypothesis.

Tackling Feature Explosion

- ❖ Introduction to Feature Explosion
- ❖ Feature Selection
- ❖ Regularization (more detailed)

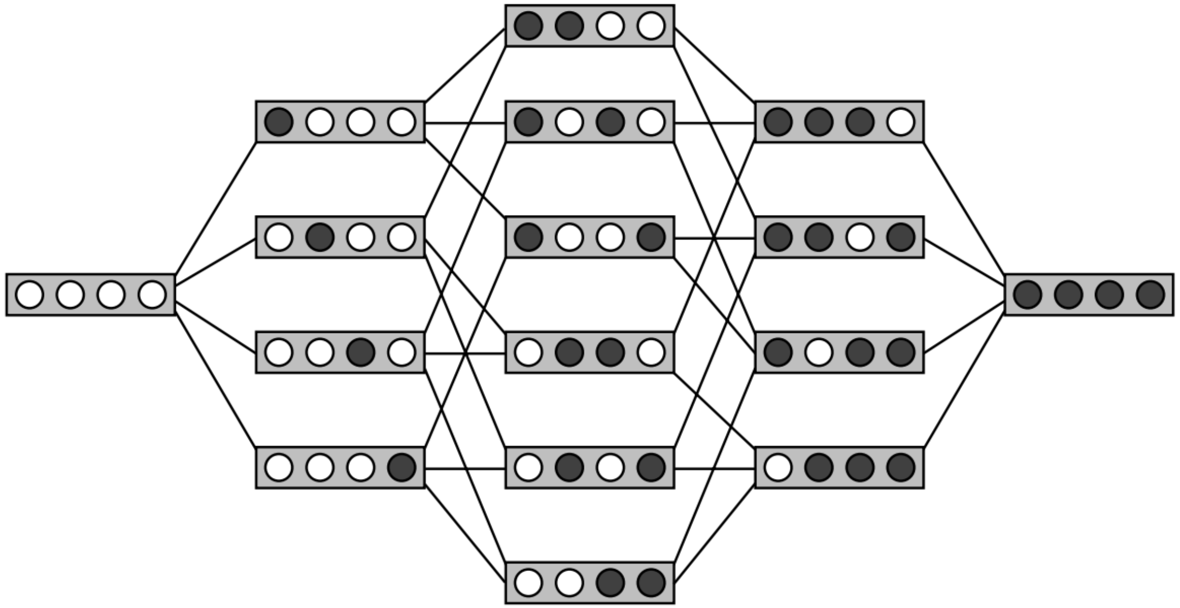
Selecting Feature Subset

- Reduce the original feature space by throwing out some features.
- Assumption: features are redundant or irrelevant.
- Motivation: Training data are limited.
  - Restricting #features is a feasible control mechanism.
  - Compact and representative explanation of the task follows Occam's razor.
- Research Question: How to select 'good' features from the feature space?
- Feature selection is a search problem.

Feature Selection is a Search Problem

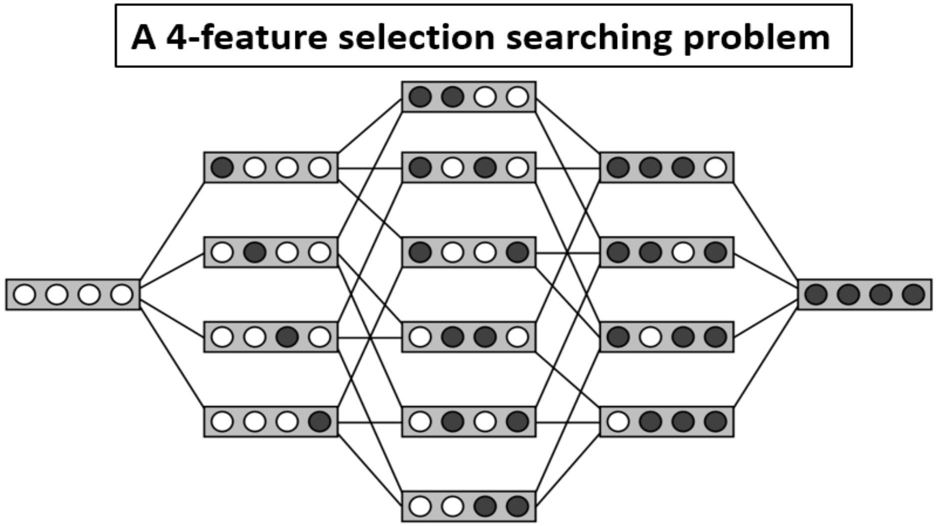
- **The state-space formulation:**
  - states: all possible feature subset
  - initial state: ?
  - actions: ?
  - next state: updated feature subset
  - goal test: ?
  - cost: computational cost
- **Technical question:**
  - How to search?
  - How to evaluate selected features?
  - When to stop?

A 4-feature selection searching problem



Search Space of Feature Selection

- Question: How large is the search space?
- Answer: 2^d, d is #features.
- The search space of the illustration is $2^4 = 16$ and is feasible to search.
- When d gets larger and larger, it will become infeasible to search in practice.
- We need heuristics to guide our search → heuristic search.



Heuristic Search for Feature Selection

- Question: How to do heuristic search in the entire 2^d space?
- One possible idea: Greedy heuristic search.

Initial State

- Empty feature set: one starts with an empty set and progressively add features yielding to the improvement of a performance index.

  → **forward selection.**

- Full feature set: one starts with all the features and progressively eliminate the least useful ones.

  → **backward elimination.**

Actions

- Forward selection: add one feature each step.
- Backward elimination: remove one feature each step.

- They are called *sequential feature selection* (SFS) methods.

- Optimality: They do not examine all possible feature subsets, so no guarantee of finding the optimal subset.

Compare Forward Selection to Backward Elimination

- Both procedures are reasonably fast and robust against overfitting
- Both procedures provide **nested** feature subsets.
- However, they may lead to different subsets and one may be preferred over the other.

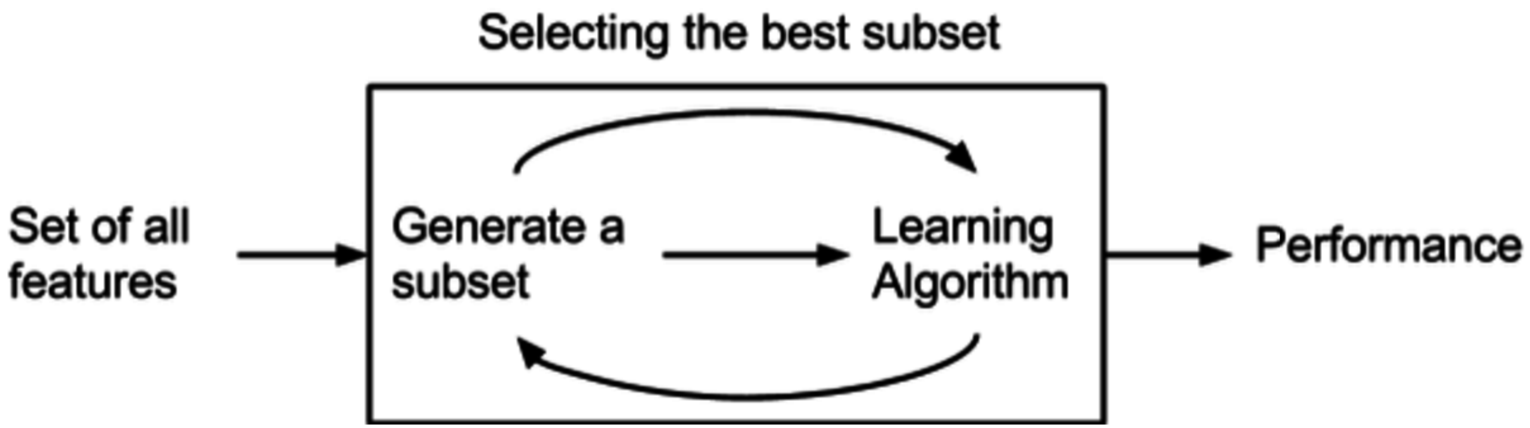
Goal Test

- How to evaluate selected features? e.g.,
  - information theory;
  - prediction accuracy on the training set or validation set.
- When to stop?
  - Simply use the change of a performance metric.
  - Adding or deleting a feature cannot further improves some prediction accuracy.
  - Reach the empty or full feature set.

Three Typical Methods

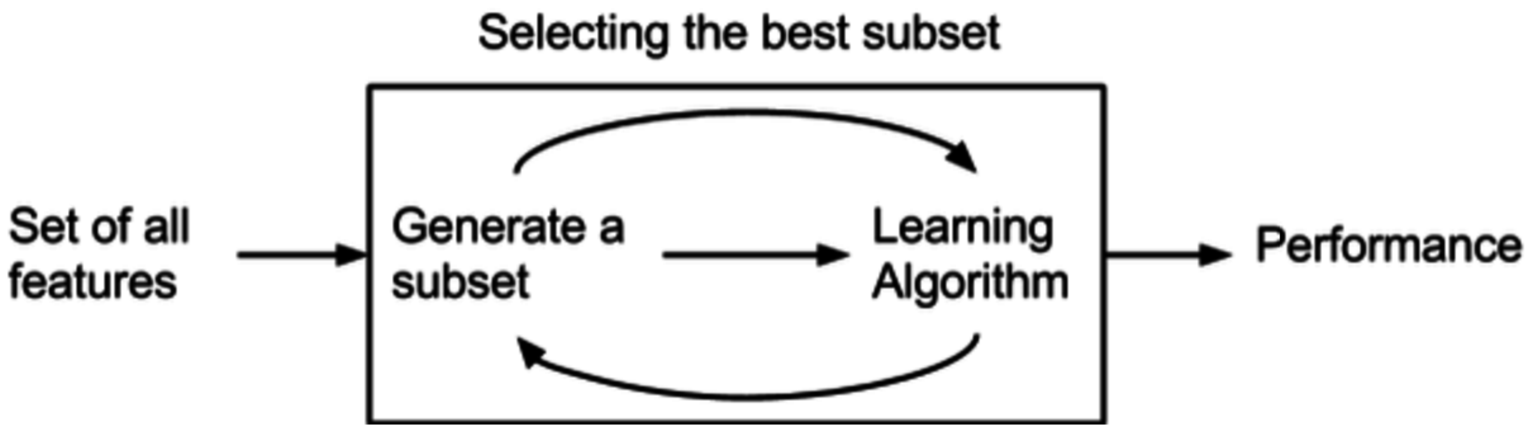
1. Wrapper methods
2. Filter methods
3. Embedded methods

1. Wrapper Methods



- Basic idea: model dependent
  - Navigate feature subsets by adding/removing features.
  - Evaluate the performance of the chosen model on the validation set.
  - Repeat until no improvement to the validation set accuracy.
- Assessment: use cross validation

1. Wrapper Methods



- Advantage: highly accurate
- Disadvantage: computationally expensive, risk of overfitting
- Examples: recursive feature elimination, sequential feature selection, genetic algorithms

2. Filter Methods

- Basic idea: independent of learning models
  - Rank features on some *heuristic* score based on their relevance to the AI task
  - Choose a subset based on the sorted scores
- Heuristic score: many popular scores [4]
  - Does the individual feature seem helpful in prediction?
  - Classification with categorical features: $\mathcal{X}^2$, information gain, document frequency
  - Regression: correlation, mutual information
- Assessment: use statistical tests

2. Filter Methods

- Advantages
  - very fast & simple to apply
  - usually better generalization
- Disadvantage
  - not take into account interactions between features
  - not as accurate as Wrappers
- Suggestions:
  - use it as a pre-processing for further Wrapper feature selection
- Examples: Belief, correlation-based filters, fast correlated-based filters

Example: Correlation-based Filters

- Hypothesis: A good feature should be highly correlated to the output but not very correlated with each other.

- Technical questions: for a classification problem
  1. Whether a feature is relevant to the class?
  2. Whether a relevant feature is redundant with other relevant features?

Example: Correlation Scores

Two groups of correlation metrics between random variables X and Y:

1) Classical linear correlation: e.g. Pearson correlation

$$\rho(X,Y) = \frac{\sum_i (x_i - \bar{x})\,(y_i - \bar{y})}{\left[\sum_i (x_i - \bar{x})^2 \cdot \sum_i (y_i - \bar{y})^2\right]^{\frac{1}{2}}} \in [-1,+1]$$

- Advantage: easy and fast to compute
- Disadvantage:
    - cannot capture nonlinear correlation
    - calculation requires all features contain numerical values

Example: Information Gain

Two groups of correlation metrics between random variables X and Y:

2) Information theory: e.g. information gain [Quinlan, 1993]

$$IG(X;Y) = \mathcal{H}(X) - \mathcal{H}(X|Y)$$

- $\mathcal{H}(X) \triangleq -\sum_k p(x_k) \log_2 p(x_k)$ is entropy of X
- $\mathcal{H}(X|Y) \triangleq \sum_j p(Y = y_j) \cdot \mathcal{H}(X|Y = y_j)$ is conditional entropy

- Advantage: capture nonlinear correlation
- Disadvantage:
  - higher computational cost
  - IG is biased in favor of features with more values

Example: Symmetric Uncertainty

Two groups of correlation metrics between random variables X and Y:

2) Information theory: e.g. symmetric uncertainty [Press et al., 1988]

$$IG(X;Y) = \mathcal{H}(X) - \mathcal{H}(X|Y) = -\sum_{\mathrm{j}} \sum_{k} p(x_k, y_j) \log_2 \frac{p(x_k, y_j)}{p(x_k)p(y_j)}$$

$$IG(Y;X) = \mathcal{H}(Y) - \mathcal{H}(Y|X) = -\sum_{k} \sum_{j} p(y_j, x_k) \log_2 \frac{p(y_j, x_k)}{p(y_j)p(x_k)}$$

$$SU(X;Y) = 2\left[\frac{IG(X;Y)}{\mathcal{H}(X) + \mathcal{H}(Y)}\right] \in [0,1]$$

- Advantage:
  - compensate for IG's bias towards features with more values
  - normalize its values to [0,1]

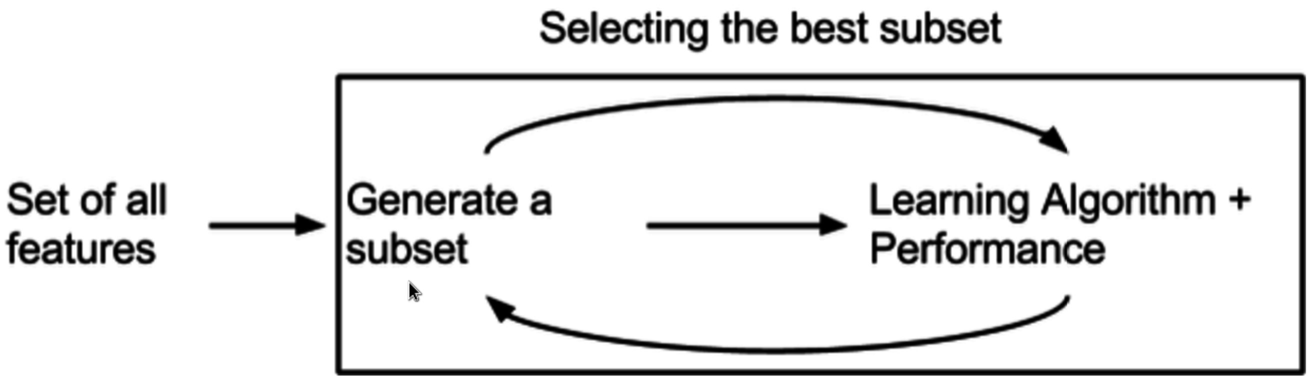
Example: Correlation-based Filters

- Main Procedure

  1) C-correlation: Use some *correlation score* to rank features according to their correlation to the class.
  2) Ranking cut-off is determined by the user to form the relevant feature set.
  3) F-correlation: Some relevant features are removed by redundancy detection based on the same *correlation measure*.

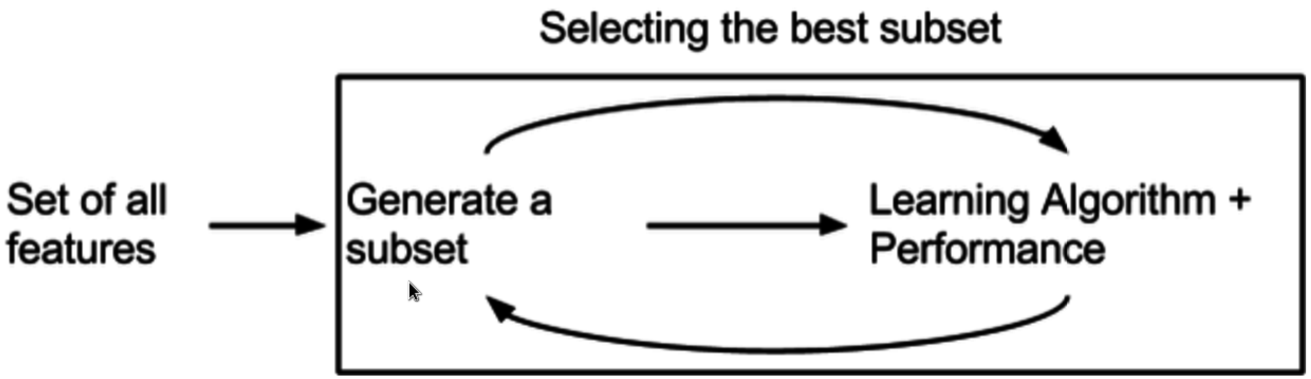
- Read paper [6] for details.

3. Embedded Methods



- Basic idea: Feature selection is part of model construction, and feature search is guided by the learning process.
- Assessment: use cross validation
- They use the specific structure of the model returned by the algorithm to get the set of 'relevant' features.

3. Embedded Methods



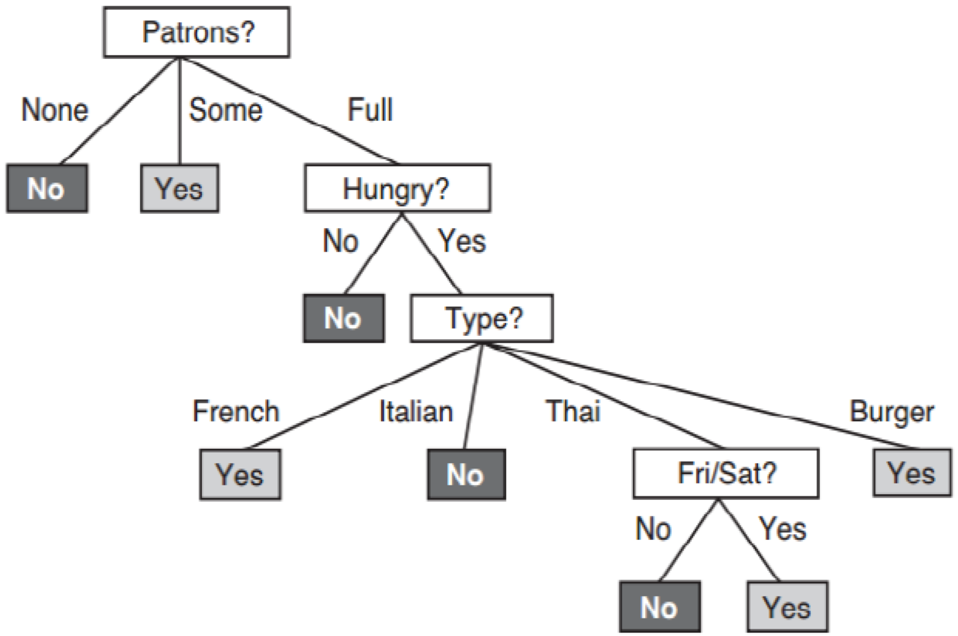
- Advantages:
  - similar to Wrappers, but
  - less computationally expensive & less prone to overfitting
- Examples: classification and regression trees, C4.5, random forest

3. Embedded Methods

- They are not too far away from wrapper techniques.
- They are a good inspiration to design new feature selection techniques for your own algorithms.
  - Find a function of features that represents your prior knowledge about what a good model is.

Example: Decision Tree

- Review: construct decision tree
  - Start from an empty tree.
  - Split the next best feature based on *information gain*.
  - Repeat.
- Tree construction is the process of feature selection.
- Not all features are used in the constructed tree.



Four features out of total 10 are used in constructing the decision tree.

Summary: Three Typical Methods

- Wrapper methods: model specific
- Filter methods: independent of model
- Embedded methods: feature selection is embedded in model learning

Tackling Feature Explosion

- ❖ Introduction to Feature Explosion
- ❖ Feature Selection
- ❖ Regularization (more detailed)

Regularization

- Basic idea:
  - The more features matter in the model, the bigger complexity.
  - Regularization = introducing penalty for complexity → reduce features
- Interpretation:
  - It bias the model toward lower complexity (fewer features).
  - Application of Occam's razor: the model should be simple (fewer coefficients).
  - Bayesian viewpoint: regularization = imposing prior knowledge that the world is simple on the learning model.

Regularization Formulation

- Find $f \in \mathcal{F}$ minimizing

$$\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_{tr}\big(y_i, f(x_i)\big) + \lambda \cdot \Omega(f)$$

  - $\mathcal{F}$: a class of candidate functions
  - $\Omega(f)$: the complexity of a model f
  - $\lambda > 0$: a regularization parameter
- Question: How do we pick parameter λ?
- Answer: Cross validation.

Examples of Regularization Methods

- Ridge regression [Hoerl and Kennard 1970]
- Lasso regression [Tibshirani 1996]
- Smoothing splines [Wahba 1990]
- Support vector machines [Vapnik 1998]
- Regularized neural networks
- etc.

Review: Multivariate Linear Regression

- Given: data $\boldsymbol{X} \in \mathbb{R}^{N\times D}$, and output $\boldsymbol{y} \in \mathbb{R}^{N\times 1}$.
  - N: #samples, D: #features
- Aim: find $\boldsymbol{\theta} \in \mathbb{R}^{D\times 1}$ to minimize $\frac{1}{2}\left|\left|\boldsymbol{X\theta} - \boldsymbol{y}\right|\right|_2^2$.
- Solution: $\boldsymbol{\theta} = \left(\boldsymbol{X}^{T}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{T}\boldsymbol{y}$

Feature Selection in MLR Model

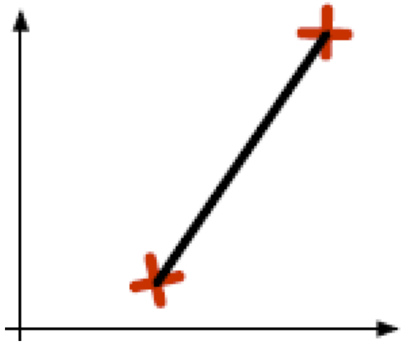
- In a MLR model, each θ_i corresponds to one feature.
- Feature selection can be treated as the penalty on $\boldsymbol{\theta}$:

$$\Omega(f) := \left|\left|\boldsymbol{\theta}\right|\right|_p$$

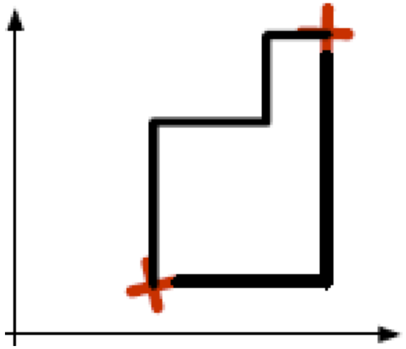
  - $\theta_i = 0$: remove the i^{th} feature from the model
  - ℓ_p norm of $\boldsymbol{\theta}$

Penalty ℓ_p

- Euclidean $p = 2, ||\boldsymbol{\theta}||_2 = \sqrt{(\theta_1\text{^}2\ ) + \cdots + \theta_D\text{^}2}$
- ℓ_2 can be viewed as a Gaussian prior on model paramete



- Manhattan $p = 1, ||\boldsymbol{\theta}||_1 = |\theta_1| + \cdots + |\theta_D|$
  - ℓ_1 can be viewed as a Laplace prior on model parameters



- Generally $0 < p < \infty, ||\boldsymbol{\theta}||_p = \sqrt[p]{|\theta_1|^p + \cdots + |\theta_D|^p}$

Ridge Regression

- Ridge regression model: $\min_{\boldsymbol{\theta}} \frac{1}{2}||\boldsymbol{X\theta} - \boldsymbol{y}||_2^2 + \lambda||\boldsymbol{\theta}||_2^2$
- Solution: $\boldsymbol{\theta} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$

- Lead to a solution with many small $\boldsymbol{\theta}$.
  - ℓ_2 does not strongly zero parameters (remove features), but still limits model complexity and get fewer features.
  - It also solves the problem that $\boldsymbol{X}^T\boldsymbol{X}$ is not invertible

Lasso Regression

- Lasso regression model: $\min_{\boldsymbol{\theta}} \frac{1}{2}||\boldsymbol{X\theta} - \boldsymbol{y}||_2^2 + \lambda||\boldsymbol{\theta}||_1$
- Solution: no analytical solution
  - Need sub-gradient of ℓ_1 norm
- Lead to a sparse solution, i.e., $\boldsymbol{\theta}$ has many zero elements.
  - Remove many features and preferable for high-dimensional problems

Regression with Penalty $\ell_{1/2}$

- Penalty $\ell_{1/2}$ model: $\min_{\boldsymbol{\theta}} \frac{1}{2} ||\boldsymbol{X\theta} - \boldsymbol{y}||_2^2 + \lambda ||\boldsymbol{\theta}||_{\frac{1}{2}}$
- Solution: non-convex and thus hard to optimize
  - Initialize with ℓ_1 penalty solution
  - Further perform gradient steps
  - Not optimal but give sparser solutions than ℓ_1.
- Lead to an even sparser solution, and often better performance.

Remarks on ℓ_1

Two types of ℓ_1 penalty used in regression:

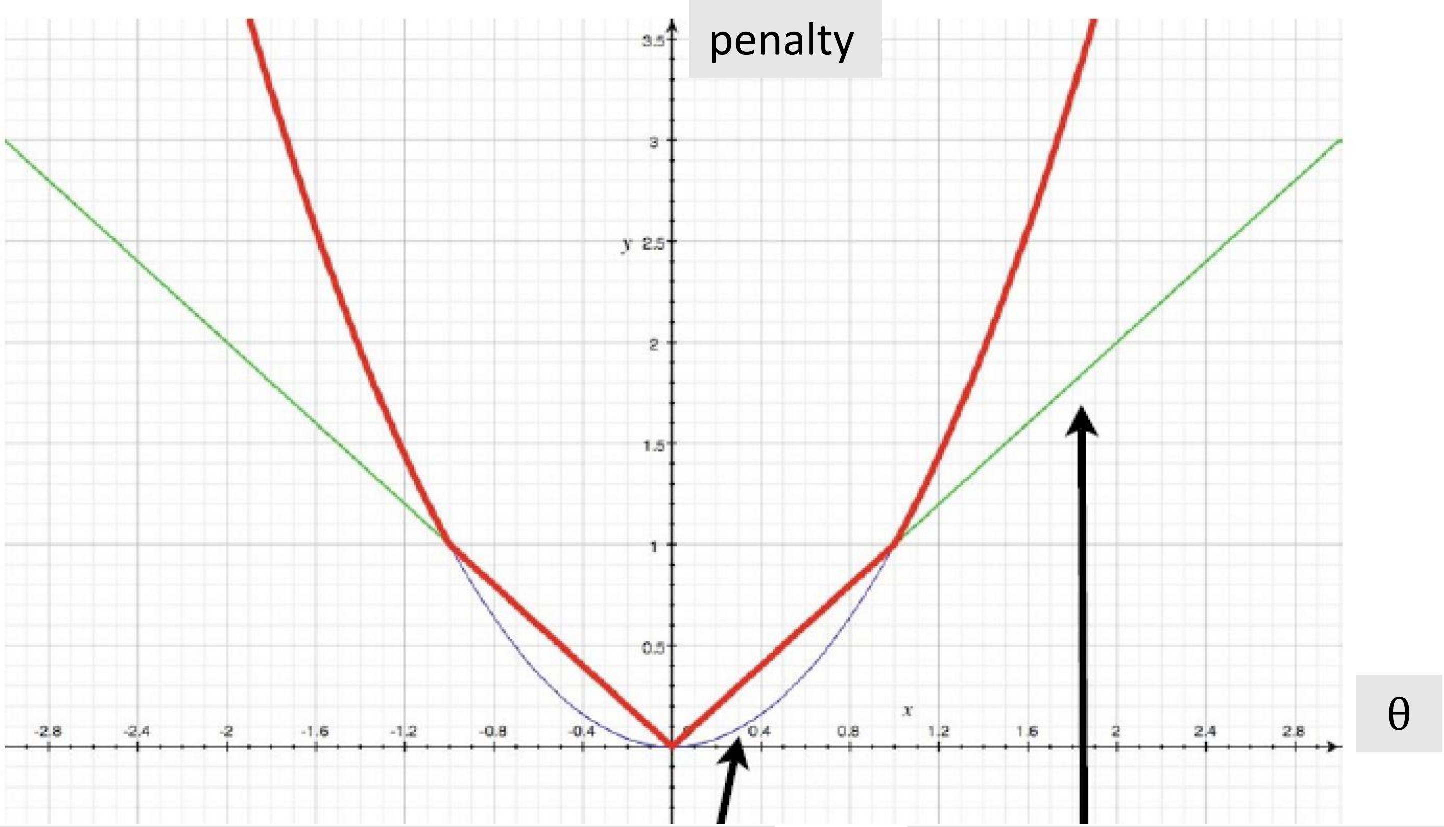
- Lasso for sparsity

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}||\boldsymbol{X\theta} - \boldsymbol{y}||_2^2 + \lambda||\boldsymbol{\theta}||_1$$

- ℓ_1 loss for robustness

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} ||\boldsymbol{X\theta} - \boldsymbol{y}||_1 + \lambda||\boldsymbol{\theta}||_p$$

Remarks on ℓ_1 *Continue*



ℓ_1 penalizes more than ℓ_2 when θ is small → use for sparsity

ℓ_1 penalizes less than ℓ_2 when θ is big → use for robustness

Summary

- Feature engineering is often crucial to get good results.
- Manual feature learning requires knowledge for the task.
- Automated feature learning is much more preferred.
- Strategies for tackling feature explosion:
  - Feature selection is a *heuristic* search problem.
  - Use regularization on all possible features to prevent overfitting.

Reading Materials For This Lecture

[1] Online course: http://clopinet.com/isabelle/Projects/ETH/

[2] A. Zheng. and A. Casari. 2017. *Mastering Feature Engineering for Machine Learning Models*. Chapter 3.

[3] Curse of dimensionality: https://en.wikipedia.org/wiki/Curse_of_dimensionality

[4] Y. Yang and J. O. Pedersen. 1997. *A Comparative Study on Feature Selection in Text Categorization*. ICML. pp:412-420.

[5] Blog: https://machinelearningmastery.com/an-introduction-to-feature-selection/

[6] L. Yu and H. Liu. 2003. Feature Selection for High-dimensional Data: A Fast Correlation-based Filter Solution. ICML. pp: 856-863

[7] An Introduction to Feature Extraction. Isabelle Guyon and André Elisseeff.

[8] https://developers.google.com/machine-learning/data-prep