UVA CS 6316/4501 – Fall 2016 Machine Learning

Lecture 22: Review

Dr. Yanjun Qi

University of Virginia

Department of
Computer Science

Announcements: Final Exam

- Closed Note
- Allowing a paper (us letter size) of cheat sheet
- No laptop / No Cell phone / No internet access / No electronic devices

- Recital session this Friday (@OSL120, 4pm-5pm) for HW7

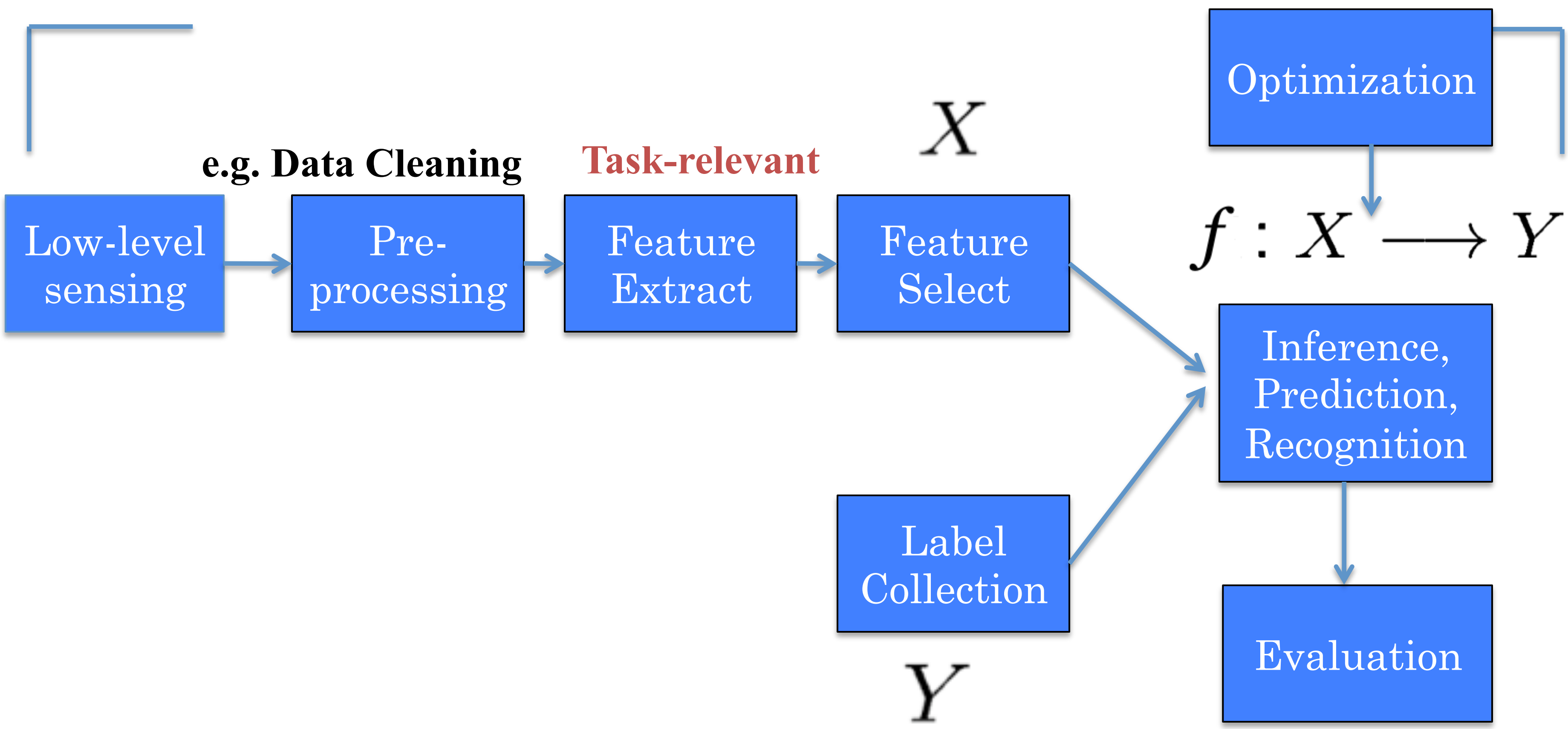
- Covering post-midterm contents (L12-) till today
  - Practice with sample questions in HW7
  - HW7 due next Monday noon
  - Please review course slides carefully

Today

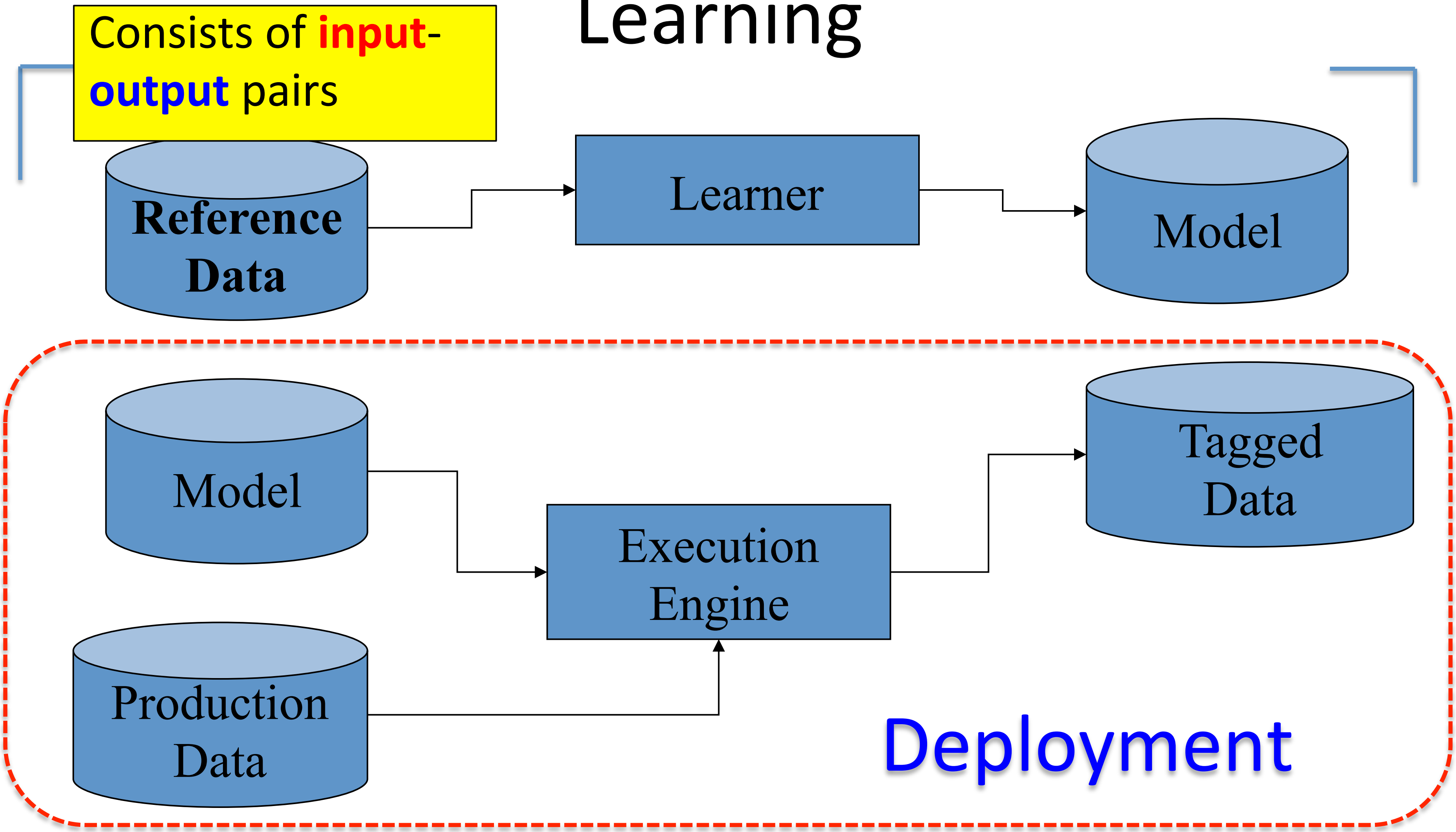
- ❑ Review of ML methods covered so far
  - ❑ Regression (supervised)
  - ❑ Classification (supervised)
  - ❑ Unsupervised models
  - ❑ Learning theory

- ❑ Review of Assignments covered so far

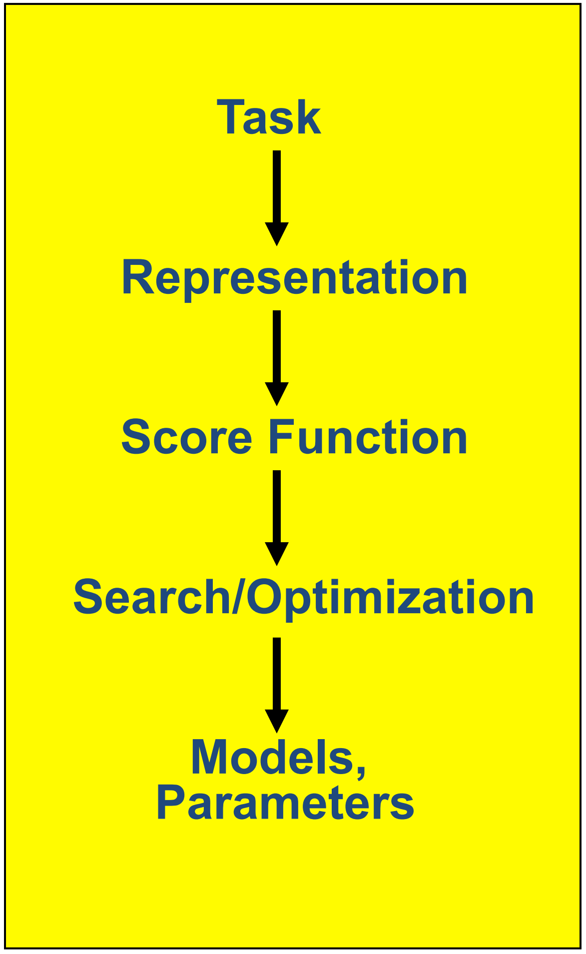
A Typical Machine Learning Pipeline



An Operational Model of Machine Learning



Machine Learning in a Nutshell



ML grew out of work in AI

Optimize a performance criterion using example data or past experience,

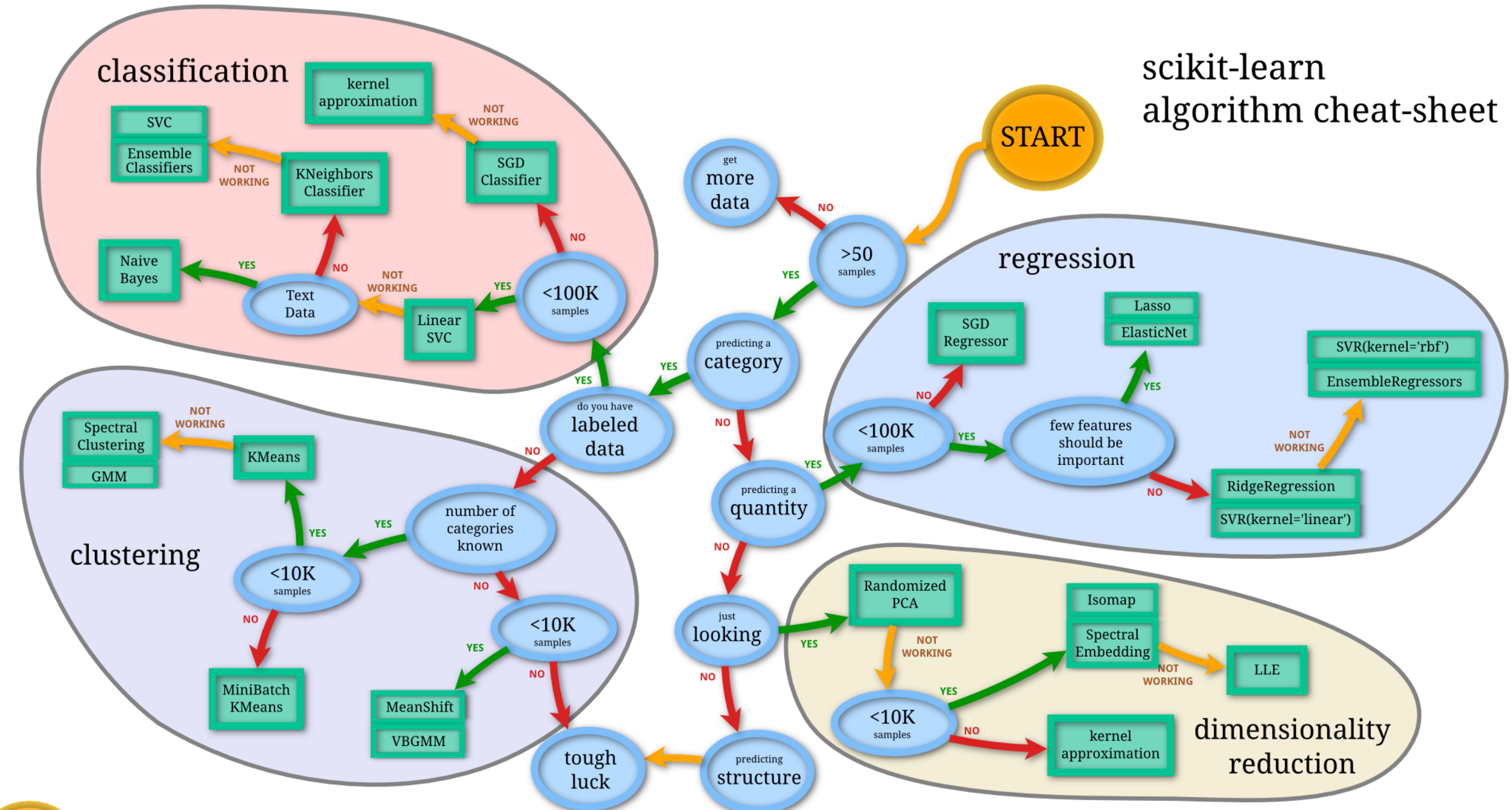
Aiming to generalize to unseen data

What we have covered

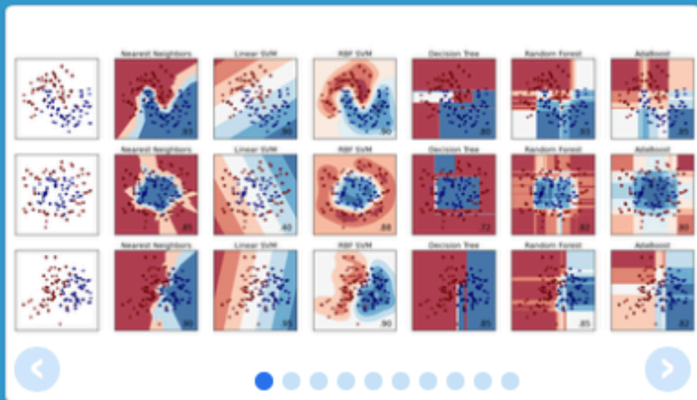
Task	
Representation	
Score Function	
Search/ Optimization	
Models, Parameters	

http://scikit-learn.org/stable/tutorial/machine_learning_map/

Scikit-learn algorithm cheat-sheet



http://scikit-learn.org/stable/



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which set of categories a new observation belong to.

Applications: Spam detection, Image recognition.
Algorithms: *SVM*, *nearest neighbors*, *random forest*, ... — *Examples*

Regression

Predicting a continuous value for a new example.

Applications: Drug response, Stock prices.
Algorithms: *SVR*, *ridge regression*, *Lasso*, ... — *Examples*

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: *k-Means*, *spectral clustering*, *mean-shift*, ... — *Examples*

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: *PCA*, *feature selection*, *non-negative matrix factorization*. — *Examples*

Model selection

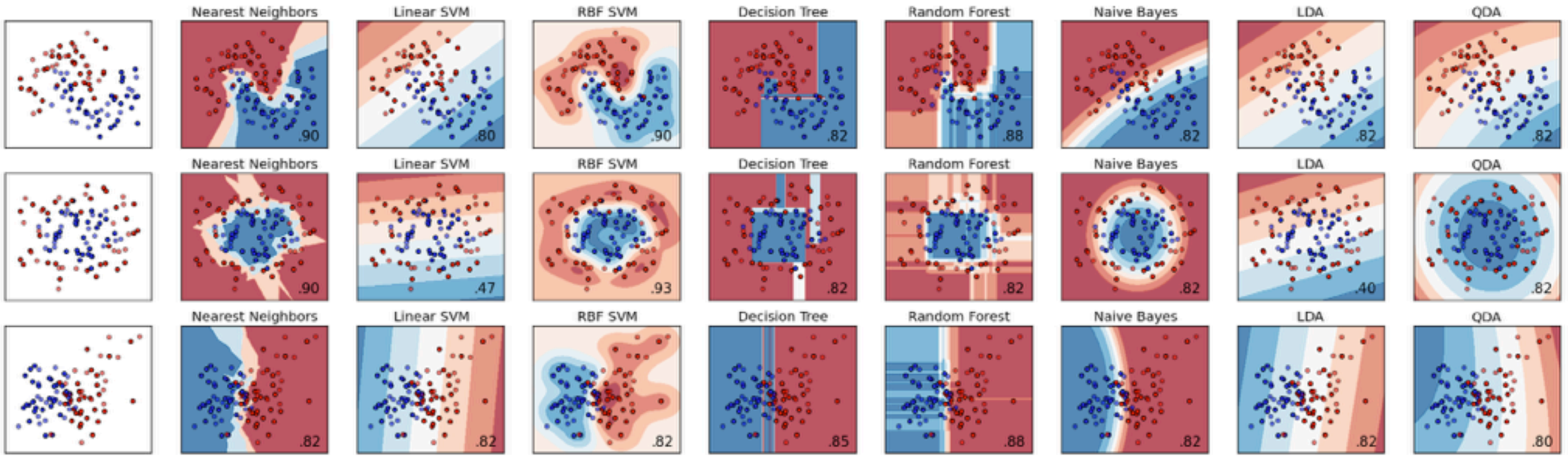
Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: *grid search*, *cross validation*, *metrics*. — *Examples*

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: *preprocessing*, *feature extraction*. — *Examples*



- ✓ different assumptions on data
- ✓ different scalability profiles at training time
- ✓ different latencies at prediction (test) time
- ✓ different model sizes (embedability in mobile devices)

Olivier Grisel's talk

Today

- ❑ Review of ML methods covered so far
  - ❑ Regression (supervised)
  - ❑ Classification (supervised)
  - ❑ Unsupervised models
  - ❑ Learning theory

- ❑ Review of Assignments covered so far

SUPERVISED LEARNING

$$f : X \longrightarrow Y$$

- Find function to map input space X to output space Y

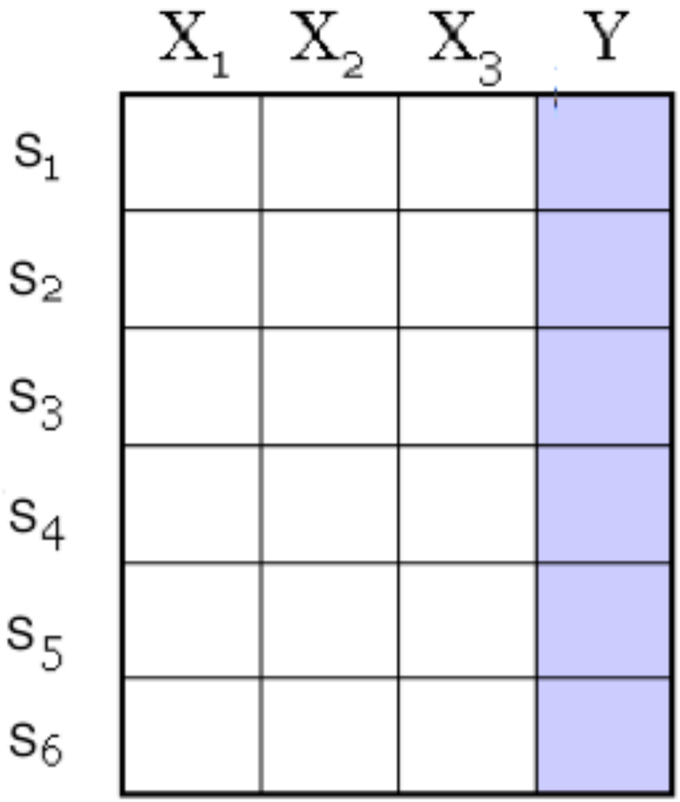
- Generalisation: learn function / hypothesis from past data in order to "explain", "predict", "model" or "control" new data examples

KEY

What we have covered (I)

- Supervised Regression models
  - Linear regression (LR)
  - LR with non-linear basis functions
  - Locally weighted LR
  - LR with Regularizations
  - Feature selection *

A Dataset



$$f : \boxed{X} \longrightarrow \boxed{Y}$$

Output Y as continuous values

- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/predictors/regressors*: [columns, except the last]
- **Target**/*outcome/response/label/dependent variable*: special column to be predicted [last column]

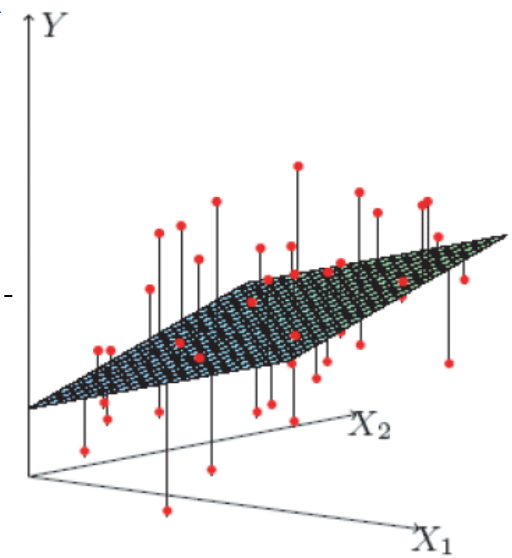
(1) Multivariate Linear Regression

Task → Representation → Score Function → Search/Optimization → Models, Parameters

Regression → Y = Weighted linear sum of X’ s → Least-squares → Linear algebra → Regression coefficients

$$\hat{y} = f(x) = \theta_0 + \theta_1 x^1 + \theta_2 x^2$$

$\rightarrow$ θ



(2) Multivariate Linear Regression with basis Expansion

Task	Regression
↓	↓
Representation	Y = Weighted linear sum of (X basis expansion)
↓	↓
Score Function	Least-squares
↓	↓
Search/Optimization	Linear algebra
↓	↓
Models, Parameters	Regression coefficients

$$\hat{y} = \theta_0 + \sum_{j=1}^{m} \theta_j \varphi_j(x) = \varphi(x)\theta$$

(3) Locally Weighted / Kernel Regression

Task	Regression
↓	↓
Representation	Y = local weighted linear sum of X’s
↓	↓
Score Function	Least-squares
↓	↓
Search/Optimization	Linear algebra
↓	↓
Models, Parameters	Local Regression coefficients (conditioned on each test point)

$$\min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_i, x_0)[y_i - \alpha(x_0) - \beta(x_0)x_i]^2$$

$$\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0$$

(4) Regularized multivariate linear regression

Task	Regression
Representation	Y = Weighted linear sum of X' s
Score Function	Least-squares+ Regularization -norm
Search/Optimization	Normal Eq / SGD / GD
Models, Parameters	Regularized Regression coefficients

$$\min J(\beta) = \sum_{i=1}^{n}\left(Y - \hat{Y}\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

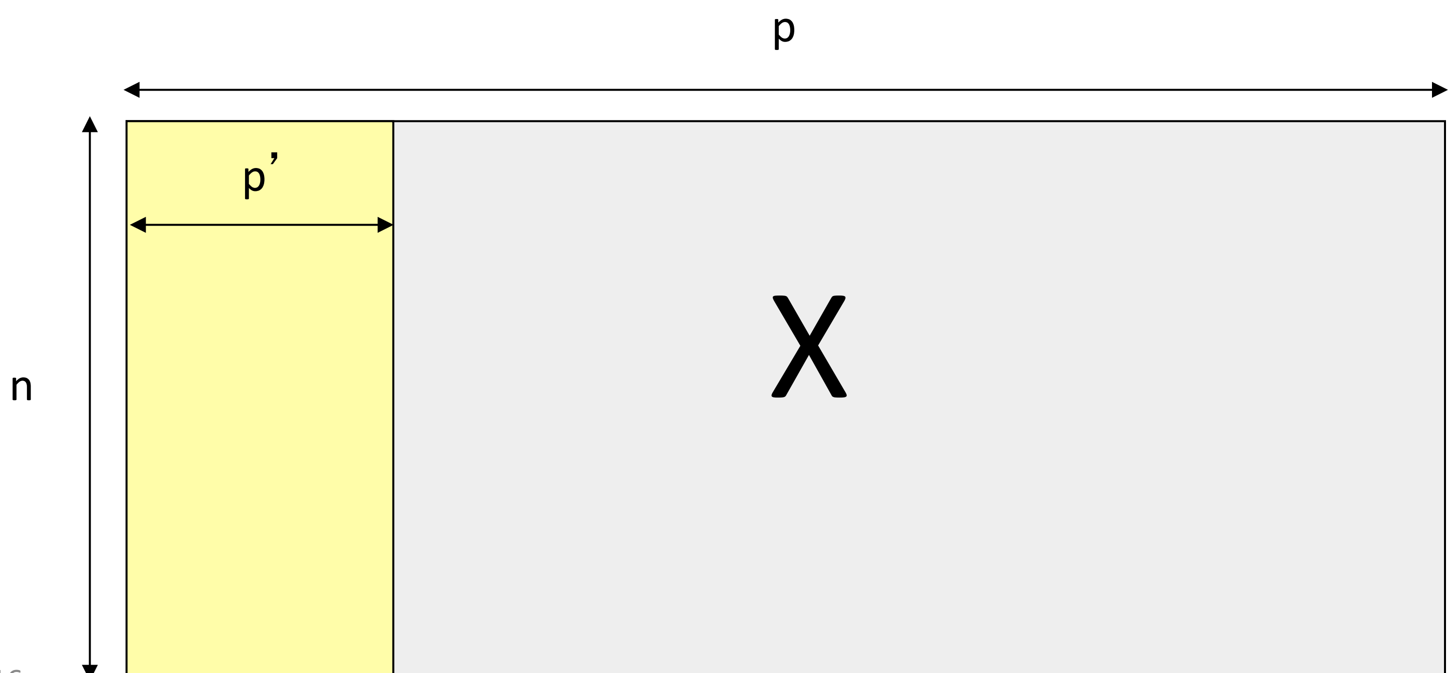
(5) Feature Selection

Task
↓
Representation
↓
Score Function
↓
Search/Optimization
↓
Models, Parameters

Dimension Reduction
↓
n/a
↓
Many possible options
↓
Greedy (mostly)
↓
Reduced subset of features

(5) Feature Selection

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



From Dr. **Isabelle Guyon**

Today

- ❑ Review of ML methods covered so far
  - ❑ Regression (supervised)
  - ❑ Classification (supervised)
  - ❑ Unsupervised models
  - ❑ Learning theory

- ❑ Review of Assignments covered so far

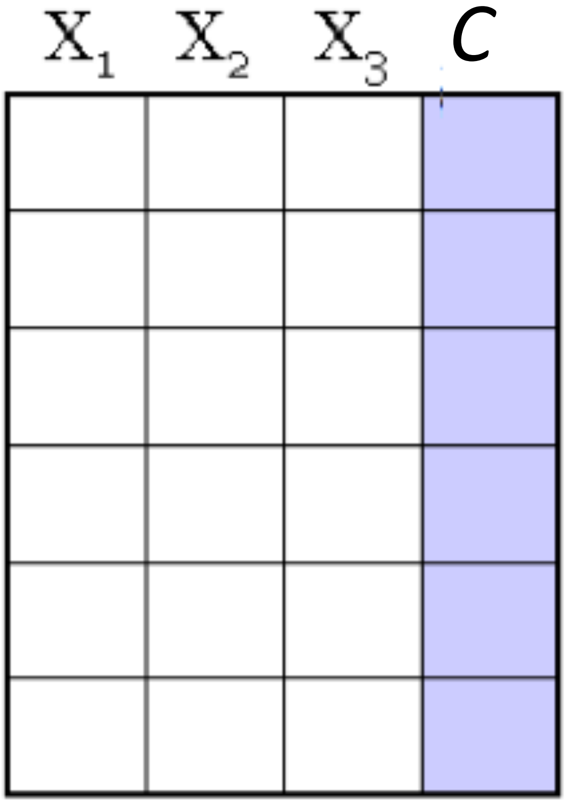
What we have covered (II)

- Supervised Classification models
  - Support Vector Machine
  - Bayes Classifier
  - Logistic Regression
  - K-nearest Neighbor
  - Random forest / Decision Tree
  - Neural Network (e.g. MLP)

Three major sections for classification

- We can divide the large variety of classification approaches into roughly three major types

1. Discriminative
   - directly estimate a decision rule/boundary
   - e.g., logistic regression, support vector machine, decisionTree
2. Generative:
   - build a generative statistical model
   - e.g., naïve bayes classifier, Bayesian networks
3. Instance based classifiers
   - Use observation directly (no models)
   - e.g. K nearest neighbors



A Dataset for classification

$$f : \boxed{X} \longrightarrow \boxed{C}$$

Output as Discrete Class Label

$C_1, C_2, \ldots, C_L$

- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/predictors/regressors*: [columns, except the last]
- **Target**/*outcome/response/label/dependent variable*: special column to be predicted [last column]

(1) Support Vector Machine

Task → Representation → Score Function → Search/Optimization → Models, Parameters

classification → Kernel Func K(xi, xj) → Margin + Hinge Loss (optional) → QP with Dual form → Dual Weights

$$K(\mathbf{x}, z) := \Phi(\mathbf{x})^T \Phi(z)$$

$$w = \sum_i \alpha_i x_i y_i$$

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 + C\sum_{i=1}^{n} \varepsilon_i$$

$$\text{subject to } \forall \mathbf{x}_i \in Dtrain : y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 1 - \varepsilon_i$$

$$\underset{k}{\operatorname{argmax}} P(C\_k \mid X) = \underset{k}{\operatorname{argmax}} P(X,C) = \underset{k}{\operatorname{argmax}} P(X \mid C)P(C)$$

(2) Bayes Classifier

Task	classification
↓	↓
Representation	Prob. models p(X|C)
↓	↓
Score Function	EPE with 0-1 loss, with Log likelihood(optional)
↓	↓
Search/Optimization	MLE
↓	↓
Models, Parameters	Prob. Models' Parameter

$$P(X_1, \cdots, X_p \mid C)$$

Gaussian Naive

$$\hat{P}(X_j \mid C = c_k) = \frac{1}{\sqrt{2\pi}\sigma_{jk}} \exp\left(-\frac{(X_j - \mu_{jk})^2}{2\sigma_{jk}^2} \right)$$

Bernoulli Naïve

$$p(W_i = true \mid c_k) = p_{i,k}$$

Multinomial

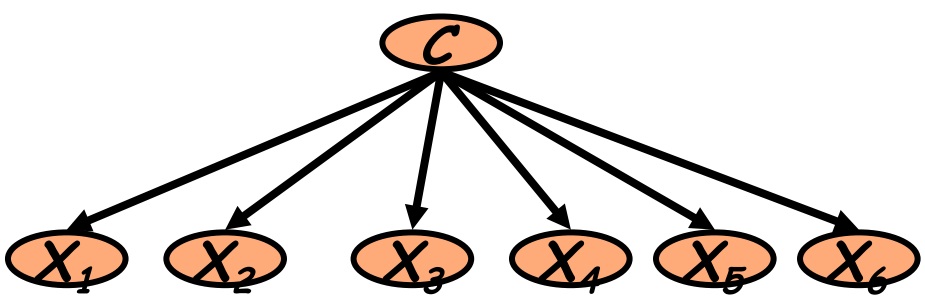
$$P(W_1 = n_1, \ldots, W_v = n_v \mid c_k) = \frac{N!}{n_{1k}!\, n_{2k}! \ldots n_{vk}!} \theta_{1k}^{n_{1k}} \theta_{2k}^{n_{2k}} \ldots \theta_{vk}^{n_{vk}}$$

Naïve Bayes Classifier

Difficulty: learning the joint probability $P(X_1, \cdots, X_p | C)$

- Naïve Bayes classification
  - Assumption that all input attributes are conditionally independent!

$$
\begin{aligned}
P(X_1, X_2, \cdots, X_p | C) &= P(X_1 | X_2, \cdots, X_p, C) P(X_2, \cdots, X_p | C) \\
&= P(X_1 | C) P(X_2, \cdots, X_p | C) \\
&= P(X_1 | C) P(X_2 | C) \cdots P(X_p | C)
\end{aligned}
$$



Adapt from Prof. Ke Chen NB slides

(3) Logistic Regression

Task → Representation → Score Function → Search/Optimization → Models, Parameters

classification → Log-odds(Y) = linear function of X's → EPE, with conditional Log-likelihood → Iterative (Newton) method → Logistic weights

$$P(c=1|x) = \frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}}$$

Logistic Regression—when?

Logistic regression models are appropriate for target variable coded as 0/1.

We only observe "0" and "1" for the target variable—but we think of the target variable conceptually as a probability that "1" will occur.

This means we use Bernoulli distribution to model the target variable with its Bernoulli parameter p=p(y=1|x) predefined.

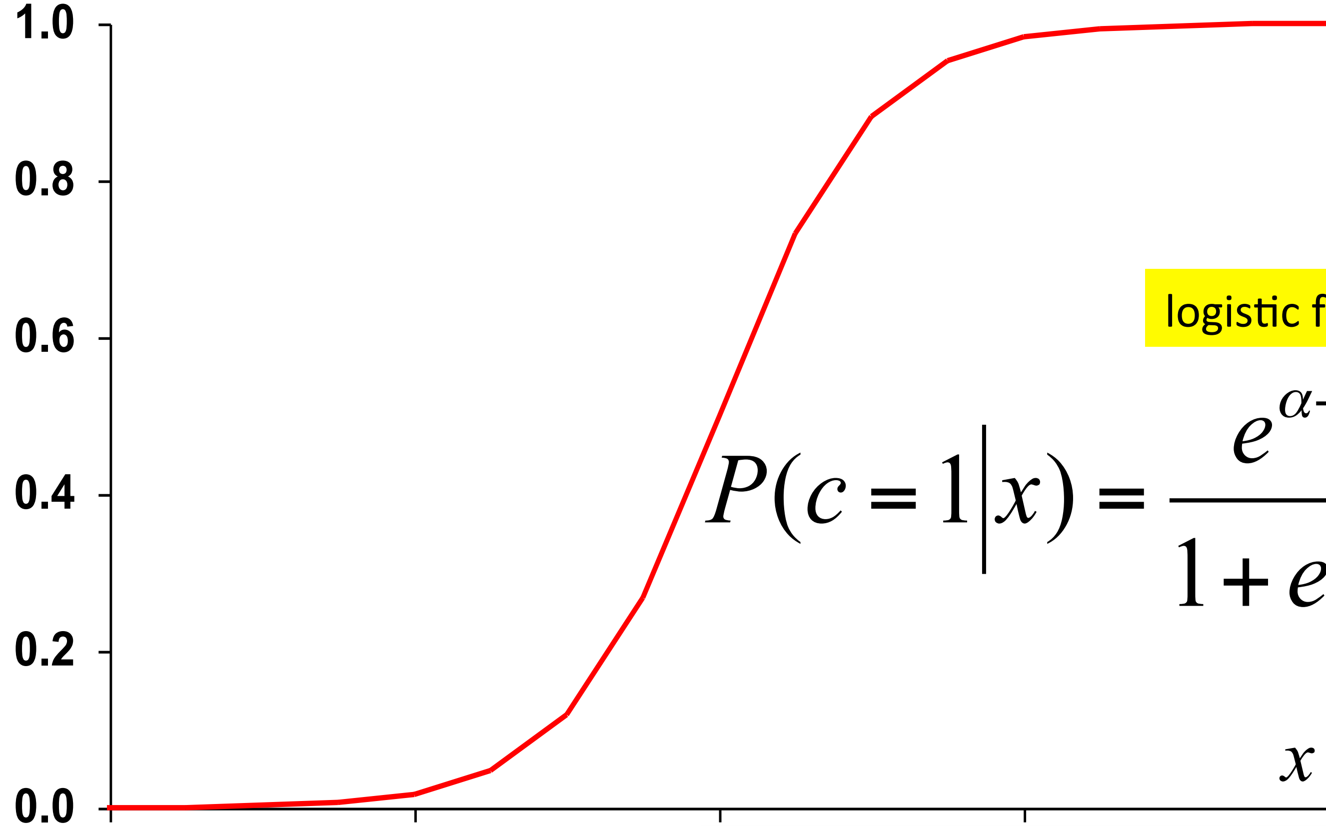
The main interest → predicting the probability that an event occurs (i.e., the probability that p(y=1|x)).

Discriminative

Logistic regression models for binary target variable coded 0/1.

P (C=1|X)

e.g. Probability of disease



logistic function

$$P(c=1|x)=\frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}}$$

Logit function

Decision Boundary → equals to zero

$$\ln\left[\frac{P(c=1|x)}{P(c=0|x)}\right]=\ln\left[\frac{P(c=1|x)}{1-P(c=1|x)}\right]=\alpha+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_p x_p$$

Discriminative vs. Generative

Generative approach

- Model the joint distribution $p(X, C)$ using

  $p(X \mid C = c_k)$ and $p(C = c_k)$

Class prior

Discriminative approach

- Model the conditional distribution $p(c \mid X)$ directly

e.g.,

$$\frac{1}{1 + e^{-(\beta_0 + \beta_1 * X))}}$$

Discriminative vs. Generative

- Empirically, generative classifiers approach their asymptotic error faster than discriminative ones
  - Good for small training set
  - Handle missing data well (EM)
- Empirically, discriminative classifiers have lower asymptotic error than generative ones
  - Good for larger training set

(4) K-Nearest Neighbor

Task	classification
Representation	Local Smoothness
Score Function	EPE with L2 loss ➔ conditional mean
Search/Optimization	NA
Models, Parameters	Training Samples

Nearest neighbor classification

- *k*-Nearest neighbor classifier is a lazy learner
  - Does not build model explicitly.
  - Unlike eager learners such as decision tree induction and rule-based systems.
  - Classifying unknown samples is relatively expensive.
- *k*-Nearest neighbor classifier is a local model, vs. global model of linear classifiers.

(5) Decision Tree / Random Forest

Task

Representation

Score Function

Search/Optimization

Models, Parameters

Classification

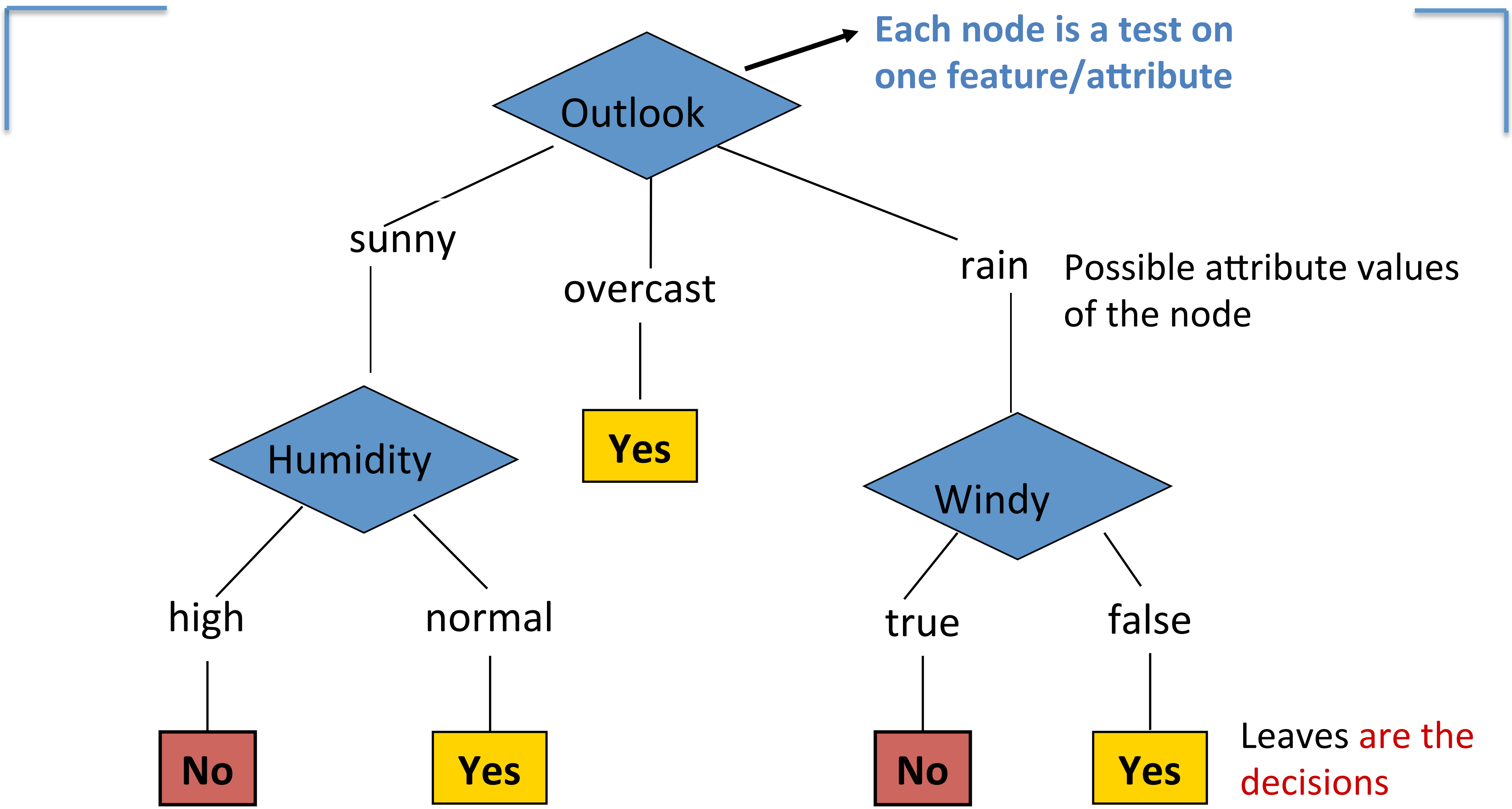
Partition feature space into set of rectangles, local smoothness

Greedy to find partitions

Split Purity measure / e.g. IG / cross-entropy / Gini /

Tree Model (s), i.e. space partition

Anatomy of a decision tree



Decision trees

- Decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.

- (Outlook ==overcast)
- OR
- ((Outlook==rain) and (Windy==false))
- OR
- ((Outlook==sunny) and (Humidity=normal))
- => yes play tennis

Information gain

- $IG(X\_i,Y)=H(Y)-H(Y|X\_i)$

Reduction in uncertainty by knowing a feature X_i

Information gain:

= (information before split) – (information after split)

= entropy(parent) – [average entropy(children)]

Fixed

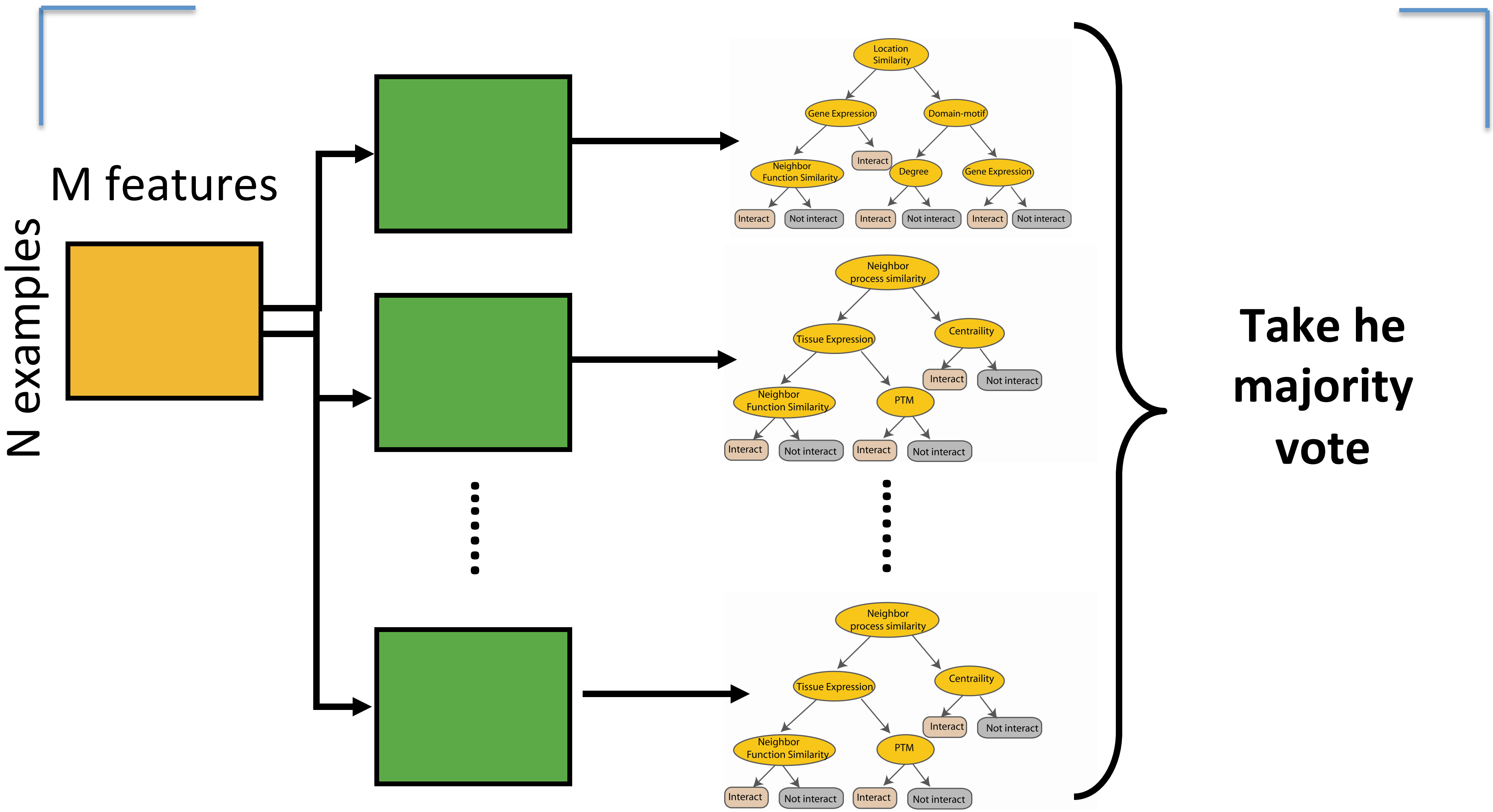
–

the lower, the better (children nodes are purer)

=

For IG, the higher, the better

Random Forest Classifier



(6) Neural Network

Task → Representation → Score Function → Search/Optimization → Models, Parameters

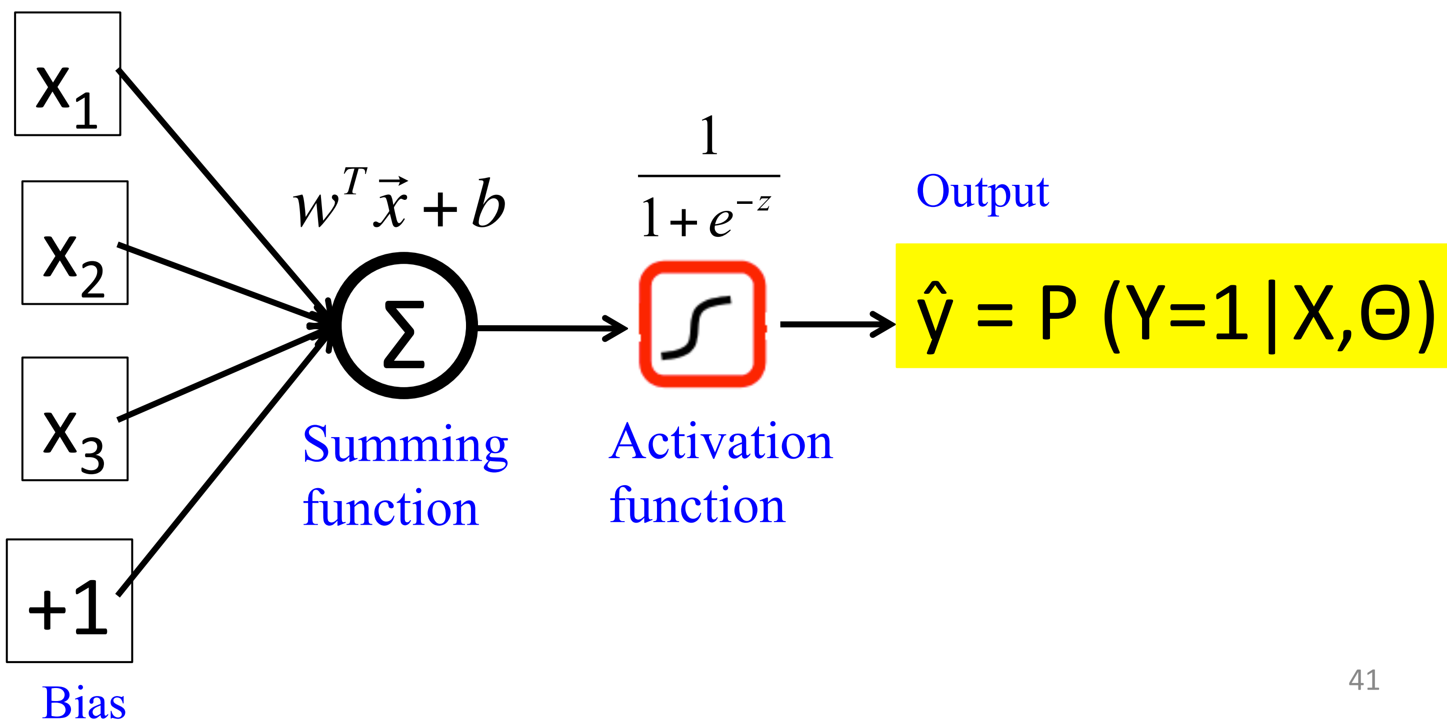
Classification / Regression → Multilayer Network topology → conditional Log-likelihood , Cross-Entropy / MSE → SGD / Backprop → NN network Weights

Logistic regression

Logistic regression could be illustrated as a module

On input x, it outputs ŷ:

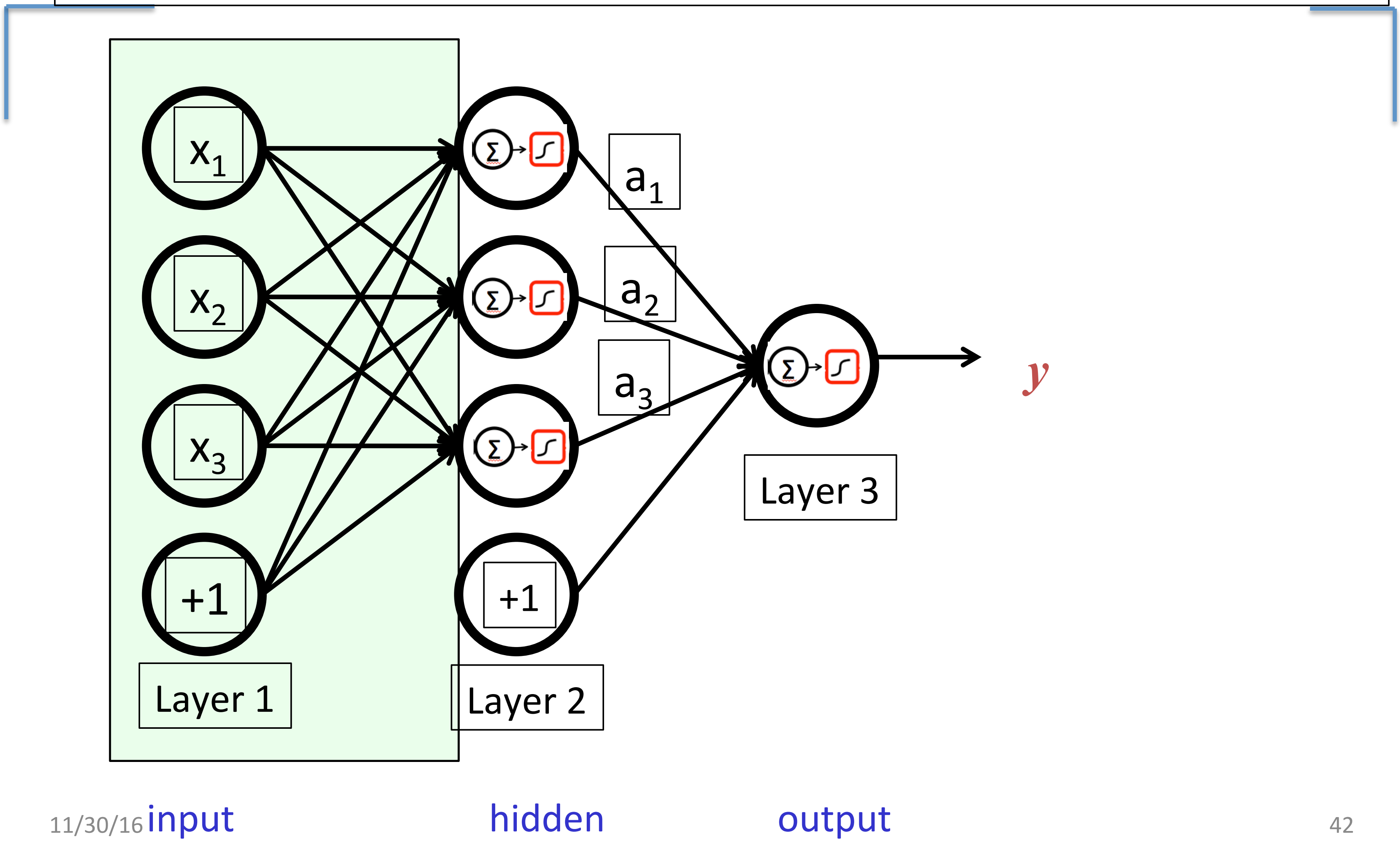
where



Draw a logistic regression unit as:

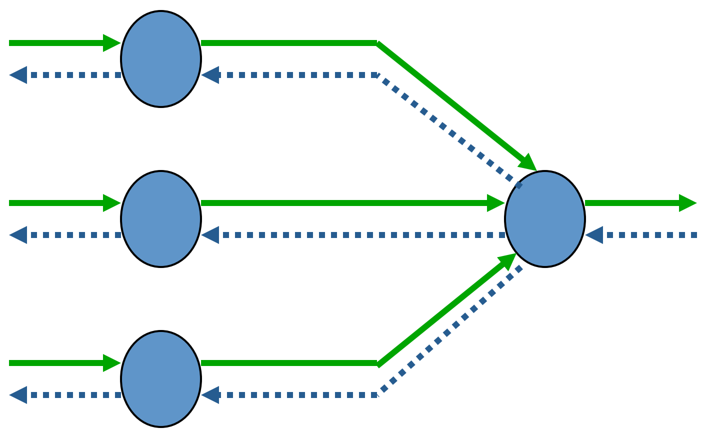
Multi-Layer Perceptron (MLP)

String a lot of logistic units together. Example: 3 layer network:



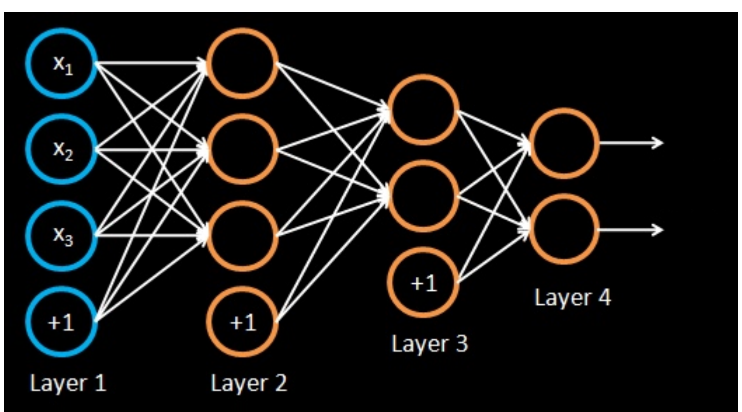
Backpropagation

- Back-propagation training algorithm

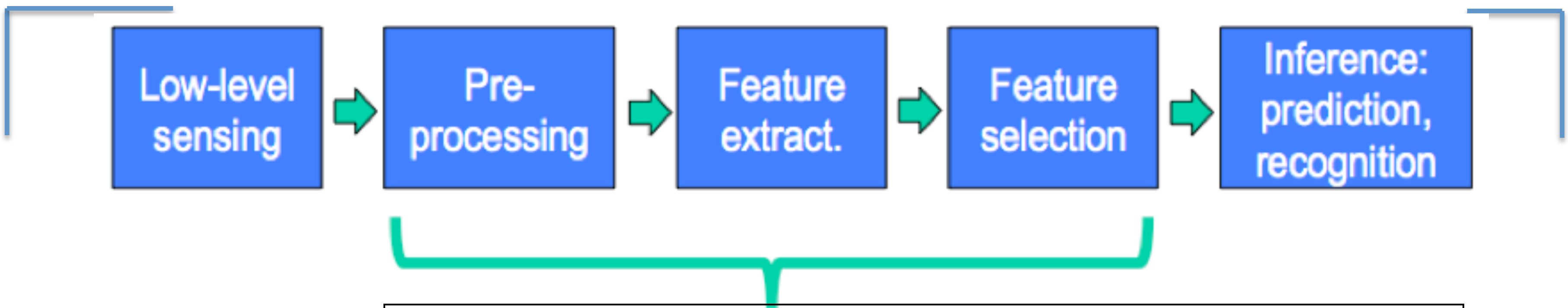


Network activation
Forward Step

Error propagation
Backward Step



Deep Learning Way: Learning features / Representation from data



Feature Engineering

- ✓ Most critical for accuracy
- ✓ Account for most of the computation for testing
- ✓ Most time-consuming in development cycle
- ✓ Often hand-craft and task dependent in practice

Feature Learning

- ✓ Easily adaptable to new similar tasks
- ✓ Layerwise representation
- ✓ Layer-by-layer unsupervised training
- ✓ Layer-by-layer supervised training

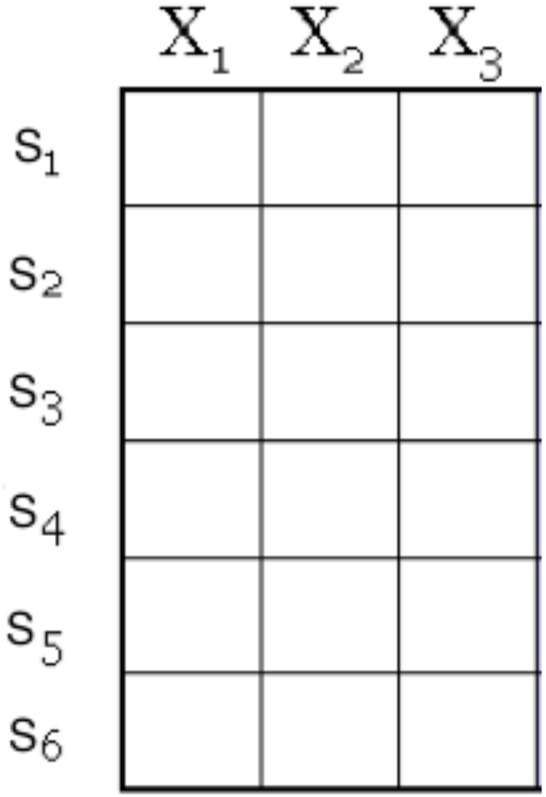
Today

- Review of ML methods covered so far
  - Regression (supervised)
  - Classification (supervised)
  - Unsupervised models
  - Learning theory

- Review of Assignments covered so far

What we have covered (III)

- Unsupervised models
  - Dimension Reduction (PCA)
  - Hierarchical clustering
  - K-means clustering
  - GMM/EM clustering



An unlabeled Dataset X

a data matrix of n observations on p variables $x_1,x_2,\ldots x_p$

Unsupervised learning = learning from raw (unlabeled, unannotated, etc) data, as opposed to supervised data where a label of examples is given

- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/predictors/regressors*: [columns]

(0) Principal Component Analysis

Task → Dimension Reduction

Representation → Gaussian assumption (Implicit)

Score Function → Direction of maximum variance

Search/Optimization → Eigen-decomp

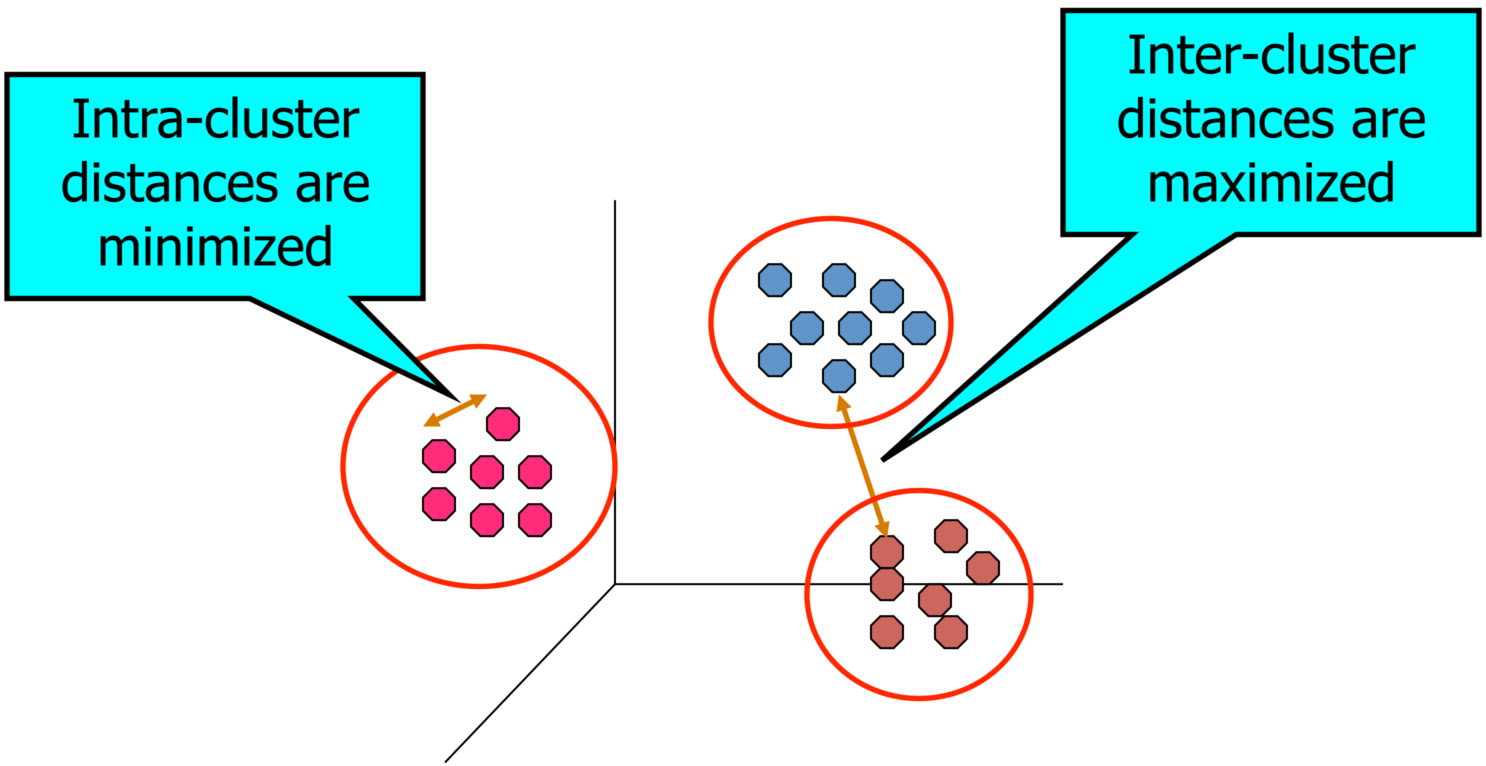
Models, Parameters → Principal components

What we have covered (III)

- Unsupervised models
  - Dimension Reduction (PCA)
  - Hierarchical clustering
  - K-means clustering
  - GMM/EM clustering

What is clustering?

- Find groups (clusters) of data points such that data points in a group will be similar (or related) to one another and different from (or unrelated to) the data points in other groups

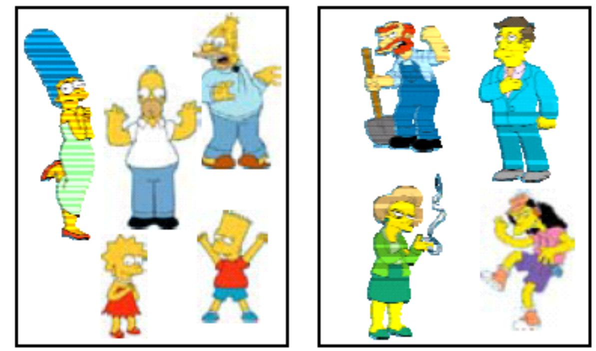


Issues for clustering

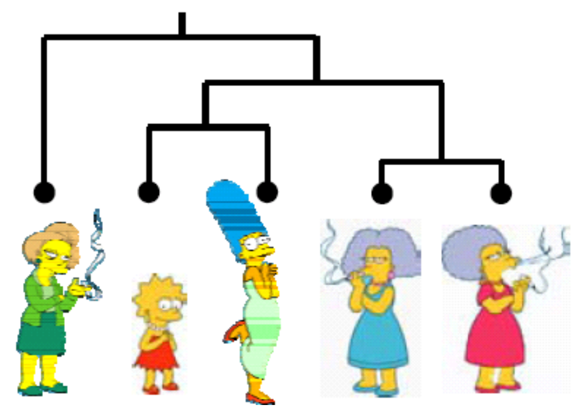
- What is a natural grouping among these objects?
  - Definition of "groupness"
- What makes objects "related"?
  - Definition of "similarity/distance"
- Representation for objects
  - Vector space? Normalization?
- How many clusters?
  - Fixed a priori?
  - Completely data driven?
    - Avoid "trivial" clusters - too large or small
- Clustering Algorithms
  - Partitional algorithms
  - Hierarchical algorithms
- Formal foundation and convergence

Clustering Algorithms

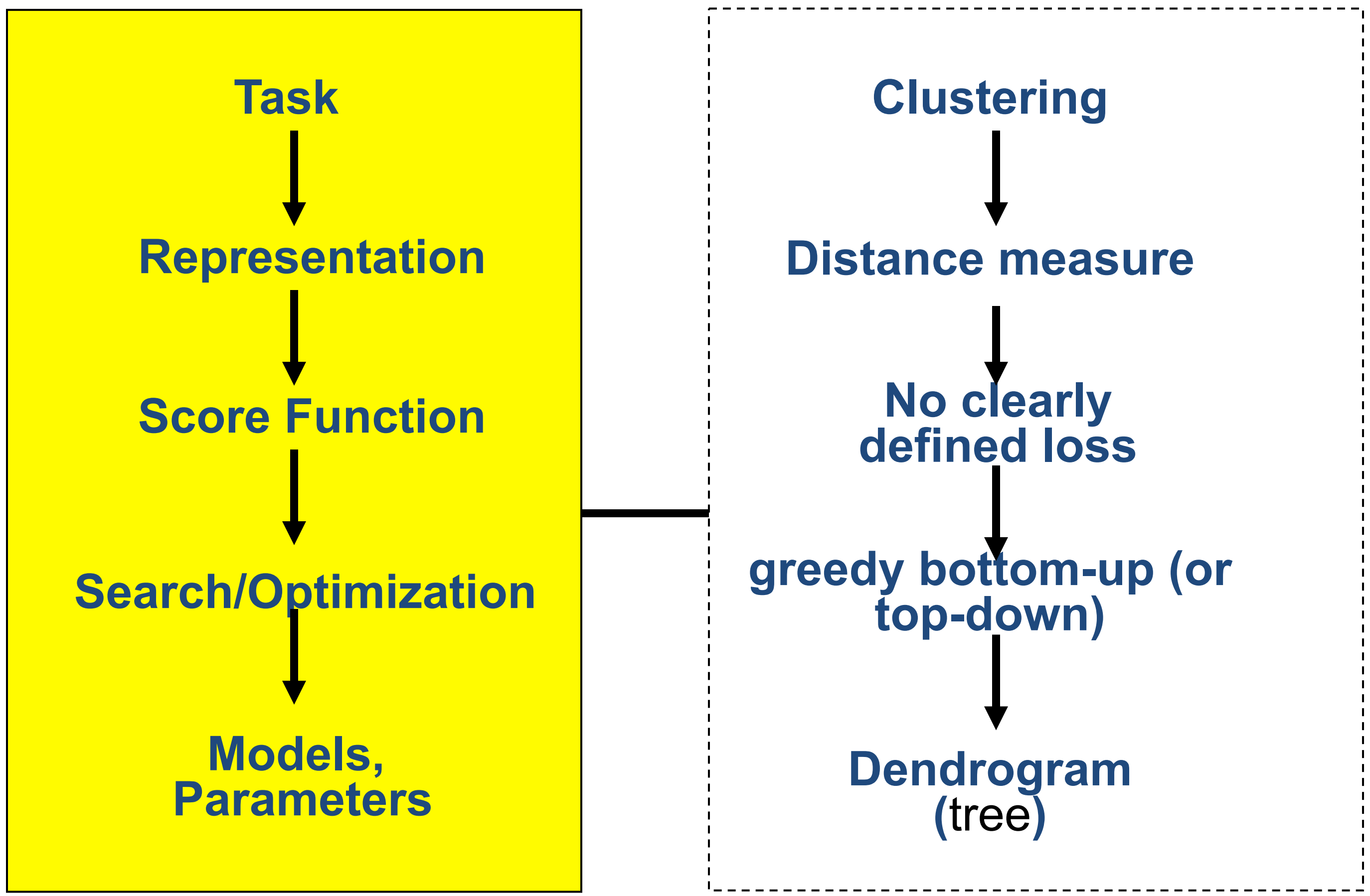
- Partitional algorithms
  - Usually start with a random (partial) partitioning
  - Refine it iteratively
    - K means clustering
    - Mixture-Model based clustering



- Hierarchical algorithms
  - Bottom-up, agglomerative
  - Top-down, divisive



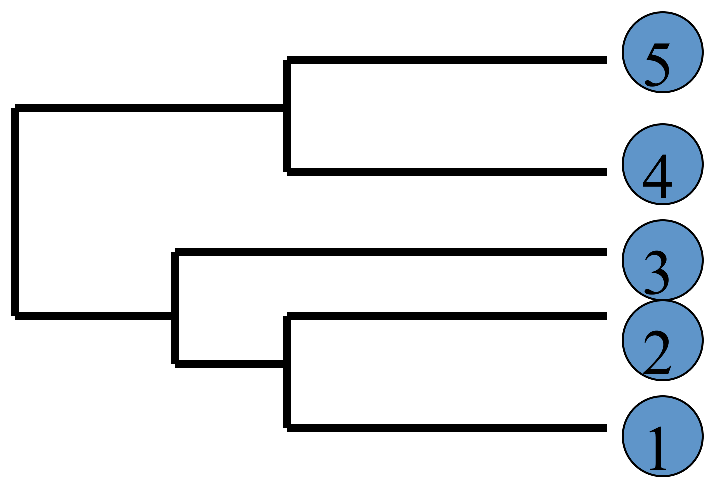
(1) Hierarchical Clustering



Example: single link

$$
\begin{array}{c c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix} 0 & & & & \\ 2 & 0 & & & \\ 6 & 3 & 0 & & \\ 10 & 9 & 7 & 0 & \\ 9 & 8 & 5 & 4 & 0 \end{bmatrix}
\end{array}
\Rightarrow
\begin{array}{c c}
 & \begin{array}{cccc} (1,2) & 3 & 4 & 5 \end{array} \\
\begin{array}{c} (1,2) \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix} 0 & & & \\ 3 & 0 & & \\ 9 & 7 & 0 & \\ 8 & 5 & 4 & 0 \end{bmatrix}
\end{array}
\Rightarrow
\begin{array}{c c}
 & \begin{array}{ccc} (1,2,3) & 4 & 5 \end{array} \\
\begin{array}{c} (1,2,3) \\ 4 \\ 5 \end{array} &
\begin{bmatrix} 0 & & \\ 7 & 0 & \\ 5 & 4 & 0 \end{bmatrix}
\end{array}
$$

$$d_{(1,2,3),(4,5)} = \min\{ d_{(1,2,3),4}, d_{(1,2,3),5} \} = 5$$



(2) K-means Clustering

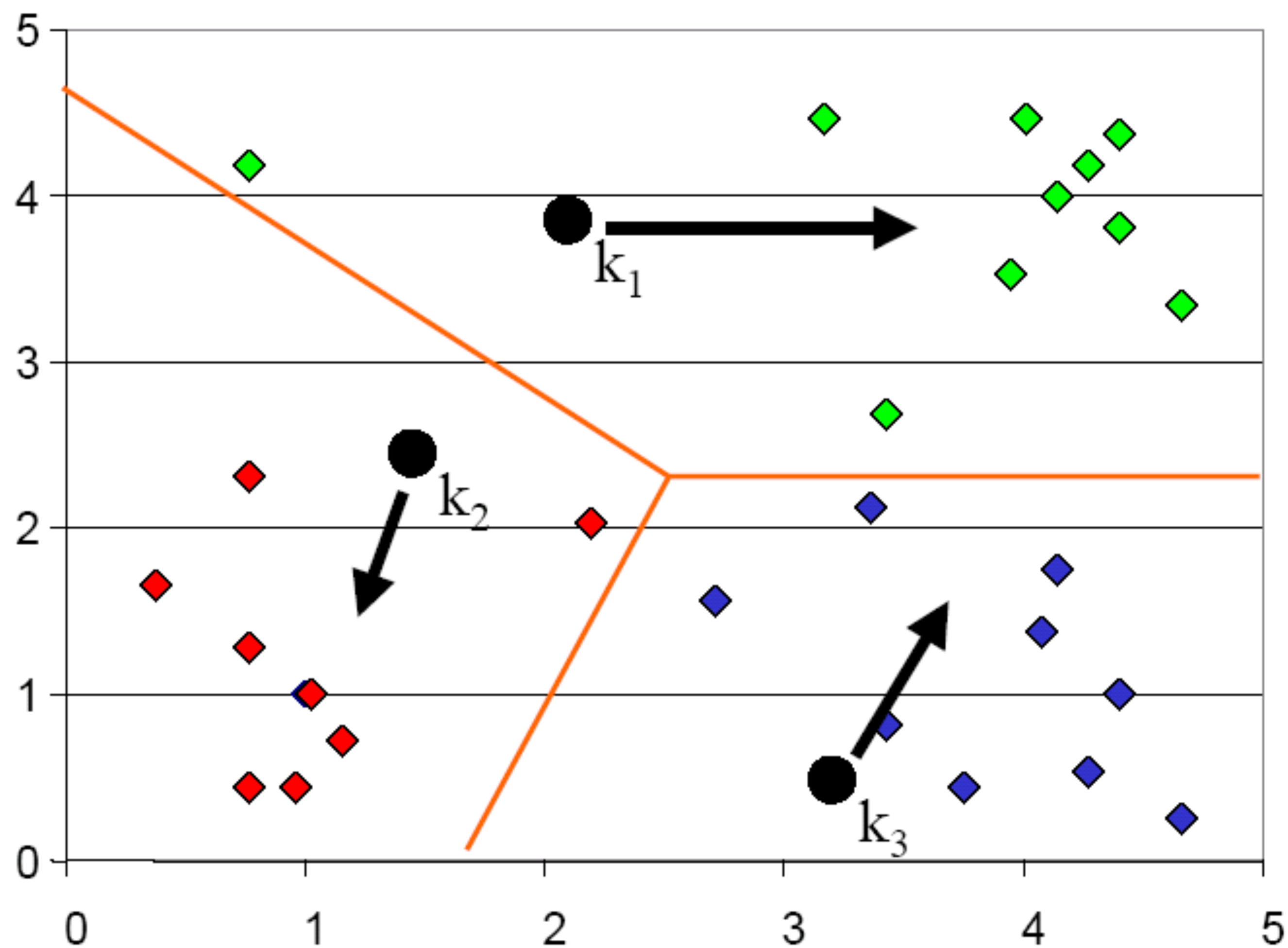
Task
↓
Representation
↓
Score Function
↓
Search/Optimization
↓
Models, Parameters

Clustering
↓
Spherical clusters

Gaussian with spherical

↓
Sum-of-square distance to centroid
↓
K-means (special case of EM) algorithm
↓
Cluster membership & centroid

K-means Clustering: Step 2
- Determine the membership of each data points



(3) GMM Clustering

Task ↓ **Representation** ↓ **Score Function** ↓ **Search/Optimization** ↓ **Models, Parameters**

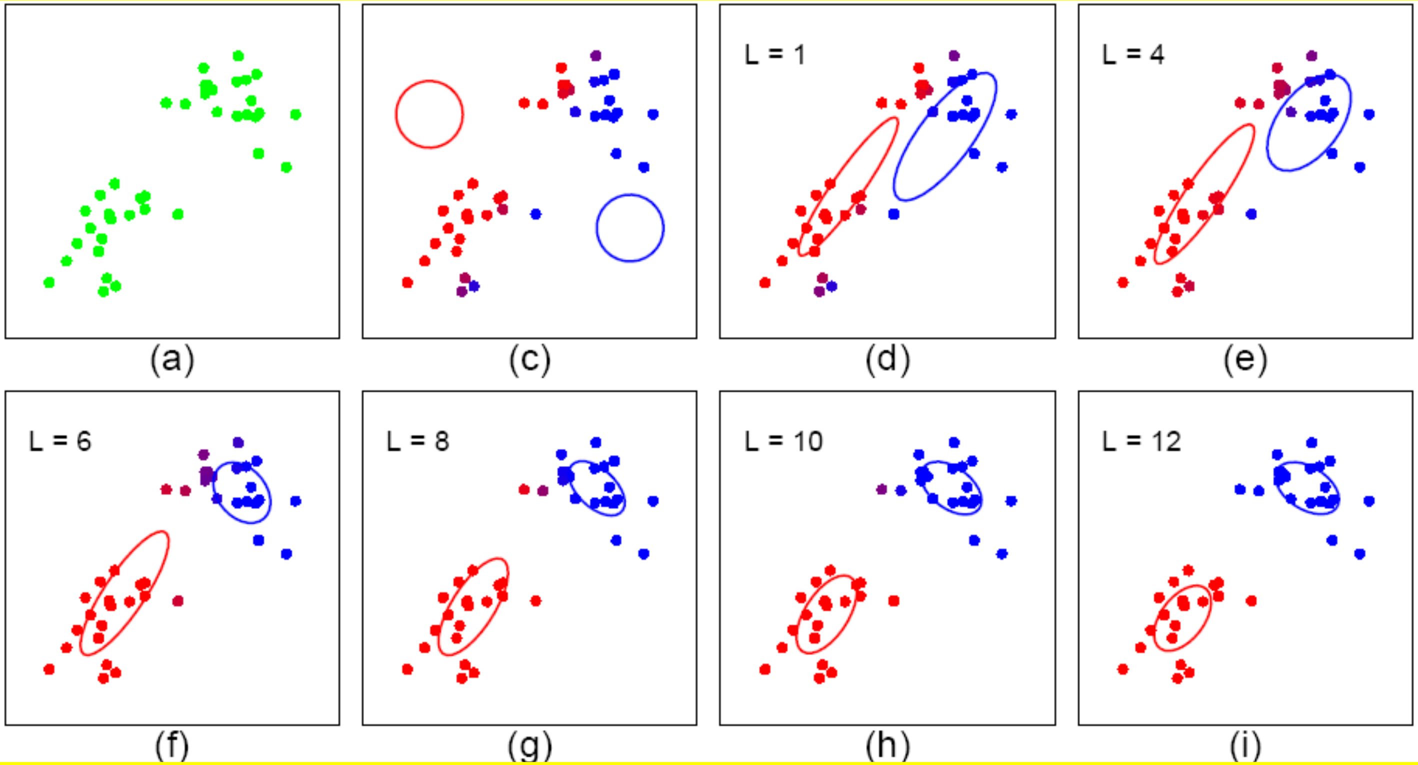
Clustering ↓ **Mixture of Gaussians** ↓ **Likelihood** ↓ **EM algorithm** ↓ **Each point's soft membership & mean / covariance per cluster**

$$\sum_i \log \prod_{i=1}^{n} p(x = x_i) = \sum_i \log \left[\sum_{\mu_j} p(\mu = \mu_j) \frac{1}{(2\pi)^{p/2} \left|\Sigma_j\right|^{1/2}} e^{-\frac{1}{2}\left(\vec{x}-\vec{\mu}_j\right)^T \Sigma_j^{-1}\left(\vec{x}-\vec{\mu}_j\right)} \right]$$

Expectation-Maximization for training GMM

- Start:
  - "Guess" the centroid m_k and covariance S_k of each of the K clusters
- Loop

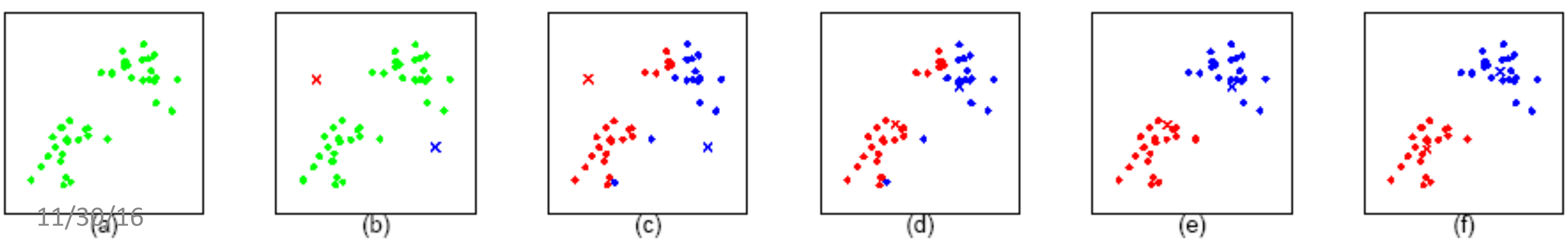
each cluster, revising both the mean (centroid position) and covariance (shape)



For each point, revising its proportions belonging to each of the K clusters

Compare: K-means

- The EM algorithm for mixtures of Gaussians is like a "soft version" of the K-means algorithm.
- In the K-means "E-step" we do hard assignment:
- In the K-means "M-step" we update the means as the weighted sum of the data, but now the weights are 0 or 1:



Today

- ❑ Review of ML methods covered so far
  - ❑ Regression (supervised)
  - ❑ Classification (supervised)
  - ❑ Unsupervised models
  - ❑ Learning theory
- ❑ Review of Assignments covered so far

What we have covered (IV)

- ❑ Learning theory / Model selection
  - – K-folds cross validation
  - – Expected prediction error
  - – Bias and variance tradeoff

CV-based Model Selection

We're trying to decide which algorithm / hyperparameter to use.

- We train each model and make a table...

i	f_i	TRAINERR	10-FOLD-CV-ERR	Choice
1	f_1			
2	f_2			
3	f_3			√
4	f_4			
5	f_5			
6	f_6			

Hyperparameter tuning ....

Which kind of cross-validation ?

	Downside	Upside
Test-set	Variance: unreliable estimate of future performance	Cheap
Leave-one-out	Expensive. Has some weird behavior	Doesn't waste data
10-fold	Wastes 10% of the data. 10 times more expensive than test set	Only wastes 10%. Only 10 times more expensive instead of R times.
3-fold	Wastier than 10-fold. Expensivier than test set	Slightly better than test-set
R-fold	Identical to Leave-one-out	

What we have covered (IV)

- Learning theory / Model selection
  - K-folds cross validation
  - Expected prediction error
  - Bias and variance tradeoff

Statistical Decision Theory

- Random input vector: X
- Random output variable: Y
- Joint distribution: $\Pr(X,Y)$
- Loss function $L(Y, f(X))$

- Expected prediction error (EPE):
- $$\mathrm{EPE}(f) = \mathrm{E}(L(Y,f(X))) = \int L(y,f(x))\Pr(dx,dy)$$

$$\text{e.g.} = \int (y - f(x))^2 \Pr(dx,dy)$$

e.g. Squared error loss (also called L2 loss)

Consider population distribution

Bias-Variance Trade-off for EPE:

$$\text{EPE}\left(\overset{f(x_0)}{\cancel{x\_0}}\right) = \text{noise}^2 + \text{bias}^2 + \text{variance}$$

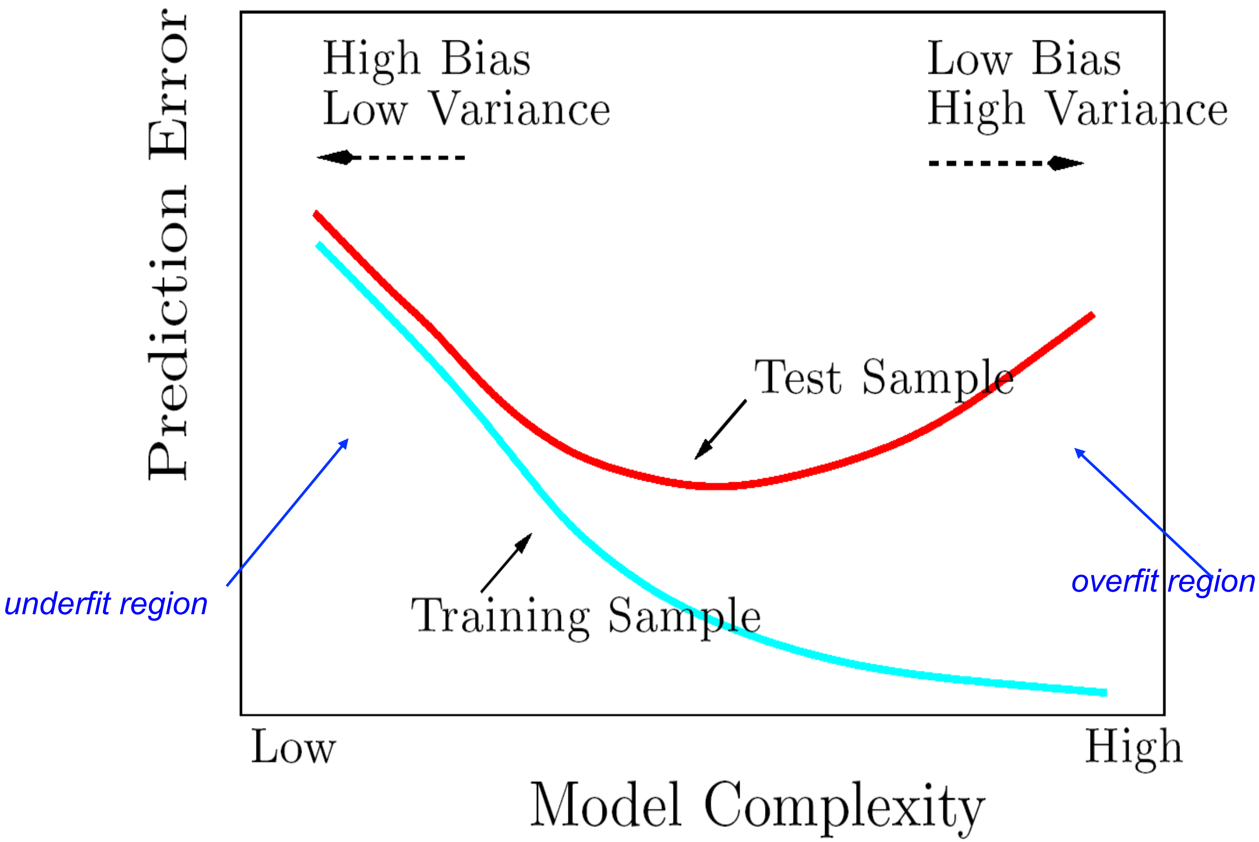
Unavoidable error

Error due to incorrect assumptions

Error due to variance of training samples

Slide credit: D. Hoiem

Bias-Variance Tradeoff / Model Selection



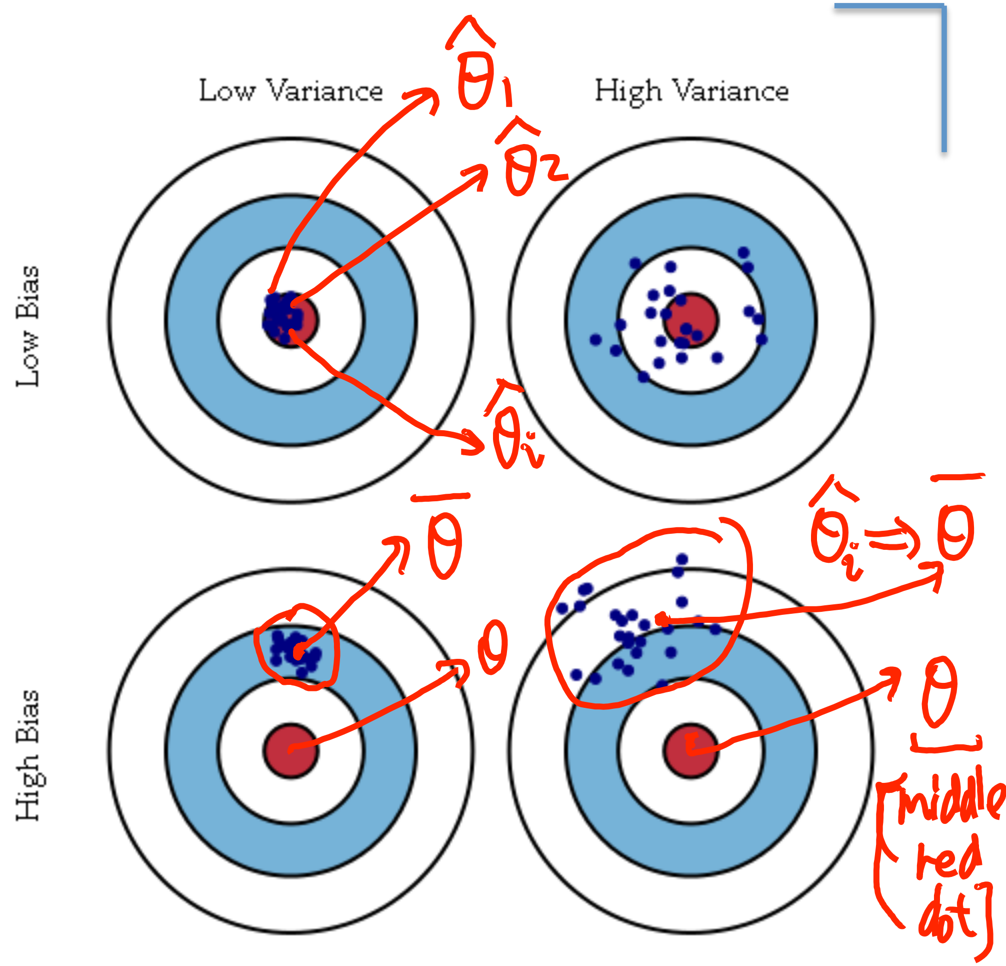
Model "bias" & Model "variance"

- Middle RED:
  - TRUE function θ [middle red]
- Error due to bias:
  - How far off in general from the middle red

$$E(\theta - \underbrace{\bar{\theta}}_{\text{mean of } \hat{\theta}})$$

- Error due to variance:
  - How wildly the blue points spread

$$E((\hat{\theta} - \bar{\theta})^2) \quad \hat{\theta} \rightarrow \{\hat{\theta}_1, \hat{\theta}_2, \hat{\theta}_3, \dots\} \text{ Blue dots}$$



need to make assumptions that are able to generalize

- Components of generalization error
  - **Bias:** how much the average model over all training sets differ from the true model?
    - Error due to inaccurate assumptions/simplifications made by the model
  - **Variance:** how much models estimated from different training sets differ from each other
- **Underfitting:** model is too “simple” to represent all the relevant class characteristics
  - High bias and low variance
  - High training error and high test error
- **Overfitting:** model is too “complex” and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error

Slide credit: L. Lazebnik

Today

- ❑ Review of ML methods covered so far
  - ❑ Regression (supervised)
  - ❑ Classification (supervised)
  - ❑ Unsupervised models
  - ❑ Learning theory
- ❑ Review of Assignments covered so far

Machine Learning in a Nutshell

Task

↓

Representation

↓

Score Function

↓

Search/Optimization

↓

Models, Parameters

ML grew out of work in AI

Optimize a performance criterion using example data or past experience,

Aiming to generalize to unseen data

What we have covered for each component

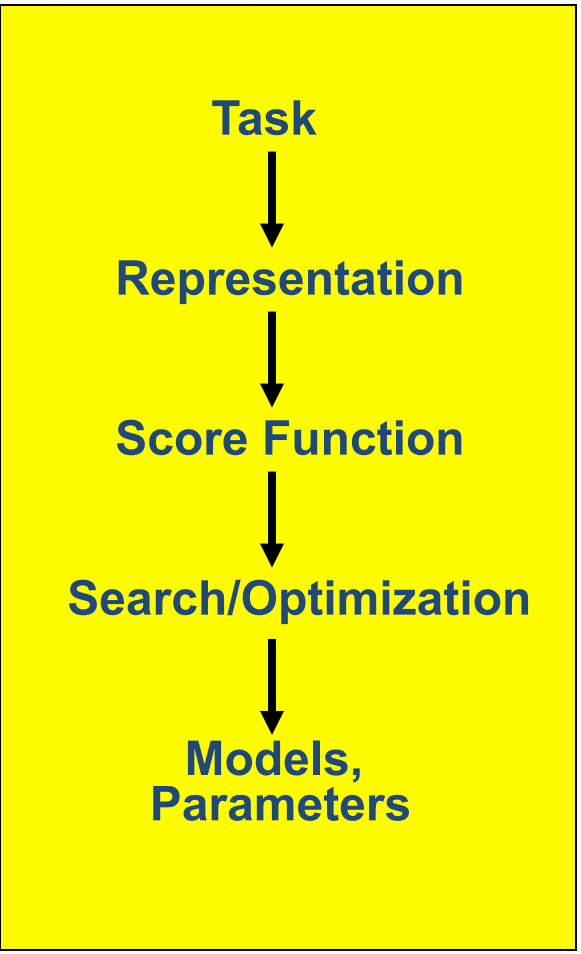
Task	Regression, classification, clustering, dimen-reduction
Representation	Linear func, nonlinear function (e.g. polynomial expansion), local linear, logistic function (e.g. p(c|x)), tree, multi-layer, prob-density family (e.g. Bernoulli, multinomial, Gaussian, mixture of Gaussians), local func smoothness, kernel matrix, local smoothness, partition of feature space,
Score Function	MSE, Margin, log-likelihood, EPE (e.g. L2 loss for KNN, 0-1 loss for Bayes classifier), cross-entropy, cluster points distance to centers, variance, conditional log-likelihood, complete data-likelihood, regularized loss func (e.g. L1, L2) ,
Search/ Optimization	Normal equation, gradient descent, stochastic GD, Newton, Linear programming, Quadratic programming (quadratic objective with linear constraints), greedy, EM, asyn-SGD, eigenDecomp, backprop
Models, Parameters	Linear weight vector, basis weight vector, local weight vector, dual weights, training samples, tree-dendrogram, multi-layer weights, principle components, member (soft/hard) assignment, cluster centroid, cluster covariance (shape), …

Today

- ❑ Review of ML methods covered so far
  - ❑ Regression (supervised)
  - ❑ Classification (supervised)
  - ❑ Unsupervised models
  - ❑ Learning theory

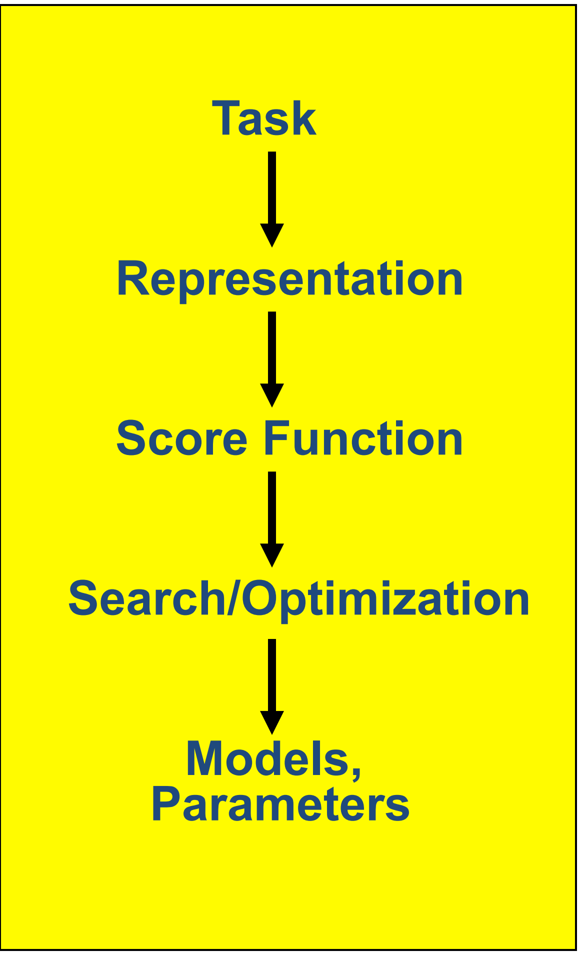
- ❑ Review of Assignments covered so far

HW1



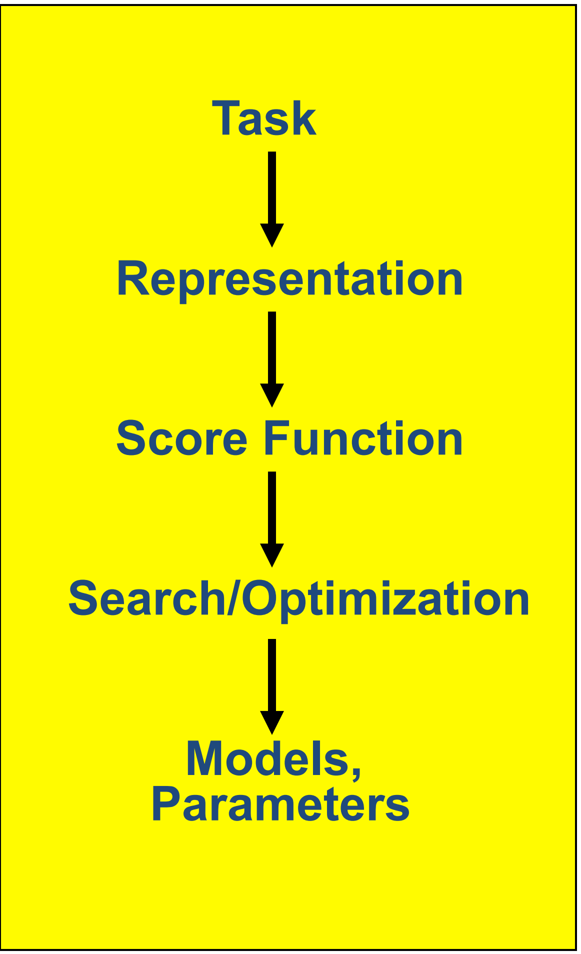
- Q1: Linear algebra review
- Q2: Linear regression + LOOCV
  - Regression
  - Evaluation pipeline
- Q3: Machine learning pipeline practice
  - Basic pipeline
  - GUI Toolbox
  - Evaluation

HW2



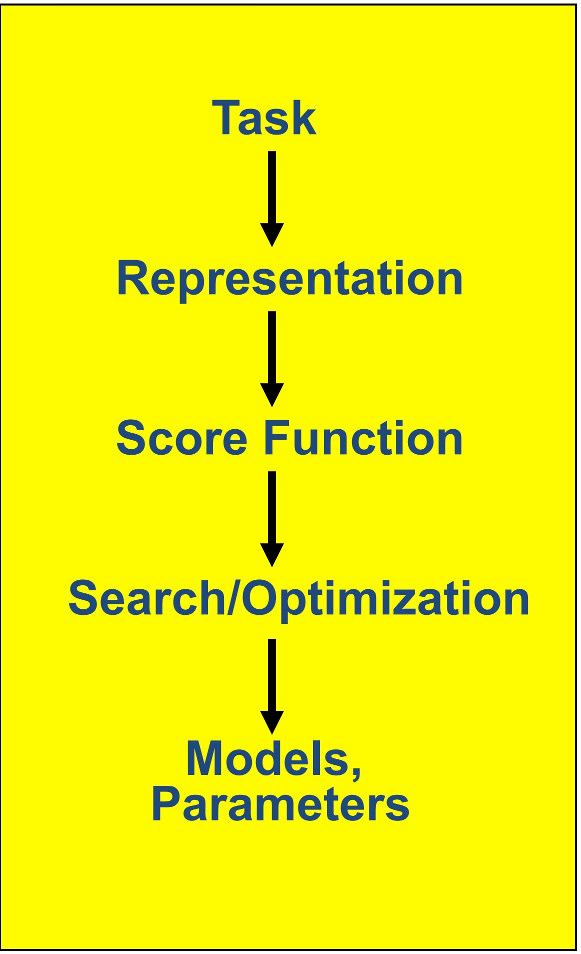
- Q1: Linear regression model fitting
  - Data loading
  - Basic linear regression
  - Three ways to train : Normal equation / SGD / Batch GD
  - Polynomial regression
- Q2: Ridge regression
  - Math derivation of ridge
  - Understand why/how Ridge
  - Model selection of Ridge with K-CV

HW3



- Q1: Support Vector Machines with Scikit-Learn
  - Data preprocessing
  - How to use SVM package
  - Model selection for SVM
  - Model selection pipeline with train-vali, or train-CV; then test

HW5



- Q1: Naive Bayes Classifier for Text-base Movie Review Classification
  - Preprocessing of text samples
  - BOW Document Representation
  - Multinomial Naive Bayes Classifier
    - BOW way
    - Language model way
  - Multivariate Bernoulli Naive Bayes Classifier

HW6

Task

↓

Representation

↓

Score Function

↓

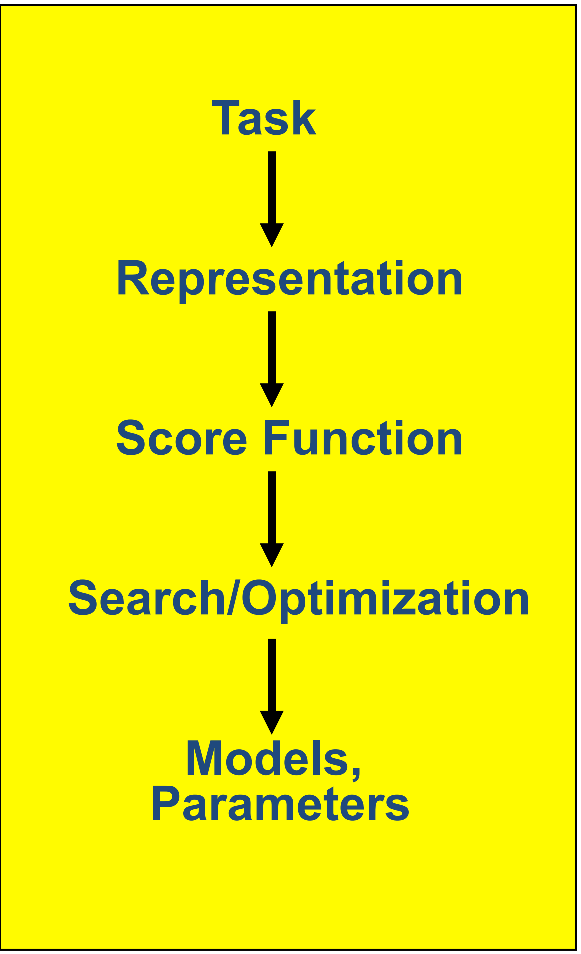
Search/Optimization

↓

Models, Parameters

- Q1: Neural Network Tensorflow Playground
  - Interactive learning of MLP
  - Feature engineering vs.
  - Feature learning
- Q2: Image Classification
  - Tool using
  - DT / KNN / SVM
  - PCA effect for image classification

HW6



- Q3: Unsupervised Clustering of audio data and consensus data
  - Data loading
  - K-mean clustering
  - GMM clustering
  - How to find K: knee-finding plot
  - How to measure clustering: purityMetric

References

- Hastie, Trevor, et al. The elements of statistical learning. Vol. 2. No. 1. New York: Springer, 2009.
- Prof. M.A. Papalaskar's slides
- Prof. Andrew Ng's slides