Framework for Learning

Formal: Spam Detection

- P(**x**, y) : distribution of
  - email messages **x** and
  - their true labels y (*spam* or *not spam*)
- **training sample**: set of email messages that have been labeled by the user
- **learning algorithm**: . . . this course. . .
- *f*: classifier produced by learning alg
- **test point**: A new email message **x**
  (with a true, but hidden, label y $\in$ { *spam, not_spam*})
- loss function L(ŷ ; y) :

predicted label $\hat{y}$	true label y	
	spam	not spam
spam	0	10
not spam	1	0

Approaches to Learning Classifier

1. Learn classifier directly:
   $f : X \rightarrow Y$
   where Y is discrete
   -- e.g., Y = {+1, –1 }
2. Learn regression function:
   $r : X \rightarrow \Re$
   c1 if $r(x) > \lambda$ c2 if $r(x) \geq \lambda$
3. Learn conditional distribution:
   $P(y \mid \mathbf{x})$
4. Learn joint distribution:
   $P(y, x)$

Linear Models:

(a) Learn classifier: Perceptron Algorithm

(b) Learn regression function: LMS

(c) Learn conditional distribution: Logistic Regression

(d) Learn joint distribution: Linear discriminant analysis

Inferring Classifier *f* from P(y | x)

- Given **x**, predict class y that minimizes the expected loss:

  $f(x) = \text{argmin}_{\hat{y}} E_{y|\mathbf{x}}[L(\hat{y}, y)] = \text{argmin}_{\hat{y}} \sum_y P(y | \mathbf{x}) L(\hat{y}, y)$

- Eg: For specific email message **x**

  $P(y = \text{spam} | \mathbf{x}) = 0.6 \quad P(y = \text{not\_spam} | \mathbf{x}) = 0.4$

  (Note $P(y = \text{spam} | \mathbf{x}) > P(y = \text{not\_spam} | \mathbf{x})$)

- What is optimal prediction $\hat{y}$?
  - Expected loss of $\hat{y}$ = spam : 0 x 0.6 + 10 x 0.4 = 4
  - Expected loss of $\hat{y}$ = not spam: 1 x 0.6 + 0 x 0:4 = 0.6

  $\Rightarrow$ optimal prediction is “not spam”

“Bias” of Learning Algorithms

- Learning algorithm embodies some “bias"
  to prefer one hypothesis over another
  . . . ideally: matched to assumptions/environment
- Two types of bias:
  - **restriction bias** or **language bias**
    Specifies what hypothesis space is searched
    (Eg, Gaussian, Mixture of Gaussians, DecisionTrees,
    Piece-wise linear functions, . . .)
  - **preference bias** or **search bias**
    specifies how hypothesis space is explored
    $\Rightarrow$ leads to different (first) answer
- Tradeoff: Suppose A $\subset$ B
  +: B more likely to include correct hypothesis
  –: B more likely to include INcorrect hypothesis
  (Size of H, VCDimension of H)

Terminology

- **Labeled example**: Example of form [x, f(x)]
- **Labeled sample**: Set of {[x_i, $f(x_i)$]}
- **Classifier**: Discrete-valued function.

  Possible values $f(x) \in \{ 1, \ldots, K\}$ called “classes"; “class labels”

  Binary classification: $f(x) \in \{ +1, -1 \}$
- **Concept**: Boolean function.
  - x s.t. f(x) = 1 called “positive examples”
  - x s.t. f(x) = 0 called “negative examples”
- **Target function** (**target concept**): “True function" *f* generating the labels
- **Hypothesis**: Proposed function *h* believed to be similar to *f*.
- **Hypothesis Space**: Space of all hypotheses that can, in principle, be output by a learning algorithm

Key Issues in Machine Learning

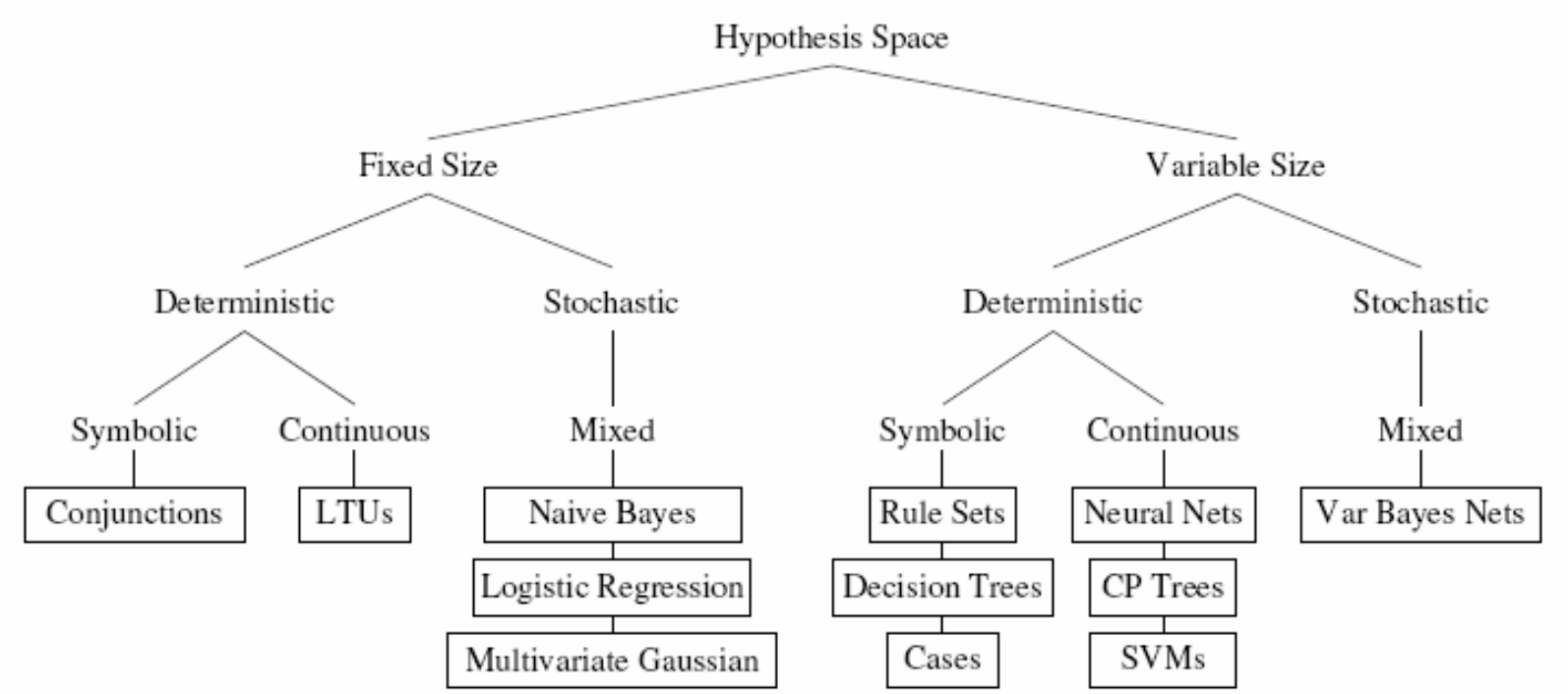
- **What are good hypothesis spaces?**
  - Which spaces are useful in practical applications; why?
- **What algorithms can work with these spaces?**
  - ∃ general design principles for machine learning alg's?
- **How can we optimize accuracy on future data points?**
  - Avoiding “overfitting”
- **How can we have confidence in results?**
  - How much training data is required to find accurate hypotheses? (statistical question)
- **Are some learning problems computationally intractable?** (computational question)
- **How can we formulate application problems as machine learning problems?** (engineering question)

Framework for Hypothesis Spaces

- Size: Is size of hypothesis space flat or stratified?
  - Flat spaces: easier to understand
  - Stratified spaces are generally more useful.
  - Stratified spaces introduce “overfitting".
- Randomness: Is each hypothesis deterministic or stochastic?
  - Affects hypotheses evaluation:
  - Deterministic: training example either consistent or inconsistent
  - Stochastic: training example either more/less likely
- Parameterization: Is each hypothesis described by symbolic (discrete) choices, or by continuous parameters (or both)?

  Typically. . .find
  - discrete parameters by combinatorial search;
  - continuous parameters by numerical search.

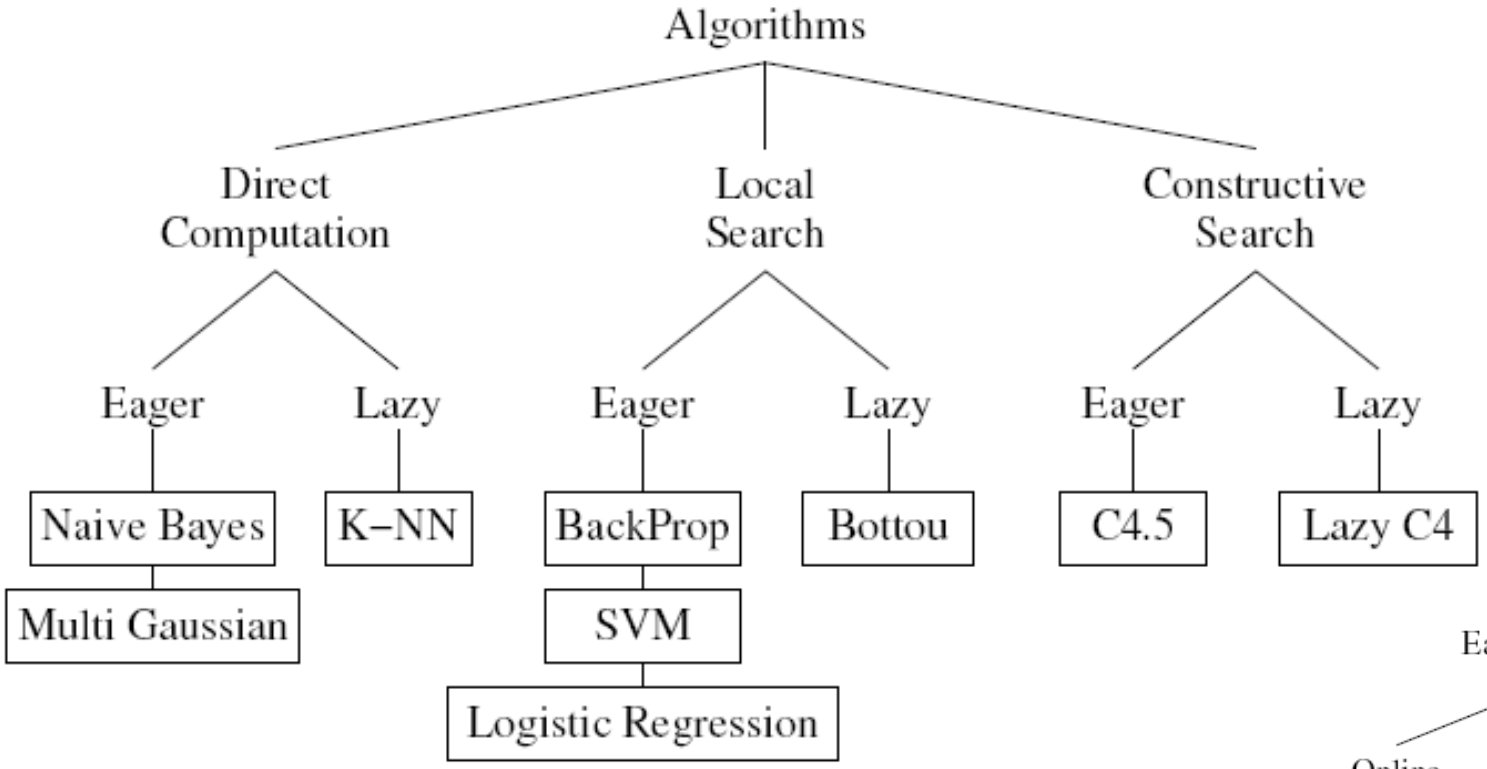
Framework for Hypothesis Spaces (2)

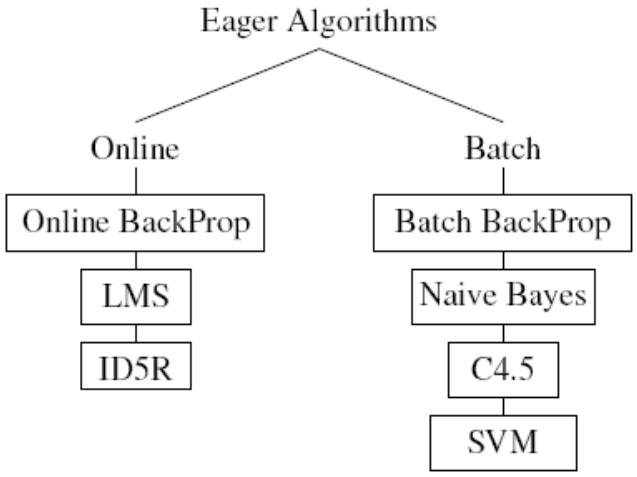


Framework for Learning Algorithms

- **Search Procedure**
  - **Direction Computation**: solve for hypothesis directly.
  - **Local Search**: start with initial hypothesis, make small improvements until a local optimum.
  - **Constructive Search**: start with empty hypothesis, gradually add structure to it until local optimum.
- **Timing**
  - **Eager**: Analyze training data and construct an explicit hypothesis.
  - **Lazy**: Store training data and wait until a test data point is presented, then construct ad hoc hypothesis to classify that one data point.
- **Online vs. Batch** (for eager algorithms)
  - **Online**: Analyze each training example as it is presented.
  - **Batch**: Collect training examples, analyze them, output an hypothesis.

Framework for Learning Alg's (2)





Three Types of Learning Tasks

- Regression: $f(\mathbf{x}) \in \Re$

- Density Estimation: $f(\mathbf{x}) \in [0, 1]$

  ... where $\sum_{\mathbf{x}} f(\mathbf{x}) = 1$

  or $\forall \mathbf{x} \; \sum_y f(y, \mathbf{x}) = 1$

Evaluating Any Classifier

- Natural handling of **"mixed" data types**
  - Continuous, ordered discrete, unordered discrete
- Handling of **missing values**
- **Robustness** to outliers in input space
- Insensitive to **monotone transformations** of inputs
- Computational **scalability** for large data sets
- Ability to deal with **irrelevant inputs**
- Ability to extract **linear combinations** of features
- **Interpretability**
- **Predictive power**

Handling Mixed Data Types

- **Indicator Variables**
  - *gender*: Convert to 0/1 variable.
  - *county of residence*: Introduce a 0/1 variable for each county
- Ordered discrete variables

  Eg: {small, medium, large}, {mild, moderate, severe}
  - Treat as unordered
  - Treat as real-valued
    - Can sometimes to measure “distances" between discrete terms.

      (Eg: How often is one value mistaken for another?)
    - Can combine “distances" with “multi-dimensional scaling" to assign real values

Missing Values: Two cases

- **Missing at random**: Independent errors cause certain features to be missing.
  - Clouds prevent satellite from seeing ground
  - Data transmission (wireless network connection) is lost
- **Missing for cause**:
  - Physician decided not to perform a particular medical measurement
  - Fail to record very large or very small values.
  - Human subjects systematically refuse to answer personal questions

Dealing with Missing Values

- **Missing at random**:
  - “Generative models" (learning P(x; y)) can produce a model of P(x) even when some features are not measured.
  - Can apply EM to “fill in" missing features
  - Replace each missing value by
    - its average value or
    - its most likely value

      (either within class y or after pooling all classes)
  - ∃ specialized methods for decision trees.
- **Missing for cause**:
  - Should model the causes of missingness; then fit the combined model.
  - Treat “missing" as a separate value
    - easy if feature is discrete,
    - if real-valued feature, introduce indicator feature

      (1 iff feature was measured)

Comments wrt Missing Data

- Special algorithms for missing data
  (eg, version of Nearest Neighbor that just sets each [?; x] pair to Max value)
- Just ignore tuples with missing data
  - . . . either completely or partially
- Imputation: fill in single value for each missing value x.v
  - . . . based on values of other attributes x.w
  - . . . based on (stochastic) relations learned from other tuples
  - May impute many completed datasets (“multiple imputation") at one time
    . . . or sequentially – related to Gibbs, MCMC, . . .
- Explicitly compute DISTRIBUTION over (each) missing value
  - $\Rightarrow$EM
- Types of missing-ness
  - MCAR: missing completely at random
    … random benign coin-flips, uncorrelated with value of attribute, ...
  - MAR: can depend on values of other
  - NMAR

Robust to Outliers in Input Space

- **LMS**: Any outlier has strong impact on squared error objective function
  - $\Rightarrow$ not robust.
- **Logistic Regression**: Outliers far from the decision boundary have little impact.
- **Multivariate Gaussian Models/LDA**: Outliers will have a strong impact on the models of $P(x \mid y)$

Scaling Criteria

- **Monotone Scaling**:
  Linear classifiers sensitive to non-linear transformations of the inputs, as non-linear xfor may make data less linearly separable.

- **Computational Scaling**: All linear methods scale well to large data sets.
  - Gaussian models scale $O(mn^3)$
    m = #examples,
    n = #attributes
  Others: O(mn)

Remaining Criteria

- **Ability to Deal with Irrelevant Inputs**:
  - Theory: irrelevant inputs receive 0 weights
  - Practice: irrelevant features cause trouble !
  - Multivariate Gaussian methods are worst

    … matrix becomes singular⇒ cannot be inverted!

    . . . but see \regularization".
- **Ability to Extract Linear Combinations of Features**:
  - Linear methods work extracting linear combinations of input features!
- **Interpretability**:
  - All easy to interpret.
- **Predictive power**:
  - For small data sets, these methods often perform best.

Scoring Linear Methods: LMS, Logistic regression, LDA

Criterion	LMS	Logistic	LDA
Mixed data	No	No	No
Missing values	No	No	Yes
Outliers	No	Yes	No
Monotone transformations	No	No	No
Scalability	Yes	Yes	Yes
Irrelevant inputs	No	No	No
Linear combinations	Yes	Yes	Yes
Interpretable	Yes	Yes	Yes
Predictive power	Yes	Yes	Yes