(Binary) Classification: Learning a Class from labeled Examples

- "Things" represented by a feature vector $\mathbf{x}$ and a label $r$ (also called $y$), often $r \in \{1,0\}$ or $\{+,-\}$
- Domain $D$ is set of all possible feature vectors
- A Hypothesis (sometimes called a Concept) partitions the Domain into + and - regions
  - or just the + region,

Assumption: iid Examples

- Distribution of things and measurements defines some unknown (but fixed) $P(\mathbf{x})$ on domain $D$
- Target concept $C$ gives the "correct" labels, $C_{\mathbf{x}}$, as a function of the features, $\mathbf{x}$
- Find an $h$ in $\mathcal{H}$ from examples that is "close" to $C$
  - A loss function $l(r, r')$ measures error of predictions, usually $l(r, r') = 0$ if $r = r'$ and $l(r, r') = 1$ otherwise
  - Want to minimize $\int P(\mathbf{x}) l(C_{\mathbf{x}}, h(\mathbf{x}))$ -- probability of error for usual loss

Tasty Coffee example

- Objects are cups of coffee
- Measure strength and sugar
- Each measurement is a feature or attribute
- Other features? (cream, temperature, roast)
- Features numeric (precision? Accuracy?)
- Label (or class) is "+" (tasty) or "-" (not)
- Example is $(\mathbf{x}, y)$ pair, $\mathbf{x}$ in $\mathbb{R}^2$, $y \in \{+,-\}$

More Terminology

- Domain: set of all possible $\mathbf{x}$ vectors
- Concept: a boolean function on domain, a mapping from $\mathbf{x}$'s to "+1" and "-1", or "T" and "F"; or "+" and "-", or a subset of domain
- Target: the concept to be learned
- Hypothesis class/space: is the set of hypotheses (concepts) that can be output by a given learning algorithm
- Strength and sugar measured 0 to 10
- Domain has 121 different instances
- How to predict from these examples?

```
<table>
<thead>
<tr>
<th>strength</th>
<th>sugar</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>-</td>
</tr>
</tbody>
</table>
```

Version space: all concepts in hypotheses space consistent with training set.
- If hypoth. Space is all concepts, then version space evenly split on every unseen instance
- Need inductive bias (smaller hypothesis space), otherwise generalization is hopeless
- Assume "tasty coffee" is a rectangle in $\mathbb{R}^2$
- Rectangles in $\mathcal{H}$ are concepts $C$ for which there exists so that $C(x) = 1$ iff $c_1 \leq x_1 \leq c_2$

**Hypothesis class $\mathcal{H}$ of rectangles**

$h(0) = 1$ if $h$ classifies $x$ as positive
$0$ if $h$ classifies $x$ as negative

```
Hypothesis class $\mathcal{H}$ of rectangles
```

```
Error of $h$ on data $X$
$E(h|X) = \sum_{x \in X} h(x) \cdot r(x)
```

**Bad use of "( )"**
Key interplay

- Underlying pattern being learned
- Features available
- Hypothesis space
- Number of examples available

The trick is finding the right mix, but...

Triple Trade-Off

- There is a trade-off between three factors (Dietterich, 2003):
  - Complexity of $\mathcal{H}$, $c(\mathcal{H})$
  - Training set size, $N$
  - Generalization error, $E$, on new data

  - As $N^1$, $E_1$
  - As $c(\mathcal{H})$, first $E_1$ and then $E_1$

Model Selection & Generalization

- Learning is an ill-posed problem; data is not sufficient to find a unique solution
- The need for inductive bias, assumptions about $\mathcal{H}$
- Generalization: How well a model performs on new data - What we are really interested in!
- Overfitting: $\mathcal{H}$ more complex than $C$ or $f$
- Underfitting: $\mathcal{H}$ less complex than $C$ or $f$

Example

- Not a rectangle in $x_1, x_2$
- How to make it a rectangle?

- It is a rectangle in three dimensions: $x_1, x_2$, and $x_1^* x_2$

- Not a rectangle in $x_1, x_2$
- How to make it a rectangle?

- It is a rectangle in three dimensions: $x_1, x_2$, and $x_1^* x_2$
VC Dimension - Shattering

- \( N \) points can be labeled in \( 2^N \) ways as +/-.
- \( H \) shatters a set if:
  - for each labeling of the set there is an \( h \in H \) consistent with the labeling
- \( \text{VC}(H) = \text{size of a largest shattered set} \)

An axis-aligned rectangle shatters 4 points only!

VC Dimension

- Vapnik-Chervonenkis dimension is a measure of hypothesis space capacity
- VC-dim of rectangles in the plane is 4
- PAC (Probably approximately correct) bounds: if target concept in hypothesis class then any hypothesis in class consistent with \( O(\text{VC-dim} \ln 1/\alpha / \alpha) \) random examples usually has error \( \leq \alpha \)

Noise

- Data not always perfect
- Attribute noise
- Label noise
- Noise can model hypothesis space approximations

Domain

Multiple Classes, \( C_i \)

\( i = 1, \ldots, K \)

\[ X = [x_1, \ldots, x_N] \]

Train hypotheses \( h(x)/i, i = 1, \ldots, K \)

\[ h(x)/i = \begin{cases} 1 & \text{if } x \in C_i \cap \cap_{j \neq i} \overline{C_j} \\ 0 & \text{otherwise} \end{cases} \]
Regression

\[ X = \{ x' \}_t^N \]
\[ x' \in \mathbb{R}^d \]
\[ x' = f(x) + \epsilon \]
\[ E(\theta|X) = \frac{1}{N} \sum_{t=1}^N (y'_t - [Wx'_t + b])^2 \]

Estimating Errors

- To estimate generalization error, we need data unseen during training. We split the data as:
  - Training set (50%)
  - Validation set (25%) (is training good?)
  - Test (publication) set (25%)
- Resampling when there is few data (cross validation)

Supervised Learning as parameter estimation

Model (hypothesis class):
\[ g(x|\theta) \]
Loss function:
\[ E(\theta|X) = \sum_{t} L(\hat{y}_t, y_t) \]
Optimization procedure:
\[ \theta^* = \arg \min_{\theta} E(\theta|X) \]

Why Reduce Dimensionality?
- Reduces time complexity: Less computation
- Reduces space complexity: Less parameters
- Saves the cost of observing the feature
- Simpler models are more robust on small datasets
- More interpretable; simpler explanation
- Data visualization (structure, groups, outliers, etc) if plotted in 2 or 3 dimensions

Feature Selection vs Extraction

- **Feature selection**: Choosing \( k \) of \( d \) important features, ignoring the remaining \( d - k \)
  Subset selection algorithms
- **Feature extraction**: Project the original \( x_i, i = 1,...,d \) dimensions to new \( k \) dimensions, \( z_j, j = 1,...,k \)
  - Principal components analysis (PCA)
  - Linear discriminant analysis (LDA)
  - Factor analysis (FA)
  (also clustering-based approaches)
Feature Ranking  (see Guyon-Elisseeff)

- Find features with a high “score”:
  - Correlation with labels (regression)
  - Predictive power of attribute (1-attribute classifier)
  - Mutual information between labels and targets

- Relatively quick and simple

Subset Selection

- There are $2^d$ subsets of $d$ features
- Forward search: Add the best feature at each step
  - Set of features $F$ initially Ø.
  - At each iteration, find the best new feature
    - $f = \arg\min_i E(F \cup x_i)$
    - Add $x_f$ to $F$ if $E(F \cup x_f) < E(F)$
  - This is Hill-climbing $O(d^2)$ runs of algorithm ($O(dk)$ to pick $k$ features)
- Backward search: Start with all features and remove one at a time, if possible.
- Floating search (Add $k$, remove $l$)