CSCE 478/878 Lecture 2: Supervised Learning

Stephen Scott

(Adapted from Ethem Alpaydin)
Supervised learning is most fundamental, “classic” form of machine learning

“Supervised” part comes from the part of labels for examples (instances)
Outline

- Learning a class from labeled examples
  - Definition
  - Thinking about $C$
  - Hypotheses and error
  - Margin
- Noise and other problems
  - Noise
  - Model selection
  - Inductive bias
- Regression
- Multi-class problems
- General steps of machine learning
Learning a Class from Examples

Let $C$ be the target concept to be learned
- Think of $C$ as a function that takes as input an example (or instance) and outputs a label

Goal: Given a training set $\mathcal{X} = \{(x^t, r^t)\}_{t=1}^N$ where $r^t = C(x^t)$, output a hypothesis $h \in \mathcal{H}$ that approximates $C$ in its classifications of new instances

Each instance $x$ represented as a vector of attributes or features
- E.g., let each $x = (x_1, x_2)$ be a vector describing attributes of a car; $x_1 =$ price and $x_2 =$ engine power
- In this example, label is binary (positive/negative, yes/no, 1/0, $+1/-1$) indicating whether instance $x$ is a “family car”
Learning a Class from Examples (cont’d)

$x_2$: Engine power

$x_1$: Price

$x_1^t$: Price

$x_2^t$: Engine power
Can think of target concept $C$ as a *function*
- In example, $C$ is an axis-parallel box, equivalent to upper and lower bounds on each attribute
- Might decide to set $\mathcal{H}$ (set of candidate hypotheses) to the same family that $C$ comes from
- Not required to do so

Can also think of target concept $C$ as a *set* of positive instances
- In example, $C$ the continuous set of all positive points in the plane

Use whichever is convenient at the time
Thinking about $C$ (cont’d)

- $x_1$: Price
- $x_2$: Engine power
- $p_1$, $p_2$, $e_1$, $e_2$, $C$
Hypotheses and Error

- A learning algorithm uses training set $\mathcal{X}$ and finds a hypothesis $h \in \mathcal{H}$ that approximates $C$
- In example, $\mathcal{H}$ can be set of all axis-parallel boxes
- If $C$ guaranteed to come from $\mathcal{H}$, then we know that a perfect hypothesis exists
  - In this case, we choose $h$ from the version space = subset of $\mathcal{H}$ consistent with $\mathcal{X}$
  - What learning algorithm can you think of to learn $C$?
- Can think of two types of error (or loss) of $h$
  - Empirical error is fraction of $\mathcal{X}$ that $h$ gets wrong
  - Generalization error is probability that a new, randomly selected, instance is misclassified by $h$
    - Depends on the probability distribution over instances
  - Can further classify error as false positive and false negative
Hypotheses and Error (cont’d)

- Hypotheses and Error (cont’d)
- Margin
- Noise and Other Problems
- Regression
- Multi-Class Problems
- General Steps of Machine Learning

---

**Outline**

- Learning a Class from Examples
  - Definitions
  - Thinking about C
- Hypotheses and Error (cont’d)
- Margin
- Noise and Other Problems
- Regression
- Multi-Class Problems
- General Steps of Machine Learning
Since we will have many (infinitely?) choices of $h$, often will choose one with maximum margin (min distance to any point in $\mathcal{X}$).

Why?
In reality, it’s unlikely that there exists an $h \in \mathcal{H}$ that is perfect on $\mathcal{X}$
- Could be *noise* in the data (attribute errors, labeling errors)
- Could be attributes that are *hidden* or *latent*, which impact the label but are unobserved

Could find a better (or even perfect) fit to $\mathcal{X}$ if we choose a more powerful (expressive) hypothesis class $\mathcal{H}$

Is this a good idea?
For what reasons might we prefer $h_1$ over $h_2$?
Model Selection

- Might prefer simpler hypothesis because it is:
  - Easier/more efficient to evaluate
  - Easier to train (fewer parameters)
  - Easier to describe/justify prediction
  - Better fits *Occam’s Razor:* Tend to prefer simpler explanation among similar ones

- *Model selection* is the act of choosing a hypothesis class $\mathcal{H}$
  - Need to balance $\mathcal{H}$’s complexity with that of the model that labels the data:
    - If $\mathcal{H}$ not sophisticated enough, might *underfit* and not generalize well (e.g., fit line to data from cubic model)
    - If $\mathcal{H}$ too sophisticated, might *overfit* and not generalize well (e.g., fit the noise)
  - Can validate choice of $h$ (and $\mathcal{H}$) if some data held back from $\mathcal{X}$ to serve as *validation set*
    - Still part of training, but not directly used to select $h$
  - Independent *test set* often used to do final evaluation of chosen $h$
Inductive Bias

- Must assume something about the learning task.
- Otherwise, learning becomes rote memorization.
- Imagine allowing $\mathcal{H}$ to be set of arbitrary functions over set of all possible instances.
  - Every hypothesis in version space $\mathcal{V} \subseteq \mathcal{H}$ is consistent with all instances in $\mathcal{X}$.
  - For every other instance, exactly half the hypotheses in $\mathcal{V}$ will predict positive, the rest negative (see next slide).
  - No way to generalize on new, unseen instances without way to favor one hypothesis over another.

Inductive bias is a set of assumptions that we make to enable generalization over rote memorization.

- Manifests in choice of $\mathcal{H}$.
- Instead (or in addition), can have bias in preference of some hypotheses over others (e.g., based on specificity or simplicity).
Inductive Bias (cont’d)

- E.g., if \( \mathcal{X} = \{ (\langle 0, 0, 0 \rangle, +), (\langle 1, 1, 0 \rangle, +), (\langle 0, 1, 0 \rangle, -), (\langle 1, 0, 1 \rangle, -) \} \) then version space \( \mathcal{V} \) is the set of truth tables satisfying

\[
\begin{array}{cccc}
000 & + & 010 & - \\
001 & 011 & 100 & - \\
110 & + & 111 & - \\
\end{array}
\]

- Since there are 4 holes, \( |\mathcal{V}| = 2^4 = 16 = \text{number of ways to fill holes} \), and for any yet unclassified example \( \mathbf{x} \), \textit{exactly half} of hyps in \( \mathcal{V} \) classify \( \mathbf{x} \) as + and half as −.
Regression

- When labels $f(x)$ are real-valued rather than discrete, we call it \textit{regression}.
- Error of hypothesis $g$ measured by \textit{squared error} instead of number of misclassifications: $(f(x) - g(x))^2$.
  - Empirical error is now average squared error and generalization performance is expected squared error.
- Model selection now consists of choosing the complexity of hypothesis $g$, e.g., degree of polynomial:
  - Linear: $g(x) = w_1x + w_0$
  - Quadratic: $g(x) = w_2x^2 + w_1x + w_0$
  - And so on, where higher-order polynomials can better fit data based on more complex models, but are also more inclined to overfit.
- Learning consists of inferring parameters $w_i$. 
Polynomials of degree 1, 2, and 6
Some classification problems have discrete-valued labels, but not binary

E.g., instead of "family car" versus "not family car", have labels ("family car", "luxury sedan", "sports car")

How we handle this depends on the type of hypothesis/learning algorithm we use

- Some hypothesis classes (e.g., decision trees, \( k \) nearest neighbor) naturally have the ability to classify with non-binary labels
- Some are binary only (e.g., artificial neural networks, support vector machines, axis-parallel boxes)
  - In this case, can cast the multi-class problem as a collection of binary problems
  - In a \( K \)-class problem, can give each instance a \textit{vector} of \( K \) binary labels
Multi-Class Problems (cont’d)

- E.g., if original training set is

  \[ Y = \{(x^t, s^t)\}_{i=1}^N \]

  for each \( s^t \in \{C_1, \ldots, C_K\} \), then map it to

  \[ X = \{(x^t, r^t)\}_{i=1}^N \]

  where each \( r^t \) is a \( K \)-dimensional binary vector:

  \[ r^t_i = \begin{cases} 1 & \text{if } x^t \in C_i \\ 0 & \text{if } x^t \in C_j, j \neq i \end{cases} \]

- Can then train \( K \) separate binary classifiers in \textit{one-versus-rest} scheme

- (Other encodings of \( r \) also possible)
Multi-Class Problems (cont’d)

Three axis-parallel boxes as three binary classifiers, one per class
General Steps of Machine Learning

- Acquire training set \( \mathcal{X} = \{(x^t, r^t)\}_{t=1}^N \)
  - Assume *independent and identically distributed* (iid)
  - Assume probability distribution on \( \mathcal{X} \) is same as what we will see in practice
  - Labels \( r^t \) could be binary, multi-valued, real

- Choose hypothesis class \( \mathcal{H} \)

- Choose loss function \( L \)
  - 0-1 loss versus hinge loss versus squared loss ...

- Choose optimization procedure to find \( h \)
  - E.g., analytic solution for linear regression, backpropagation for artificial neural network, sequential minimal optimization for SVM

- Evaluate quality of \( h \) via estimation of generalization performance using *independent test set*