N-fold cross validation

- Instead of a single test-training split:
  - Split data into $N$ equal-sized parts
- Train and test $N$ different classifiers
- Report average accuracy and standard deviation of the accuracy
You have two different classifiers, A and B

You train and test them on the same data set using N-fold cross-validation

For the n-th fold:

\[ p_n = \text{accuracy}(A, n) - \text{accuracy}(B, n) \]

Is the difference between A and B’s accuracies significant?
Hypothesis testing

- You want to show that hypothesis $H$ is true, based on your data
  - (e.g. $H$ = “classifier A and B are different”)

- Define a null hypothesis $H_0$
  - ($H_0$ is the contrary of what you want to show)

- $H_0$ defines a distribution $P(m | H_0)$ over some statistic
  - e.g. a distribution over the difference in accuracy between A and B

- Can you refute (reject) $H_0$?
Rejecting $H_0$

- $H_0$ defines a distribution $P(M \mid H_0)$ over some statistic $M$
  - (e.g. $M =$ the difference in accuracy between $A$ and $B$)
- Select a significance value $S$
  - (e.g. 0.05, 0.01, etc.)
  - You can only reject $H_0$ if $P(m \mid H_0) \leq S$
- Compute the test statistic $m$ from your data
  - e.g. the average difference in accuracy over your $N$ folds
- Compute $P(m \mid H_0)$
- Refute $H_0$ with $p \leq S$ if $P(m \mid H_0) \leq S$
Paired t-test

- Null hypothesis ($H_0$; to be refuted):
  - There is no difference between A and B, i.e. the expected accuracies of A and B are the same
- That is, the expected difference (over all possible data sets) between their accuracies is 0:
  $$H_0: E[p_D] = 0$$
- We don’t know the true $E[p_D]$
- $N$-fold cross-validation gives us $N$ samples of $p_D$
Paired t-test

- Null hypothesis $H_0$: $E[\text{diff}_D] = \mu = 0$
- $m$: our estimate of $\mu$ based on $N$ samples of $\text{diff}_D$
  \[ m = \frac{1}{N} \sum_n \text{diff}_n \]
- The estimated variance $S^2$:
  \[ S^2 = \frac{1}{(N-1)} \sum_{1,N} (\text{diff}_n - m)^2 \]
- Accept Null hypothesis at significance level $\alpha$ if the following statistic lies in $(-t_{\alpha/2, N-1}, +t_{\alpha/2, N-1})$
  \[ \frac{\sqrt{N} m}{S} \sim t_{N-1} \]
Imbalanced data distributions

• Sometimes training examples are drawn from an imbalanced distribution
• This results in an imbalanced training set
  – “needle in a haystack” problems
  – E.g., find fraudulent transactions in credit card histories
• Why is this a big problem for the ML algorithms we know?
Learning with imbalanced data

• We need to let the learning algorithm know that we care about some examples more than others!

• 2 heuristics to balance the training data
  – Subsampling
  – Weighting
Recall: Machine Learning as Function Approximation

Problem setting
• Set of possible instances $X$
• Unknown target function $f : X \rightarrow Y$
• Set of function hypotheses $H = \{h \mid h : X \rightarrow Y\}$

Input
• Training examples $\{(x^{(1)}, y^{(1)}), ..., (x^{(N)}, y^{(N)})\}$ of unknown target function $f$

Output
• Hypothesis $h \in H$ that best approximates target function $f$
Recall: Loss Function

\[ l(y, f(x)) \] where \( y \) is the truth and \( f(x) \) is the system’s prediction

e.g. \( l(y, f(x)) = \begin{cases} 0 & \text{if } y = f(x) \\ 1 & \text{otherwise} \end{cases} \)

Captures our notion of what is important to learn
Recall: Expected loss

- $f$ should make good predictions
  - as measured by loss $l$
  - on future examples that are also drawn from $D$

- Formally
  - $\varepsilon$, the expected loss of $f$ over $D$ with respect to $l$ should be small

$$\varepsilon \triangleq \mathbb{E}_{(x,y) \sim D}\{l(y, f(x))\} = \sum_{(x,y)} D(x,y)l(y, f(x))$$
Task: Binary Classification

Given:

1. An input space $\mathcal{X}$

2. An unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, +1\}$

Compute: A function $f$ minimizing: $\mathbb{E}_{(x,y) \sim \mathcal{D}}[f(x) \neq y]$
We define cost of misprediction as:

\[ \alpha > 1 \text{ for } y = +1 \]

\[ 1 \text{ for } y = -1 \]

Given a good algorithm for solving the binary classification problem, how can I solve the \( \alpha \)-weighted binary classification problem?
Solution: Train a binary classifier on an induced distribution

---

Algorithm 11 \textsc{SubsampleMap}(D^{\text{weighted}}, \alpha)

1: \textbf{while true do}
2:     \((x, y) \sim D^{\text{weighted}}\) \hfill // draw an example from the weighted distribution
3:     \(u \sim \text{uniform random variable in } [0, 1]\)
4:     \textbf{if } y = +1 \textbf{ or } u < \frac{1}{\alpha} \textbf{ then}
5:         \textbf{return } (x, y)
6:     \textbf{end if}
7: \textbf{end while}
Subsampling optimality

• **Theorem:** If the binary classifier achieves a binary error rate of $\varepsilon$, then the error rate of the $\alpha$-weighted classifier is $\alpha \varepsilon$

• Proof (CIML 6.1)
Strategies for inducing a new binary distribution

• Undersample the negative class

• Oversample the positive class
Strategies for inducing a new binary distribution

• Undersample the negative class
  – More computationally efficient

• Oversample the positive class
  – Base binary classifier might do better with more training examples
  – Efficient implementations incorporate weight in algorithm, instead of explicitly duplicating data!
Algorithm 1 \textbf{DecisionTreeTrain}(data, remaining features)

1. \texttt{guess} $\leftarrow$ most frequent answer in \texttt{data} \hfill // default answer for this data
2. \textbf{if} the labels in \texttt{data} are unambiguous \textbf{then}
3. \textbf{return} \texttt{Leaf}(\texttt{guess}) \hfill // base case: no need to split further
4. \textbf{else if} remaining features is empty \textbf{then}
5. \textbf{return} \texttt{Leaf}(\texttt{guess}) \hfill // base case: cannot split further
6. \textbf{else}
7. \textbf{for all} $f \in$ remaining features \textbf{do}
8. \hspace{2em} \texttt{NO} $\leftarrow$ the subset of \texttt{data} on which $f$=\texttt{no}
9. \hspace{2em} \texttt{YES} $\leftarrow$ the subset of \texttt{data} on which $f$=\texttt{yes}
10. \hspace{2em} \texttt{score}[$f$] $\leftarrow$ \# of majority vote answers in \texttt{NO} \hfill // the accuracy we would get if we only queried on $f$
11. \hspace{5em} + \# of majority vote answers in \texttt{YES}
12. \textbf{end for}
13. $f$ $\leftarrow$ the feature with maximal \texttt{score}(\texttt{f})
14. \texttt{NO} $\leftarrow$ the subset of \texttt{data} on which $f$=\texttt{no}
15. \texttt{YES} $\leftarrow$ the subset of \texttt{data} on which $f$=\texttt{yes}
16. \texttt{left} $\leftarrow$ \textbf{DecisionTreeTrain}(\texttt{NO}, remaining features \setminus \{f\})
17. \texttt{right} $\leftarrow$ \textbf{DecisionTreeTrain}(\texttt{YES}, remaining features \setminus \{f\})
18. \textbf{return} \texttt{Node}(f, left, right)
19. \textbf{end if}
Given an arbitrary method for binary classification, how can we learn to make multiclass predictions?

Fundamental ML concept: reductions
Multiclass classification

• Real world problems often have multiple classes (text, speech, image, biological sequences...)

• How can we perform multiclass classification?
  – Straightforward with decision trees or KNN
  – Can we use the perceptron algorithm?
Reductions

• Idea is to re-use simple and efficient algorithms for binary classification to perform more complex tasks

• Works great in practice:
  – E.g., Vowpal Wabbit
One Example of Reduction: Learning with Imbalanced Data

**Task: \( \alpha \)-Weighted Binary Classification**

*Given:*

1. An input space \( \mathcal{X} \)
2. An unknown distribution \( \mathcal{D} \) over \( \mathcal{X} \times \{ -1, +1 \} \)

*Compute:* A function \( f \) minimizing: 
\[
\mathbb{E}_{(x,y) \sim \mathcal{D}} [\alpha^y \mathbb{1}[f(x) \neq y]]
\]

**Subsampling Optimality Theorem:**
If the binary classifier achieves a binary error rate of \( \varepsilon \), then the error rate of the \( \alpha \)-weighted classifier is \( \alpha \varepsilon \)
What you should know

• Be aware of practical issues when applying ML techniques to new problems

• How to select an appropriate evaluation metric for imbalanced learning problems

• How to learn from imbalanced data using $\alpha$-weighted binary classification, and what the error guarantees are