Dr. Bei Wang

School of Computing
Scientific Computing and Imaging Institute (SCI)
University of Utah
www.sci.utah.edu/~beiwang
beiwang@sci.utah.edu

Feb 21, 2019
Announcement Project 2

- Project 2 will be posted in 2 days with due date changed to 3/21 (2-day extension from the original due date).
Machine Learning
An Intuitive Introduction
Predicting things we have not seen by using what we have seen.

- Example: how photo app predicts who is in the photo
- Predict unseen data (test data) using seen data (training data)
- Two types of “prediction”:
  - Classification: apply label to data.
  - Example: based on my previous reviews on restaurants, decides whether I will like or dislike a new restaurant.
  - Regression: assign a value to data.
  - Example: predict the score (from 1 to 100) for a new restaurant.
  - Classification – getting the label right.
  - Regression – predicting a value that is not far from the real value.

Shah and Pahwa (2019)
Error in Machine Learning

- Error is used to evaluate performance.
- Error is where “learning” happens.
- Error is used to train the ML algorithms.
- **Train** an algorithm: use the training data to learn some prediction scheme that can then be used on the (unseen) test data.
  
  Shah and Pahwa (2019)
Training data: points with labels, \( \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\} \).

Test data: (unlabeled) points, \( \{z_1, z_2, \cdots, z_m\} \).

Classification: use training data to label test data.

A classifier (a ML algorithm) is a function \( f \) that takes some input \( z_i \) and maps it to a label \( f(z_i) \).

**Training error**

\[
\epsilon_{\text{train}} = \frac{1}{n} \sum_{i=1}^{n} [[f(x_i) \neq y_i]]
\]

\([ [ S ]] = 1 \) if the statement \( S \) is true; \([ [ S ]] = 0 \) otherwise.

**Test error**

\[
\epsilon_{\text{test}} = \frac{1}{m} \sum_{i=1}^{m} [[f(z_i) \neq \text{real label of } z_i]]
\]

Shah and Pahwa (2019)
Other Errors

- **Regression error**: varies, i.e., mean squared error.
- Errors capture performance.
- Algorithm only knows the training data; optimize for training data does not necessarily optimize for error or test data.

- **Loss functions**: An ML algorithm defines error as loss.
- Different algorithms optimize for different loss functions.

- **Underfitting**: an algorithm does not use enough information of the training data.

- **Overfitting**: an algorithm overadapts to the training data.

Shah and Pahwa (2019)
Classification

Identify to which of a set of categories a new observation belongs:

- Linear Classifiers
- Perceptron Algorithm
- Kernels
- SVM
- Neural Networks

A classifier that uses a line (hyperplane) to label the data points.

1. Define a line by its normal vector (direction) $u$ and a point on the line (e.g., offset).
2. Points on the same side of $u$ have $+$ labels; $-$ labels otherwise.
What if the data is not linearly separable?
- Solution: transform the data (via a function) into some space that is linearly separable, e.g., via kernel methods.

What if we want more than two labels?
- Solution: use multiple linear classifiers: $ABCD \rightarrow AB, CD \rightarrow A, B, C, D$

Shah and Pahwa (2019)
Idea: if you make a mistake, adjust the line based on the mistake.

Push the normal vector in the direction of the mistake if it was positively labeled, and away if it was negatively labeled.

Pay attention to the notion of inner product.

Additional reading: Shah and Pahwa (2019); Phillips (2019)
For two vectors $p = (p_1, \ldots, p_d)^T, q = (q_1, \ldots, q_d)^T \in \mathbb{R}^d$, the inner product:

$$\langle p, q \rangle = p^T q = \sum_{i=1}^{d} p_i \cdot q_i$$

Also: $p^T q = ||p|| ||q|| \cos \theta$, where $\theta$ is the angle between the two vectors.
Learn a linear binary classifier $\mathbf{w}$

$\mathbf{w}$ is a vector of weights (together with an intercept term $b$, omitted here) that is used to classify a sample vector $\mathbf{x}$ as class $+1$ or class $-1$ according to

$$\hat{y} = \text{sgn}(\mathbf{w}^T \mathbf{x})$$
Perceptron pseudocode

1. Initialize \( \mathbf{w} \) to an all-zero vector.

2. For a fixed number of iterations, or until stopping criterion is met:
   - For each training example \( x_i \) with ground truth label \( y_i \in \{-1, +1\} \):
     - Let \( \hat{y} = \text{sgn}(\mathbf{w}^T x_i) \).
     - If \( \hat{y} \neq y_i \), update \( \mathbf{w} \leftarrow \mathbf{w} + y_i x_i \).

https://en.wikipedia.org/wiki/Kernel_perceptron
Perceptron in action

https://www.youtube.com/watch?v=vGwemZhPlsA
From perceptron to kernel perceptron: dual perceptron

- The weight vector $w$ can be expressed as a linear combination of the $n$ training points:

$$w = \sum_{i}^{n} \alpha_i y_i x_i$$

- $\alpha_i$: number of times $x_i$ was misclassified, forcing an update to $w$

- A dual perceptron algorithm loops through the samples as before, making predictions, but instead of storing and updating a weight vector $w$, it updates a "mistake counter" vector $\alpha$:

$$\hat{y} = \text{sgn}(w^T x)$$

$$= \text{sgn} \left( \sum_{i}^{n} \alpha_i y_i x_i \right)^T x$$

$$= \text{sgn} \left( \sum_{i}^{n} \alpha_i y_i (x_i \cdot x) \right)$$

(1)

https://en.wikipedia.org/wiki/Kernel_perceptron
Kernel perceptron: replace *dot product* in the dual perceptron by an arbitrary *kernel* function, to get the effect of a feature map $\Phi$ without computing $\Phi(x)$ explicitly for any samples.

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle.$$  

https://en.wikipedia.org/wiki/Kernel_method
Kernel perceptron

- Perceptron: learns a linear classifier
- Kernel perceptron: learns a kernelized classifier
- A *kernel* is a (user-specified) similarity function over pairs of data points.
- A *kernel machine* is a classifier that stores a subset of its training examples $x_i$, associates with each a weight $\alpha_i$, and makes decisions for new samples $x$ by evaluating

$$\hat{y} = \text{sgn} \left( \sum_i \alpha_i y_i K(x_i, x) \right)$$

https://en.wikipedia.org/wiki/Kernel_perceptron
Pseudocode:

1. Initialize $\alpha$ to an all-zeros vector of length $n$
2. For a fixed number of iterations, or until stopping criterion is met:
   - For each training example $x_j$ with ground truth label $y_j \in \{-1, +1\}$:
     - Let $\hat{y} = \text{sgn} \left( \sum_{i=1}^{n} \alpha_i y_i K(x_i, x_j) \right)$.
     - If $\hat{y} \neq y_i$, update $\alpha_j \leftarrow \alpha_j + 1$.

https://en.wikipedia.org/wiki/Kernel_perceptron
Commonly used kernels

- Gaussian kernel: \( K(p, q) = \exp\left(-||p - q||^2/\sigma^2\right) \) with bandwidth \( \sigma \)
- Laplace kernel: \( K(p, q) = \exp\left(-||p - q||/\sigma\right) \) with bandwidth \( \sigma \)
- Polynomial kernel of power \( r \): \( K(p, q) = (\langle p, q \rangle + c)^r \) with control parameter \( c > 0 \).
Kernel perceptron in action

https://www.youtube.com/watch?v=mC0awsGOOcs
Coming up next: kernels for barcodes
http://www.cs.utah.edu/~jeffp/M4D/M4D.html.