Supervised Learning II

George Konidaris
gdk@cs.brown.edu

Fall 2021
Machine Learning

Subfield of AI concerned with learning from data.

Broadly, using:
- **Experience**
- To Improve **Performance**
- On Some **Task**

*(Tom Mitchell, 1997)*
Supervised Learning

Input:
\[ X = \{x_1, \ldots, x_n\} \quad \text{inputs} \]
\[ Y = \{y_1, \ldots, y_n\} \quad \text{labels} \]

Learn to predict new labels.
Given \( x \): \( y \)?
Supervised Learning

Formal definition:

Given training data:
\[ X = \{x_1, \ldots, x_n\} \text{ inputs} \]
\[ Y = \{y_1, \ldots, y_n\} \text{ labels} \]

Produce:
Decision function \( f : X \rightarrow Y \)

That minimizes error:
\[ \sum_i \text{err}(f(x_i), y_i) \]
Nonparametric Methods

Most ML methods are parametric:
- Characterized by setting a few parameters.
- \( y = f(x, w) \)

Alternative approach:
- Let the data speak for itself.
- No finite-sized parameter vector.
- Usually more interesting decision boundaries.
K-Nearest Neighbors

Given training data:
\[ X = \{x_1, \ldots, x_n\} \]
\[ Y = \{y_1, \ldots, y_n\} \]
Distance metric \( D(x_i, x_j) \)

For a new data point \( x_{\text{new}} \):
- find \( k \) nearest points in \( X \) (measured via \( D \))
- set \( y_{\text{new}} \) to the majority label
K-Nearest Neighbors
K-Nearest Neighbors

Decision boundary … what if k=1?
K-Nearest Neighbors

Properties:

• No learning phase.
• Must store all the data.
• \( \log(n) \) computation per sample - grows with data.

Decision boundary:

• \textit{any function, given enough data.}

\textbf{Classic trade-off:} memory and compute time for flexibility.
Classification vs. Regression

If the set of labels $Y$ is discrete:
- Classification
- Minimize number of errors

If $Y$ is real-valued:
- Regression
- Minimize sum squared error

Let’s look at regression.
Regression with Decision Trees

Start with decision trees with real-valued inputs.

```plaintext
a > 3.1

true
y=1

false

b < 0.6?

true
y=2

false
y=1
```
Regression with Decision Trees

... now real-valued outputs.
Regression with Decision Trees

Training procedure - fix a depth, $k$.

If you have $k=1$, fit the average.

If $k > 1$:
   Consider all variables to split on
   Find the one that minimizes SSE
   Recurse ($k-1$)

Choice of $k$ prevents overfitting.
Regression with Decision Trees

Decision Tree Regression

- blue: max_depth=2
- green: max_depth=5
- data

(via scikit-learn docs)
Linear Regression

Alternatively, explicit equation for prediction.

Recall the Perceptron.

If \( x = [x(1), \ldots, x(n)] \):

- Create an \( n \)-d line
- Slope for each \( x(i) \)
- Constant offset

\[
f(x) = \text{sign}(w \cdot x + c)
\]
Linear Regression

Directly represent $f$ as a linear function:

- $f(x, w) = w \cdot x + c$

What can be represented this way?
Linear Regression

How to train?

Given inputs:
  - $x = [x_1, \ldots, x_n]$ (each $x_i$ is a vector, first element = 1)
  - $y = [y_1, \ldots, y_n]$ (each $y_i$ is a real number)

**Define error function:**
Minimize summed squared error

$$\sum_{i=1}^{n}(w \cdot x_i - y_i)^2$$
Linear Regression

The usual story:

• Set derivative of error function to zero.

\[
\frac{d}{dw} \sum_{i=1}^{n} (w \cdot x_i - y_i)^2 = 0
\]

\[
2 \sum_{i=1}^{n} (w \cdot x_i - y_i)x_i^T = 0
\]

\[
\left( \sum_{i=1}^{n} x_i^T x_i \right) w = \sum_{i=1}^{n} x_i^T y_i
\]

\[
w = A^{-1} b
\]

\[
A = \left( \sum_{i=1}^{n} x_i^T x_i \right) \quad \text{matrix}
\]

\[
b = \sum_{i=1}^{n} x_i^T y_i \quad \text{vector}
\]
Polynomial Regression

More powerful:

- Polynomials in state variables.
  - 1st order: $[1, x, y, xy]$  
  - 2nd order: $[1, x, y, xy, x^2, y^2, x^2y, y^2x, x^2y^2]$ 

$y_i = w \cdot \Phi(x_i)$

What can be represented?
Polynomial Regression

As before …

\[
\frac{d}{dw} \sum_{i=1}^{n} (w \cdot \Phi(x_i) - y_i)^2
\]

\[
w = A^{-1} b
\]

\[
A = \sum_{i=1}^{n} \Phi^T(x_i) \Phi(x_i)
\]

\[
b = \sum_{i=1}^{n} \Phi^T(x_i) y_i
\]
Polynomial Regression
Overfitting
Overfitting
Ridge Regression

A characteristic of overfitting:
  • Very large weights.

Modify the objective function to discourage this:

\[
\min \sum_{i=1}^{n} (w \cdot x_i - y_i)^2 + \lambda \|w\|
\]

error term  

regularization term

\[
w = (A^T A + \Lambda^T \Lambda)^{-1} A^T b
\]
Neural Network Regression

\[ \sigma(w \cdot x + c) \]

classification
Neural Network Regression

input layer

output layer

hidden layer
Neural Network Regression

\[ w_1^{o1} h_1 + w_2^{o1} h_2 + w_3^{o1} h_3 + w_4^{o1} \]

value computed

\[ w_1^{o2} h_1 + w_2^{o2} h_2 + w_3^{o2} h_3 + w_4^{o2} \]

value computed

\[ h_1 = \sigma(w_1^{h1} x_1 + w_2^{h1} x_2 + w_3^{h1}) \]

input layer

\[ x_1, x_2 \in [0, 1] \]
Neural Network Regression

A neural network is just a parametrized function: \( y = f(x, w) \)

How to train it?

Write down an error function:

\[
(y_i - f(x_i, w))^2
\]

Minimize it! (w.r.t. \( w \))

No closed form solution to gradient = 0. Hence, stochastic gradient descent:

- Compute \( \frac{d}{dw} (y_i - f(x_i, w))^2 \)
- Descend
Image Colorization

(Zhang, Isola, Efros, 2016)
Nonparametric Regression

Most ML methods are parametric:

• Characterized by setting a few parameters.
  \[ y = f(x, w) \]

Alternative approach:

• Let the data speak for itself.
• No finite-sized parameter vector.
• Usually more interesting decision boundaries.
Nonparametric Regression

What’s the regression equivalent of $k$-means?

**Given** training data:

$X = \{x_1, \ldots, x_n\}$

$Y = \{y_1, \ldots, y_n\}$

Distance metric $D(x_i, x_j)$

For a new data point $x_{\text{new}}$:

find $k$ nearest points in $X$ (measured via $D$)

set $y_{\text{new}}$ to the (weighted by $D$) average $y_i$ labels
Nonparametric Regression

As \( k \) increases, \( f \) gets smoother.
Gaussian Processes
Applications

model and predict variations in pH, clay, and sand content in the topsoil

(Gonzalez et al., 2007)