# EN.600.475 Machine Learning

## Regression

Raman Arora
Lecture 3
February 6, 2017

- Supervised Learning
- Linear regression

Slides credit: Greg Shakhnarovich

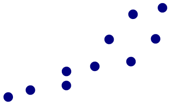
# Formal setup

- Input data space $\mathcal{X}$
- Output (label, target) space $\mathcal{Y}$
- Unknown function $f : \mathcal{X} \to \mathcal{Y}$
- We are given a set of labeled examples $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, N$, with $\mathbf{x}_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$.
- Goal: any for future $\mathbf{x}$, accurately predict $y$
  in other words: learn a mapping $f : \mathcal{X} \to \mathcal{Y}$

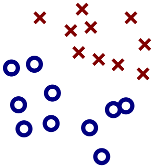
# Types of supervised problems

- Goal: learn $f : \mathcal{X} \to \mathcal{Y}$
- We will consider two sorts of $f$, based on nature of $\mathcal{Y}$:

regression: $\mathcal{Y} = \mathbb{R}$



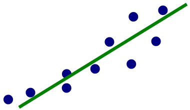
classification: $\mathcal{Y} = \{1, \ldots, C\}$



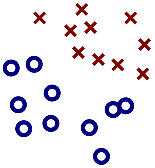
# Types of supervised problems

- Goal: learn $f : \mathcal{X} \to \mathcal{Y}$
- We will consider two sorts of $f$, based on nature of $\mathcal{Y}$:

regression: $\mathcal{Y} = \mathbb{R}$
learn a (continuous) function $f$



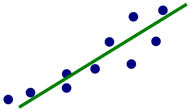
classification: $\mathcal{Y} = \{1, \ldots, C\}$



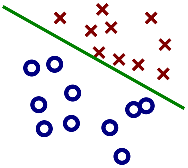
# Types of supervised problems

- Goal: learn $f : \mathcal{X} \to \mathcal{Y}$
- We will consider two sorts of $f$, based on nature of $\mathcal{Y}$:

regression: $\mathcal{Y} = \mathbb{R}$
learn a (continuous) function $f$



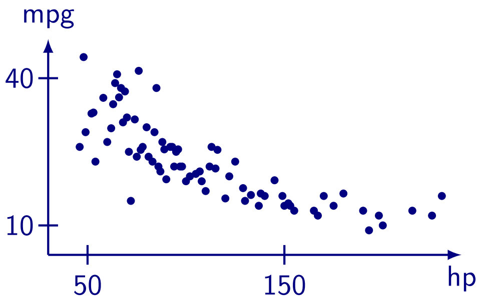
classification: $\mathcal{Y} = \{1, \ldots, C\}$
learn a separator between classes



# Regression

- We are given a set of $N$ observations $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, N$, with $y_i \in \mathbb{R}$.

- Example: predict car MPG $y$ from engine horsepower $x$



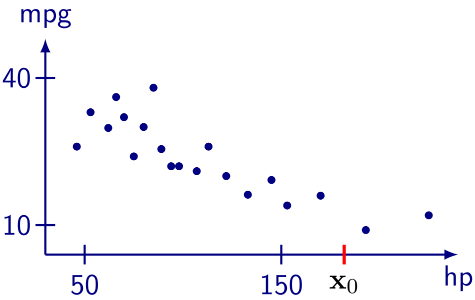

- Does it make sense to use learning here?

# Attempt 1: rote learning

- Memorize the observed $(\mathbf{x}, y)$ pairs

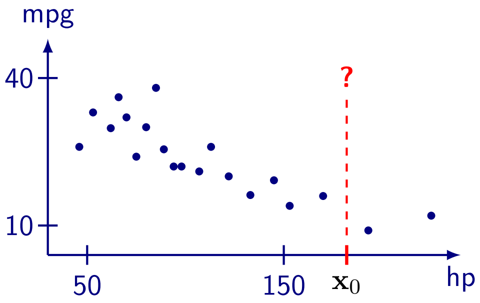
# Attempt 1: rote learning

- Memorize the observed $(\mathbf{x}, y)$ pairs
- What do we do when a new $\mathbf{x}$ comes along?



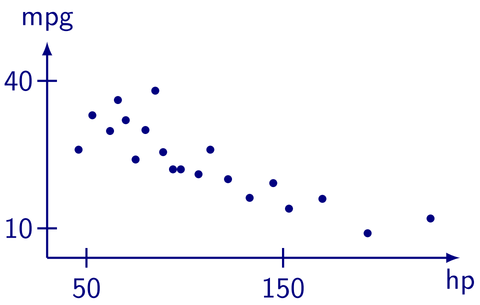
# Attempt 1: rote learning

- Memorize the observed $(\mathbf{x}, y)$ pairs
- What do we do when a new $\mathbf{x}$ comes along?



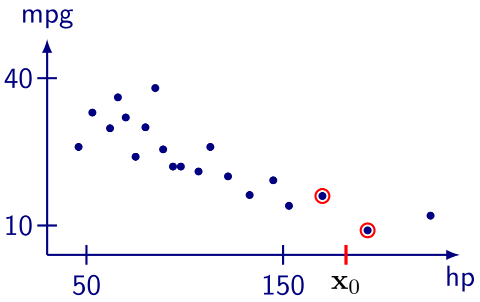

- Is this really learning?

# Attempt 2: lazy learning



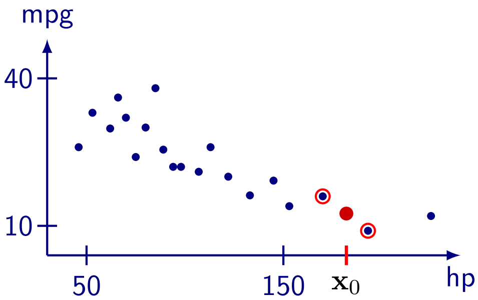

- Memorize the observed values
- For a new $\mathbf{x}_0$, find two nearest neighbors, that is, two observed $\mathbf{x}_i$ closest to it, and predict $\widehat{y}(\mathbf{x}_0)$ as the average of the nearest neighbors' $y_i$s

# Attempt 2: lazy learning



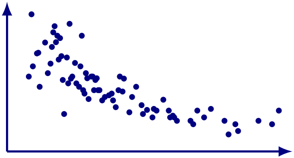

- Memorize the observed values
- For a new $\mathbf{x}_0$, find two nearest neighbors, that is, two observed $\mathbf{x}_i$ closest to it, and predict $\widehat{y}(\mathbf{x}_0)$ as the average of the nearest neighbors' $y_i$s

# Attempt 2: lazy learning



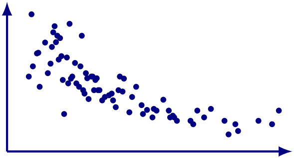

- Memorize the observed values
- For a new $\mathbf{x}_0$, find two nearest neighbors, that is, two observed $\mathbf{x}_i$ closest to it, and predict $\widehat{y}(\mathbf{x}_0)$ as the average of the nearest neighbors' $y_i$s
- This is $k$-nearest neighbors regression $(k = 2)$

# Attempt 3: fit a model



- Two goals in mind:
- Explain the data (traditional statistics)
- Make predictions (emphasized in machine learning)
- We will proceed in two steps:
  1. Choose a *model class* of functions

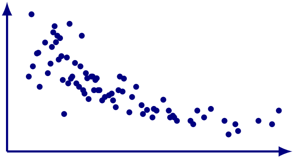
# Attempt 3: fit a model



- Two goals in mind:
- Explain the data (traditional statistics)
- Make predictions (emphasized in machine learning)
- We will proceed in two steps:
  1. Choose a *model class* of functions
  2. Design a fitting criterion, to guide selection of a function from the class.

# Attempt 3: fit a model

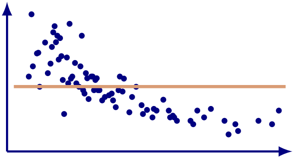
- Two goals in mind:



- Explain the data (traditional statistics)
- Make predictions (emphasized in machine learning)
- We will proceed in two steps:
  1. Choose a *model class* of functions
  2. Design a fitting criterion, to guide selection of a function from the class.
- Simplest model class: constant functions

# Attempt 3: fit a model

- Two goals in mind:

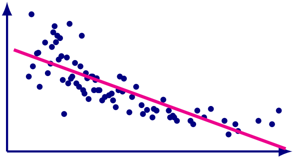


- Explain the data (traditional statistics)
- Make predictions (emphasized in machine learning)

- We will proceed in two steps:
  1. Choose a *model class* of functions
  2. Design a fitting criterion, to guide selection of a function from the class.
- Simplest model class: constant functions

# Attempt 3: fit a model

- Two goals in mind:



- Explain the data (traditional statistics)
- Make predictions (emphasized in machine learning)

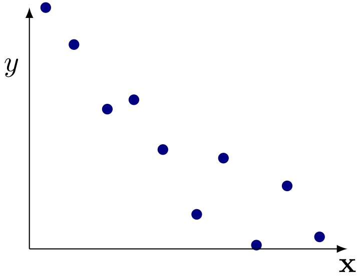
- We will proceed in two steps:
  1. Choose a *model class* of functions
  2. Design a fitting criterion, to guide selection of a function from the class.
- Simplest model class: constant functions
- Second simplest: linear functions

# Fitting a line to data

- We want to fit a linear function to an observed set of points $X = [x_1, \ldots, x_N]$ with associated labels $Y = [y_1, \ldots, y_N]$.
  - Once we fit the function, we want to use it to *predict* the $y$ for new $x$.

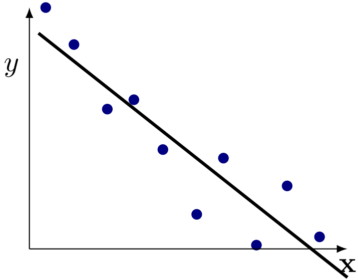
# Fitting a line to data

- We want to fit a linear function to an observed set of points $X = [x_1, \ldots, x_N]$ with associated labels $Y = [y_1, \ldots, y_N]$.
  - Once we fit the function, we want to use it to *predict* the $y$ for new $x$.



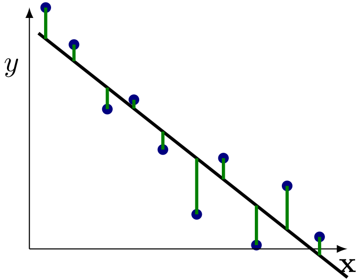
# Fitting a line to data

- We want to fit a linear function to an observed set of points $X = [x_1, \ldots, x_N]$ with associated labels $Y = [y_1, \ldots, y_N]$.
  - Once we fit the function, we want to use it to *predict* the $y$ for new $x$.
- Least squares (LSQ) fitting criterion: find the function that minimizes sum (or average) of square distances between actual $y$s in the training set and predicted ones.



# Fitting a line to data

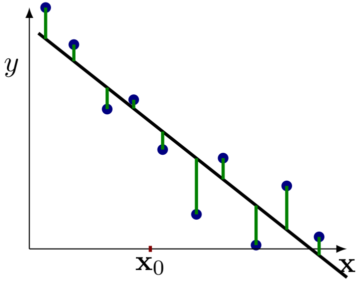
- We want to fit a linear function to an observed set of points $X = [x_1, \ldots, x_N]$ with associated labels $Y = [y_1, \ldots, y_N]$.
  - Once we fit the function, we want to use it to *predict* the $y$ for new $x$.
- Least squares (LSQ) fitting criterion: find the function that minimizes sum (or average) of square distances between actual $y$s in the training set and predicted ones.




distances along $y$, not orthogonal to line!

# Fitting a line to data

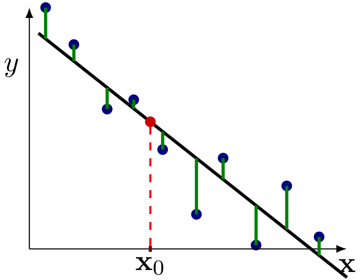
- We want to fit a linear function to an observed set of points $X = [x_1, \ldots, x_N]$ with associated labels $Y = [y_1, \ldots, y_N]$.
  - Once we fit the function, we want to use it to *predict* the $y$ for new $x$.
- Least squares (LSQ) fitting criterion: find the function that minimizes sum (or average) of square distances between actual $y$s in the training set and predicted ones.




distances along $y$, not orthogonal to line!

# Fitting a line to data

- We want to fit a linear function to an observed set of points $X = [x_1, \ldots, x_N]$ with associated labels $Y = [y_1, \ldots, y_N]$.
  - Once we fit the function, we want to use it to *predict* the $y$ for new $x$.
- Least squares (LSQ) fitting criterion: find the function that minimizes sum (or average) of square distances between actual $y$s in the training set and predicted ones.

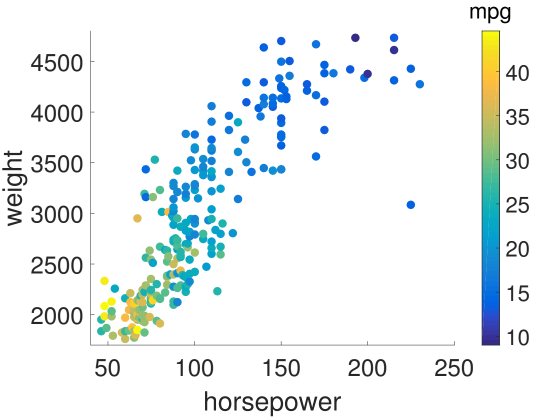



distances along $y$, not orthogonal to line! The fitted line is used as a predictor; it

*summarizes* the information $x$ provides about $y$, according to the model

# Multiple input variables

- Can consider additional features; e.g., $x_1$ horsepower and $x_2$ vehicle weight.
- We now have mapping from $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$ to $y$






colorbar: one possible way to convey multi-dimensional plots