Machine Learning Approach

Hung Le

University of Victoria

January 23, 2019

ML approach

- Most data mining algorithms try to summarize the data to help decision making.
- "Machine learning" algorithms not only summarize the data, but also provide a model to reason about future data.
  - Unsupervised learning: building a model from data without "label".
  - Supervised learning: building a model from data with labels.

Supervised Learning

Data is given as a set of pairs $\{\mathbf{x}, y\}$ where:

- $\mathbf{x}$ is a vector of *features*. Each feature could be *categorical* (such as $\{$red, green, blue$\}$) or *numerical*.
- y is the label. The value of y could be anything.

Supervised Learning

Data is given as a set of pairs $\{\mathbf{x}, y\}$ where:

- $\mathbf{x}$ is a vector of *features*. Each feature could be *categorical* (such as {red, green, blue}) or *numerical*.
- y is the label. The value of y could be anything.
  - If y is a real number $\Rightarrow$ regression problem.
  - If y is a discrete value $\Rightarrow$ classification problem.

Supervised Learning

Data is given as a set of pairs $\{\mathbf{x}, y\}$ where:

- $\mathbf{x}$ is a vector of *features*. Each feature could be *categorical* (such as {red, green, blue}) or *numerical*.
- y is the label. The value of y could be anything.
  - If y is a real number $\Rightarrow$ regression problem.
  - If y is a discrete value $\Rightarrow$ classification problem.

We typically split the data into two sets: a *training set* and a *test set*.

- The training set is used to train the model, i.e, find parameters of the model.
- The test set is used to test the performance of the trained model.

Why do we need to do so?

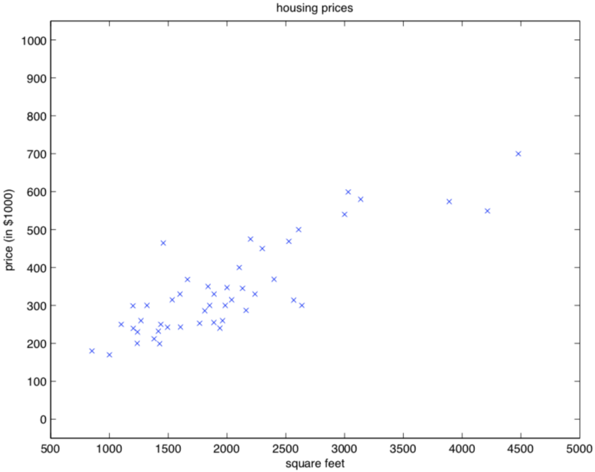
Let's start with (linear) regression.

Linear Regression - A Motivating Example[1]

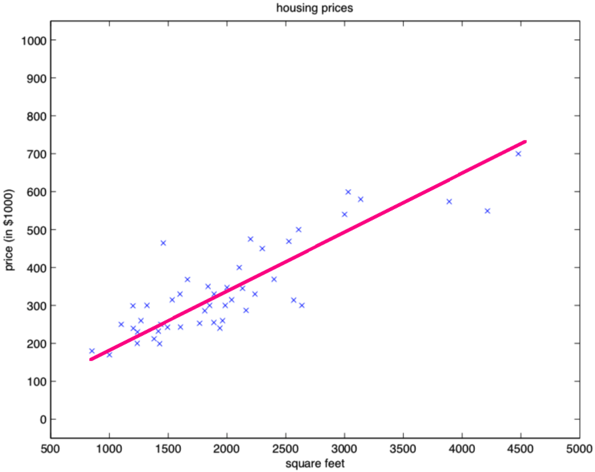
Living area ($feet^2$)	Price (1000$)
1204	400
1600	330
2400	369
1416	232
3000	540

[1] From Andrew Ng note `http://cs229.stanford.edu/notes/cs229-notes1.pdf`

Linear Regression - A Motivating Example

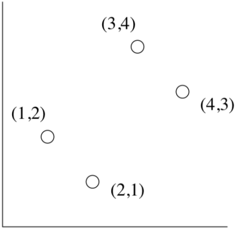


Linear Regression - A Motivating Example



Linear Regression - A Toy Example

Given four points $(1,2),(2,1),(3,4),(4,3)$, find a line $y = a \cdot x + b$ that best fits these points.



Here best fit means the sum of squares of vertical off-sets is minimum:

$$f(a,b) = (a+b-2)^2 + (2a+b-1)^2 + (3a+b-4)^2 + (4a+b-3)^2 \quad (1)$$

Optimization by solving equations

Solving Equation Approach

A (local) minimizer $\mathbf{w}_0$ of a differentiable function $f(.)$ satisfies:

$$\nabla f(\mathbf{w}_0) = 0 \tag{2}$$

Recall that given a differentiable function:

$$\begin{aligned} f : \mathbb{R}^d &\rightarrow \mathbb{R} \\ \mathbf{w} &\mapsto f(\mathbf{w}) \end{aligned} \tag{3}$$

Then:

$$\nabla f = \begin{bmatrix} \partial f / \partial w_1 \\ \partial f / \partial w_2 \\ \dots \\ \partial f / \partial w_d \end{bmatrix} \tag{4}$$

Back to our toy example

Find (a, b) that minimizes:

$$f(a,b) = (a+b-2)^2 + (2a+b-1)^2 + (3a+b-4)^2 + (4a+b-3)^2$$

Back to our toy example

Find (a, b) that minimizes:

$$f(a,b) = (a+b-2)^2 + (2a+b-1)^2 + (3a+b-4)^2 + (4a+b-3)^2$$

$$\frac{\partial f(a,b)}{\partial a} = 60a + 20b - 56$$
$$\frac{\partial f(a,b)}{\partial b} = 20a + 8b - 20$$

Solving $\nabla f(.) = 0$, we get $a = 3/5, b = 1$.

Multivariate Linear Regression

You are given a set of n data points $\mathcal{D} = \{(\mathbf{x}_1, y_1), \mathbf{x}_2, y_y), \ldots, (\mathbf{x}_n, y_n)\}$ where each $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. Find a hyperplane $y = \mathbf{w}^t\mathbf{x} + w_0$ such that:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \mathbf{w}^T\mathbf{x}_i - w_0)^2$$

is minimized.

Before we go into details of solving equations, we will "clean it up".

Multivariate Linear Regression

1st trick:

1. add an extra dimension $d+1$ and add 1 to each $\mathbf{x}_i$ in the new dimension so that $\mathbf{x}_i$ becomes $\begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}$.
2. also $\mathbf{w}$ becomes $\begin{bmatrix} \mathbf{w} \\ w_0 \end{bmatrix}$

That implies: $y = \mathbf{w}^T \mathbf{x} + w_0$ is equivalent to $y = \mathbf{w}^T \mathbf{x}$ in the new space.

$$J(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i) \tag{5}$$

in the new space.

Multivariate Linear Regression

2nd trick: write $J(\mathbf{w})$ in matrix-vector notation:

$$\mathbf{X} = \begin{bmatrix} \text{———} \mathbf{x}_1^T \text{———} \\ \text{———} \mathbf{x}_2^T \text{———} \\ \dots \\ \text{———} \mathbf{x}_n^T \text{———} \end{bmatrix} \tag{6}$$

$$J(\mathbf{w}) = \frac{1}{2}(\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y}) \tag{7}$$

Multivariate Linear Regression

3nd trick: traces and matrix derivatives.

- If $A = [A_{ij}]_{n \times n}$, then $\mathrm{Tr}(A = \sum_i A_{ii}$.

$$\begin{aligned} \mathrm{Tr}(a) &= a \quad a \in \mathbb{R} \\ \mathrm{Tr}(aB) &= a\mathrm{Tr}(B) \quad a \in \mathbb{R} \\ \mathrm{Tr}(A+B) &= \mathrm{Tr}(A) + \mathrm{Tr}(B) \\ \mathrm{Tr}(AB) &= \mathrm{Tr}(BA) \\ \mathrm{Tr}(ABC) &= \mathrm{Tr}(CAB) \end{aligned} \tag{8}$$

Multivariate Linear Regression

3nd trick: traces and matrix derivatives.

- If $A = [A_{ij}]_{n\times n}$, then $\mathrm{Tr}(A = \sum_i A_{ii}$.

$$\begin{aligned} \mathrm{Tr}(a) &= a \quad a \in \mathbb{R} \\ \mathrm{Tr}(aB) &= a\mathrm{Tr}(B) \quad a \in \mathbb{R} \\ \mathrm{Tr}(A+B) &= \mathrm{Tr}(A) + \mathrm{Tr}(B) \\ \mathrm{Tr}(AB) &= \mathrm{Tr}(BA) \\ \mathrm{Tr}(ABC) &= \mathrm{Tr}(CAB) \end{aligned} \tag{8}$$

-

$$\begin{aligned} \nabla_A \mathrm{Tr}(AB) &= B^T \\ \nabla_A \mathrm{Tr}(ABA^T C) &= CAB + C^T AB^T \\ \nabla_{A^T} f(A) &= (\nabla_A f(A))^T \end{aligned} \tag{9}$$

Multivariate Linear Regression

Apply the 3nd trick to $J(\mathbf{w})$:

$$\begin{aligned} J(\mathbf{w}) &= \frac{1}{2}(\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y}) \\ \Rightarrow \nabla_{\mathbf{w}} J(\mathbf{w}) &= \mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y} \end{aligned} \tag{10}$$

(See the board calculation)

Multivariate Linear Regression

Apply the 3nd trick to $J(\mathbf{w})$:

$$\begin{aligned} J(\mathbf{w}) &= \frac{1}{2}(\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y}) \\ \Rightarrow \nabla_{\mathbf{w}} J(\mathbf{w}) &= \mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y} \end{aligned} \tag{10}$$

(See the board calculation)
Solving $\nabla_{\mathbf{w}} J(\mathbf{w}) = 0$ (so-called the *normal equation*), we get:

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T y) \tag{11}$$

Running time and memory?

Multivariate Linear Regression

Apply the 3nd trick to $J(\mathbf{w})$:

$$\begin{aligned} J(\mathbf{w}) &= \frac{1}{2}(\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y}) \\ \Rightarrow \nabla_{\mathbf{w}} J(\mathbf{w}) &= \mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y} \end{aligned} \tag{10}$$

(See the board calculation)
Solving $\nabla_{\mathbf{w}} J(\mathbf{w}) = 0$ (so-called the *normal equation*), we get:

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T y) \tag{11}$$

Running time and memory?

- Running time $O(d^3 + d^2 n)$
- Memory: $O(d^2 + nd)$

Optimization by Gradient Descent

$$\text{minimize} \quad f(\mathbf{x}) \tag{12}$$

where $f : \mathbb{R}^d \to \mathbf{R}$ is differentiable.

Optimization by Gradient Descent

$$\text{minimize} \quad f(\mathbf{x}) \tag{12}$$

where $f : \mathbb{R}^d \to \mathbf{R}$ is differentiable.

GradientDescent($f(.)$)
  initialize value for $\mathbf{w}$ randomly
  choose a small constant η
  repeat
    $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} f(\mathbf{w})$
  until a chosen convergent criterion satisfied

Back to Multivariate Linear Regression

We have:

$$J(\mathbf{w}) = \frac{1}{2n}(\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y})$$
$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{n}\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y}$$

Note here that we add the factor $\frac{1}{n}$ to $J(\mathbf{w})$ to for numerical stability.

Back to Multivariate Linear Regression

We have:

$$J(\mathbf{w}) = \frac{1}{2n}(\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y})$$
$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{n}\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y}$$

Note here that we add the factor $\frac{1}{n}$ to $J(\mathbf{w})$ to for numerical stability.

$\textsc{GradientDescent}(f(.))$
- initialize $\mathbf{w}$ randomly
- choose a small constant η
- **repeat**
  - $\mathbf{w} \leftarrow \mathbf{w} - \frac{\eta}{n}(\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y})$
- **until** a chosen convergent criterion satisfied

Running time and memory?

Back to Multivariate Linear Regression

We have:

$$J(\mathbf{w}) = \frac{1}{2n}(\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y})$$
$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{n}\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y}$$

Note here that we add the factor $\frac{1}{n}$ to $J(\mathbf{w})$ to for numerical stability.

```
GradientDescent(f(.))
    initialize w randomly
    choose a small constant η
    repeat
        w ← w − (η/n)(XᵀXw − Xᵀy)
    until a chosen convergent criterion satisfied
```

Running time and memory?

- Running time $O(Td^2n)$ where T is the number of updates.
- Memory: $O(dn)$.

Multivariate Linear Regression

Let's look back to the original form of $J(\mathbf{w})$

$$J(\mathbf{w}) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

Multivariate Linear Regression

Let's look back to the original form of $J(\mathbf{w})$

$$J(\mathbf{w}) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

We have:

$$\begin{aligned}\frac{\partial J(\mathbf{w})}{\partial w_j} &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{w}^T\mathbf{x}_i)\mathbf{x}_i[j] \\ &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)\mathbf{x}_i[j]\end{aligned}$$

where $\hat{y}_i = \mathbf{w}^T\mathbf{x}_i$ is the predicted version of y_i.

Multivariate Linear Regression

$\textsc{GradientDescent}(f(.))$
initialize $\mathbf{w}_0$ randomly
choose a small constant η
repeat
for $j \leftarrow 1$ to $d+1$
$\mathbf{w}[j] \leftarrow \mathbf{w}[j] - \frac{\eta}{n}(\sum_{i=1}^{n}(y_i - \hat{y}_i)\mathbf{x}_i[j])w[j]$
until a chosen convergent criterion satisfied

Running time and memory and passes?

Multivariate Linear Regression

GradientDescent($f(.)$)
  initialize $\mathbf{w}_0$ randomly
  choose a small constant η
  repeat
    for $j \leftarrow 1$ to $d+1$
      $\mathbf{w}[j] \leftarrow \mathbf{w}[j] - \frac{\eta}{n}(\sum_{i=1}^{n}(y_i - \hat{y}_i)\mathbf{x}_i[j])w[j]$
  until a chosen convergent criterion satisfied

Running time and memory and passes?

- Running time $O(Td^2n)$ where T is the number of updates.
- Memory: $O(d)$.
- We pass through the data T times.

Multivariate Linear Regression

GRADIENTDESCENT($f(.)$)
  initialize $\mathbf{w}_0$ randomly
  choose a small constant η
  repeat
    for $j \leftarrow 1$ to $d+1$
      $\mathbf{w}[j] \leftarrow \mathbf{w}[j] - \frac{\eta}{n}(\sum_{i=1}^{n}(y_i - \hat{y}_i)\mathbf{x}_i[j])w[j]$
  until a chosen convergent criterion satisfied

Running time and memory and passes?

- Running time $O(Td^2n)$ where T is the number of updates.
- Memory: $O(d)$.
- We pass through the data T times.

Question: Can we reduce T?

Stochastic Gradient Descent

Let's look closer at $J(\mathbf{w})$

$$J(\mathbf{w}) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

Which can be written as:

$$J(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} L_{\mathbf{w}}(\mathbf{x}_i)$$

where $L_{\mathbf{w}}(\mathbf{x}_i) = (y_i - \mathbf{w}^T \mathbf{x}_i)^2$ which is the loss contributed by data point i, which can be think of as:

$$J(\mathbf{w}) = \mathrm{E}_{\mathbf{x} \sim \mathcal{D}}[L_{\mathbf{w}}(\mathbf{x})]$$

Stochastic Gradient Descent

In short,

$$
\begin{aligned}
J(\mathbf{w}) &= \frac{1}{n}\sum_{i=1}^{n} L_{\mathbf{w}}(\mathbf{x}_i) \\
&= \mathrm{E}_{\mathbf{x}\sim\mathcal{D}}[L_{\mathbf{w}}(\mathbf{x})]
\end{aligned} \tag{13}
$$

Stochastic Gradient Descent

In short,

$$J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} L_{\mathbf{w}}(\mathbf{x}_i) \tag{13}$$
$$= \mathrm{E}_{\mathbf{x}\sim\mathcal{D}}[L_{\mathbf{w}}(\mathbf{x})]$$

Suppose that you take m random samples $\mathbf{x}'_1, \ldots, \mathbf{x}'_m$ from $\mathcal{D}$ and calculate:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} L_{\mathbf{w}}(\mathbf{x'}_i) \tag{14}$$

Stochastic Gradient Descent

In short,

$$J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} L_{\mathbf{w}}(\mathbf{x}_i) \\ = \mathrm{E}_{\mathbf{x}\sim\mathcal{D}}[L_{\mathbf{w}}(\mathbf{x})] \tag{13}$$

Suppose that you take m random samples $\mathbf{x}'_1, \ldots, \mathbf{x}'_m$ from $\mathcal{D}$ and calculate:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} L_{\mathbf{w}}(\mathbf{x}'_i) \tag{14}$$

We have:

$$E[\mu] = \frac{1}{m}\sum_{i=1}^{m} E[L_{\mathbf{w}}(\mathbf{x}'_i)] = \frac{1}{m}\sum_{i=1}^{m} \mathrm{E}_{\mathbf{x}\sim\mathcal{D}}[L_{\mathbf{w}}(\mathbf{x})] = \mathrm{E}_{\mathbf{x}\sim\mathcal{D}}[L_{\mathbf{w}}(\mathbf{x})] \tag{15}$$

SGD for Multivariate Linear Regression

StochasticGradientDescent($f(.)$)

initialize $\mathbf{w}_0$ randomly
choose a small constant η
choose m
repeat
  Divide the data set into $\frac{n}{m}$ random parts of size m
  for each part $\mathcal{D}_i$
    for $j \leftarrow 1$ to $d+1$
      $\mathbf{w}[j] \leftarrow \mathbf{w}[j] - \frac{\eta}{m}(\sum_{\mathbf{x}_p \in \mathcal{D}_i}(y_p - \hat{y}_p)\mathbf{x}_p[j])w[j]$
until a chosen convergent criterion satisfied

Running time and memory and passes?

SGD for Multivariate Linear Regression

StochasticGradientDescent($f(.)$)
  initialize $\mathbf{w}_0$ randomly
  choose a small constant η
  choose m
  repeat
    Divide the data set into $\frac{n}{m}$ random parts of size m
    for each part $\mathcal{D}_i$
      for $j \leftarrow 1$ to $d+1$
        $\mathbf{w}[j] \leftarrow \mathbf{w}[j] - \frac{\eta}{m}(\sum_{\mathbf{x}_p \in \mathcal{D}_i}(y_p - \hat{y}_p)\mathbf{x}_p[j])w[j]$
  until a chosen convergent criterion satisfied

Running time and memory and passes?

- Running time $O(Td^2n)$ where T is the number of epoches.
- Memory: $O(d)$ amd we pass through the data T times.
- The number of parameter updates is $T\frac{n}{m}$. In practice, $m = 2^r$ where $0 \leq r \leq 10$.

For very large data set, even $T \in [1, 10]$ suffices.