EN.600.475 Machine Learning

Linear Regression

Raman Arora

Lecture 4

February 8, 2017

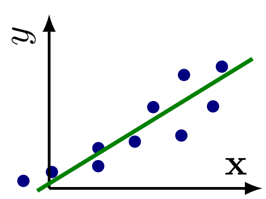
- Loss and Risk
- Least squares estimation

Slides credit: Greg Shakhnarovich

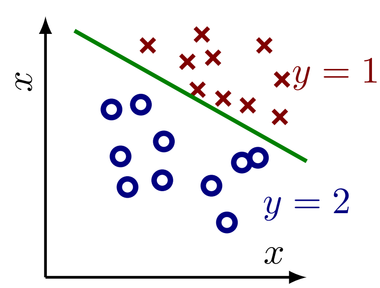
Review: supervised learning

- Task: build a mapping from input $\mathcal{X}$ output $\mathcal{Y}$
- Given a training set $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, N$, with $\mathbf{x}_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$.
- Goal (informally): predict y accurately for future xs

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



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

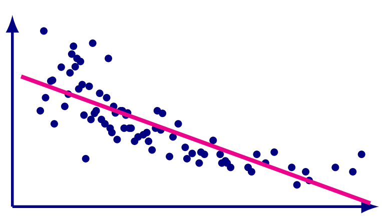


Review: supervised learning pipeline

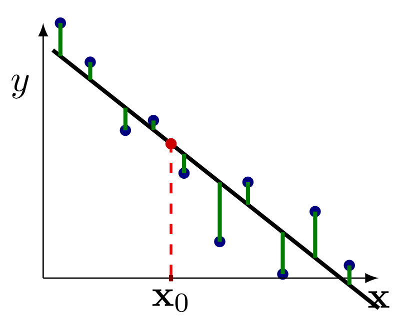
- **Set up (define)** a supervised learning problem
- **Data collection** for traning and test set.
- **Representation** choose how data are fed to the model
- **Modeling** Choose a *hypothesis (model) class*
- **Estimation (learning)** Find best hypothesis you can in the chosen class, given the data.
- **Model (class) selection**

Review: linear regression

- Two goals in mind:
  Explain the data
  Make predictions



- Model class: linear functions
- Fitting criterion, to guide selection of a function: sum of squared distances from data to the line, along y axis

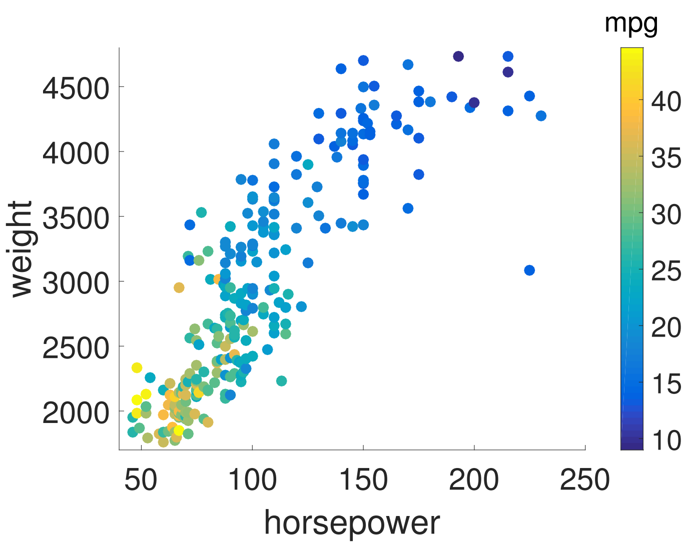


Roadmap

- General form of linear regression and least squares fit
- Loss and risk: definitions and analysis
- Analysis of error in empirical risk minimization

Multiple input variables

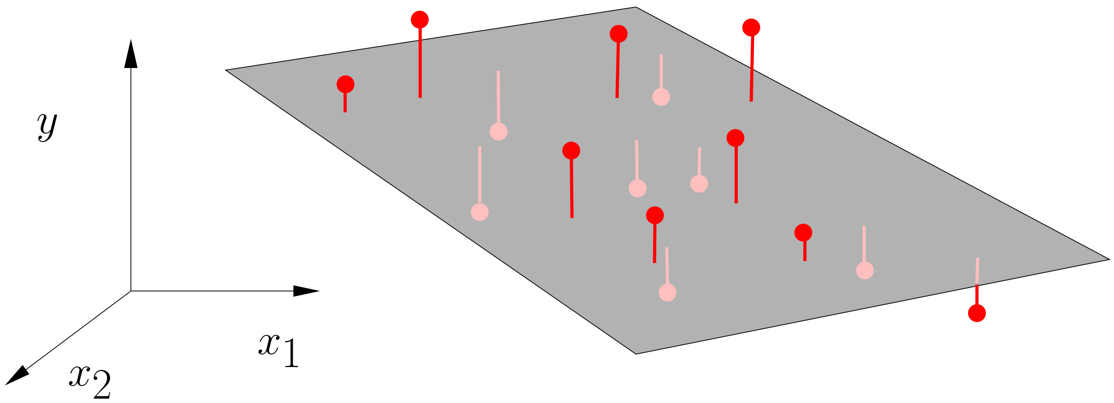
- Can consider additional features; e.g., x_1 horsepower and x_2 vehicle weight.
- We now have mapping from $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$ to y



colorbar: one possible way to convey multi-dimensional plots

Fitting a plane to data

- Can use the same criterion: minimize sum of square distances along y-axis



Linear functions

- General form:

$$\begin{aligned} f(\mathbf{x}; \mathbf{w}) &= w_0 + w_1 x_1 + \ldots + w_d x_d \\ &= \mathbf{w} \cdot \mathbf{x} \end{aligned}$$

  denoting $x_0 \equiv 1$
- 1D case ($\mathcal{X} = \mathbb{R}$): a line
- $\mathcal{X} = \mathbb{R}^2$: a plane
- *Hyperplane* in general, d-D case.

Notation

We will mostly stick to these throughout the course:

$\mathbf{x}_i$	the i-th data point in $\mathcal{X}$ (column vector)
	Often $\mathcal{X} \equiv \mathbb{R}^d$, so that $\mathbf{x}_i = [x_{i1}, \ldots, x_{id}]$
	Often assume also $x_{i0} \equiv 1$
y_i	the label of the i-th data point; $y_i \in \mathcal{Y}$
$\mathbf{x}_0, y_0$	a single test point and its (unknown) label
$\mathbf{X}$	the $N \times d$ data matrix where i-th row is $\mathbf{x}_i$
$\mathbf{y}$	the label vector $\mathbf{y} = [y_1, \ldots, y_N]$
$\mathbf{w} \cdot \mathbf{x}$	inner (dot) product, $\sum_j w_j x_j$
	sometimes write $\mathbf{w}^T \mathbf{x}$

Loss function

- Recall: target labels are in $\mathcal{Y}$ (e.g., regression: $\mathcal{Y} \equiv \mathbb{R}$)
- A *loss function* $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ maps prediction to cost, given true value:
  $\ell(\hat{y}, y)$ defines the penalty paid for predicting $\hat{y}$ when the true value is y.
- Standard choice for regression: squared loss $\ell(\hat{y}, y) = (\hat{y} - y)^2$
  is it a good loss function?..
- It is symmetric (sign of mistake doesn't matter); non-negative; gives zero loss for correct prediction
- Vaguely justifiable as "energy" of something
- Penalizes quite harshly for larger mistakes

Empirical loss

- We consider a *parametric* function $f(\mathbf{x}; \mathbf{w})$
  E.g., linear function: $f(\mathbf{x}; \mathbf{w}) = w_0 + \sum_{j=1}^{d} w_j x_{ij} = \mathbf{w} \cdot \mathbf{x}_i$
- The *empirical loss* of function $y = f(\mathbf{x}; \mathbf{w})$ on a set $\mathbf{X}$:

$$L(\mathbf{w}, \mathbf{X}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} \ell \left(f(\mathbf{x}_i; \mathbf{w}), y_i \right)$$

- LSQ minimizes the empirical loss when ℓ is squared loss.
- We care about accuracy of *predicting* labels for new examples.
  Why/when does empirical loss minimization help us achieve that?

Loss: empirical and expected

- Fundamental assumption: example $\mathbf{x}$/label y are drawn from a joint probability distribution $p(\mathbf{x}, y)$.
- Data are i.i.d.: same (unknown!) distribution for all pairs $(\mathbf{x}, y)$ in both training and test data.
- We can measure the empirical loss on training set

$$L(\mathbf{w}, \mathbf{X}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} \ell \left(f(\mathbf{x}_i; \mathbf{w}), y_i \right)$$

- The ultimate goal is to minimize the *expected loss*, also known as *risk*:

$$R(\mathbf{w}) = E_{(\mathbf{x}_0, y_0) \sim p(\mathbf{x}, y)} \left[\ell \left(f(\mathbf{x}_0; \mathbf{w}), y_0 \right) \right]$$

Loss: empirical and expected

- Empirical loss:

$$L(\mathbf{w}, \mathbf{X}, \mathbf{y}) \;=\; \frac{1}{N}\sum_{i=1}^{N} \ell\left(f(\mathbf{x}_i, \mathbf{w}), y_i\right)$$

- Risk:

$$R(\mathbf{w}) \;=\; E_{(\mathbf{x}_0, y_0)\sim p(\mathbf{x}, y)}\left[\ell\left(f(\mathbf{x}_0, \mathbf{w}), y_0\right)\right]$$

- Empirical risk minimization (ERM) approach: to the extent that the training set is a representative of the underlying distribution $p(\mathbf{x}, y)$, the empirical loss serves as a proxy for the risk (expected loss).
- Technically: estimate $p(\mathbf{x}, y)$ by the *empirical distribution* of data.

Learning via empirical loss minimization

Two steps:

- Select a restricted class $\mathcal{H}$ of *hypotheses* $f : \mathcal{X} \to \mathcal{Y}$
  Here: linear functions parametrized by $\mathbf{w}$: $f(\mathbf{x}; \mathbf{w}) \;=\; \mathbf{w} \cdot \mathbf{x}$
- Select a hypothesis $f^* \in \mathcal{H}$ based on training set (X, Y)
  Here: minimize empirical squared loss, i.e., select $f(\mathbf{x}; \mathbf{w}^*)$ where

$$\mathbf{w}^* \;=\; \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{N} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2$$

- How do we find $\mathbf{w}^* = [w_0^*, w_1^*, \ldots, w_d^*]$?

Least squares: estimation

- We need to minimize L w.r.t. $\mathbf{w}$

$$L(\mathbf{w}, \mathbf{X}, \mathbf{y}) \;=\; L(\mathbf{w}) \;=\; \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}\cdot\mathbf{x}_i)^2$$

- Necessary condition to minimize L:

$$\frac{\partial L(\mathbf{w}, \mathbf{X}, \mathbf{y})}{\partial \mathbf{w}} \;=\; \mathbf{0},$$

i.e., derivatives w.r.t. w_0, w_1,...,w_d must all be zero.

Matrix derivatives

- Scalar valued function of one variable

$$f : \mathbb{R} \to \mathbb{R} \qquad \text{derivative: } \frac{df}{dx}$$

- Scalar valued function of multiple scalar variables

$$f : \underbrace{\mathbb{R} \times \cdots \times \mathbb{R}}_{d\text{times}} \to \mathbb{R} \qquad \text{gradient: } \nabla f \;=\; \left[\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_d}\right]$$

- If we collect multiple variables in a vector: $\mathbf{x} \in \mathbb{R}^d$:

$$\nabla f \;=\; \frac{\partial f}{\partial \mathbf{x}}$$

derivative of f w.r.t. $\mathbf{x}$ has the same dimension as $\mathbf{x}$

Least squares: estimation

$$L(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}\cdot\mathbf{x}_i)^2$$

$$\begin{aligned}\frac{\partial}{\partial w_0}L(\mathbf{w}) &= \frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial w_0}(y_i - \mathbf{w}\cdot\mathbf{x}_i)^2 &= 0\\ &\Rightarrow \sum_{i=1}^{N}(y_i - \mathbf{w}\cdot\mathbf{x}_i) &= 0\end{aligned}$$

- $y_i - \mathbf{w}\cdot\mathbf{x}_i$ is the *prediction error* on the i-th example.
- $\Rightarrow$ Necessary condition for optimal $\mathbf{w}$ is that the errors have zero mean. (why does it make sense?)

Least squares: estimation

$$L(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}\cdot\mathbf{x}_i)^2$$

$$\frac{\partial}{\partial w_j}L(\mathbf{w}) = -\frac{2}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}\cdot\mathbf{x}_i)\,x_{ij} = 0.$$

- Second necessary condition: errors are *uncorrelated* with the data! (And with *any linear function* of the data)
- $d+1$ linear equations in $d+1$ unknowns $w_0, w_1, \ldots, w_d$

$$\sum_{i=1}^{N}(y_i - \mathbf{w}\cdot\mathbf{x}_i)\,x_{ij} = 0 \qquad \forall j = 1\ldots,d, \tag{1}$$

$$\sum_{i=1}^{N}(y_i - \mathbf{w}\cdot\mathbf{x}_i) = 0 \tag{2}$$

Least squares in matrix form

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ \vdots & & \vdots & \\ 1 & x_{N1} & \cdots & x_{Nd} \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_d \end{bmatrix}$$

- Predictions: $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$, errors: $\mathbf{y} - \mathbf{X}\mathbf{w}$, empirical loss:

$$\begin{aligned} L(\mathbf{w}, \mathbf{X}, \mathbf{y}) = \frac{1}{N}(\mathbf{y} - \mathbf{X}\mathbf{w}) \cdot (\mathbf{y} - \mathbf{X}\mathbf{w}) &= \frac{1}{N}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \frac{1}{N}\left(\mathbf{y}^T - \mathbf{w}^T\mathbf{X}^T\right)(\mathbf{y} - \mathbf{X}\mathbf{w}) \end{aligned}$$

Using $(AB)^T = B^T A^T$, $(A+B)^T = A^T + B^T$, $(A^T)^T = A$.

Derivative of loss

$$L(\mathbf{w}) = \frac{1}{N}\left(\mathbf{y}^T - \mathbf{w}^T\mathbf{X}^T\right)(\mathbf{y} - \mathbf{X}\mathbf{w}).$$

$$\frac{\partial \mathbf{a}^T\mathbf{b}}{\partial \mathbf{a}} = \frac{\partial \mathbf{b}^T\mathbf{a}}{\partial \mathbf{a}} = \mathbf{b}, \quad \frac{\partial \mathbf{a}^T\mathbf{B}\mathbf{a}}{\partial \mathbf{a}} = 2\mathbf{B}\mathbf{a}$$

$$\begin{aligned} \frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} &= \frac{1}{N}\frac{\partial}{\partial \mathbf{w}}\left[\mathbf{y}^T\mathbf{y} - \mathbf{w}^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{w} + \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w}\right] \\ &= \frac{1}{N}\left[\mathbf{0} - \mathbf{X}^T\mathbf{y} - (\mathbf{y}^T\mathbf{X})^T + 2\mathbf{X}^T\mathbf{X}\mathbf{w}\right] \\ &= -\frac{2}{N}\left(\mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\mathbf{w}\right) \end{aligned}$$

Least squares solution

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}) = -\frac{2}{N}\left(\mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\mathbf{w}\right) = \mathbf{0}$$
$$\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\mathbf{w} \Rightarrow \mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

- $\mathbf{X}^\dagger \triangleq \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T$ is called the *Moore-Penrose pseudoinverse* of $\mathbf{X}$.
- Linear regression in Python:

```
X[:,0]=1; X[:,1::]=x  # assumes X is right size
w=np.dot(np.linalg.pinv(X),y)
```

- Prediction: `yhat=np.dot(X,w)`

$$\hat{y} = \mathbf{w}^* \cdot \mathbf{x}_0 = \mathbf{y}^T {\mathbf{X}^\dagger}^T \mathbf{x}_0$$

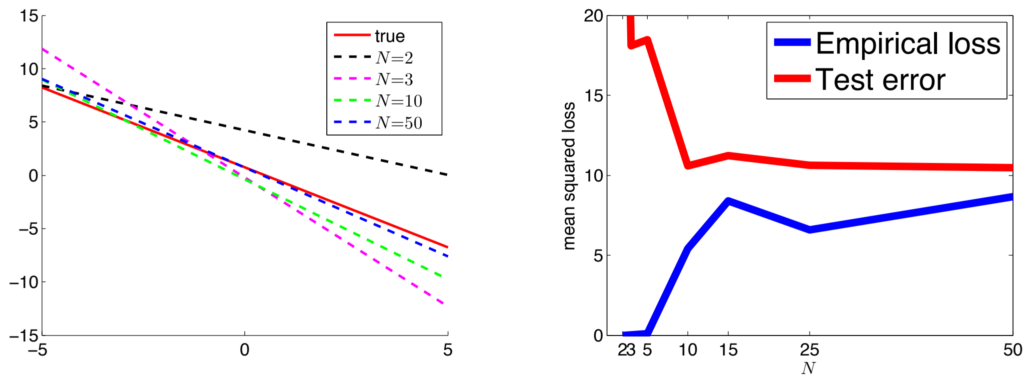
Note: we have $d+1$ numbers in $\mathbf{w}^*$ capture what the training data $\mathbf{X}, \mathbf{y}$ tell us about $\mathcal{X} \rightarrow \mathcal{Y}$ *under our model class*

Data set size and regression

- What happens when we only have a single data point (in 1D)?
  - Ill-posed problem: an infinite number of lines pass through the point and produce “perfect” fit.
- Two points in 1D?
- Two points in 2D?
- This is a general phenomenon: the amount of data needed to obtain a meaningful estimate of a model is related to the number of parameters in the model (its *complexity*).

Linear regression - generalization

- Toy experiment: fit a line to varying number of points drawn from the same distribution $p(\mathbf{x}, y)$



- A paradox?
  - The more training data we have, the "worse" is the fit;
  - But at the same time our prediction ability seems to improve.