LINEAR REGRESSION

Dr. Hilman F. Pardede

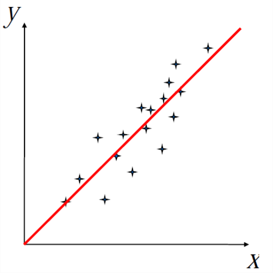
Research Center for Informatics
Indonesian Institute of Sciences

The materials are compiled from the following resources:

- https://github.com/joaquinvanschoren/ML-course
- https://www.cse.iitk.ac.in/users/piyush/courses/ml_autumn16/ML.html
- http://sli.ics.uci.edu/Classes/2015W-273a

LINEAR MODELS

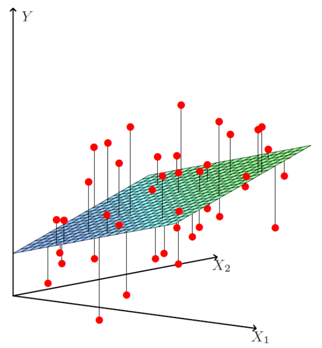
Linear models make a prediction using a linear function of the input features. Can be very powerful for or datasets with many features.
If you have more features than training data points, any target y can be perfectly modeled (on the training set) as a linear function.



- Let's assume the relationship between x and y to have a linear model $y = wx$
- Problem boils down to fitting a line to the data $\rightarrow$ optimization problem
- w is the model parameter (slope of the line here)
- Many w's (i.e., many lines) can be fit to this data
- Which one is the best

FITTING A (HYPER)PLANE TO THE DATA

- For 2-dim. inputs, we can fit a 2-dim. plane to the data



- In higher dimensions, we can likewise fit a hyperplane $w^T x = 0$
  - Defined by a D-dim vector w normal to the plane
    - Many planes are possible. Which one is the best?

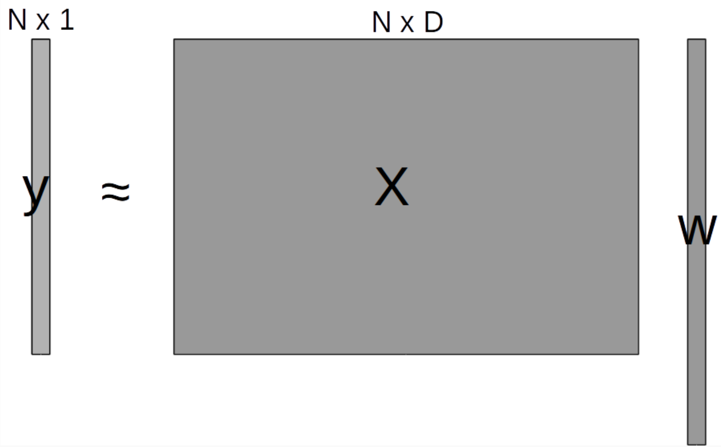
- Given: Training data with N examples $\{(x_n, y_n)\}_{n=1}^N, x_n \in \mathbb{R}^D, y_n \in \mathbb{R}$
- me the following linear model with model parameters $w \in \mathbb{R}^D$

$$y_n \approx \mathbf{w}^\top \mathbf{x}_n \Rightarrow y_n \approx \sum_{d=1}^{D} w_d x_{nd} \tag{1}$$

- The response y_n is a linear combination of the features of the inputs x_n
- $w \in \mathbb{R}^D$ is also called the (regression) weight vector
  - Can think of w_d as weight/importance of d-th feature in the data

- A simple and interpretable linear model: linear system of equations; w being the unknown

$$\mathbf{y} = \mathbf{X}\mathbf{w} \tag{2}$$



- Our linear regression model: $y_n \approx \mathbf{w}^\top x_n$. The goal is to learn $w \in \mathbb{R}^D$

- Our linear regression model: $y_n \approx \mathbf{w}^\top x_n$. The goal is to learn $w \in \mathbb{R}^D$
- The squared loss is usually used to define the loss function $\ell(y_n, \mathbf{w}^\top \mathbf{x}_n) = (y_n \mathbf{w}^\top \mathbf{x}_n)^2$

Note:

Squared loss chosen for simplicity; other losses can be used, e.g. absolute error $\ell(y_n, \mathbf{w}^\top \mathbf{x}_n) = |y_n - \mathbf{w}^\top \mathbf{x}_n|$ (more robust to outliers)

- Our linear regression model: $y_n \approx \mathbf{w}^\top x_n$. The goal is to learn $w \in \mathbb{R}^D$
- The squared loss is usually used to define the loss function $\ell(y_n, \mathbf{w}^\top \mathbf{x}_n) = (y_n \mathbf{w}^\top \mathbf{x}_n)^2$

Note:
Squared loss chosen for simplicity; other losses can be used, e.g. absolute error $\ell(y_n, \mathbf{w}^\top \mathbf{x}_n) = |y_n - \mathbf{w}^\top \mathbf{x}_n|$ (more robust to outliers)

- Using the squared loss, the total (empirical) error on the training data

$$L_{\text{emp}}(\mathbf{w}) = \sum_{n=1}^{N} \ell(y_n, \mathbf{w}^\top \mathbf{x}_n) = \sum_{n=1}^{N} (y_n - \mathbf{w}^\top \mathbf{x}_n)^2 \tag{3}$$

- Our linear regression model: $y_n \approx \mathbf{w}^\top x_n$. The goal is to learn $w \in \mathbb{R}^D$
- The squared loss is usually used to define the loss function $\ell(y_n, \mathbf{w}^\top \mathbf{x}_n) = (y_n \mathbf{w}^\top \mathbf{x}_n)^2$

Note:
Squared loss chosen for simplicity; other losses can be used, e.g. absolute error $\ell(y_n, \mathbf{w}^\top \mathbf{x}_n) = |y_n - \mathbf{w}^\top \mathbf{x}_n|$ (more robust to outliers)

- Using the squared loss, the total (empirical) error on the training data

$$L_{\text{emp}}(\mathbf{w}) = \sum_{n=1}^{N} \ell(y_n, \mathbf{w}^\top \mathbf{x}_n) = \sum_{n=1}^{N} (y_n - \mathbf{w}^\top \mathbf{x}_n)^2 \tag{3}$$

- $\mathbf{w}$ is estimated by minimizing $L_{\text{emp}}(\mathbf{w})$ w.r.t. $\mathbf{w}$ (an optimization problem)

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n=1}^{N} (y_n - \mathbf{w}^\top \mathbf{x}_n)^2 \tag{4}$$

- Taking derivative of $L_{\text{emp}}(\mathbf{w})$ w.r.t. $\mathbf{w}$ and setting it to zero:

$$\hat{\mathbf{w}} = \left(\sum_{n=1}^{N} \left(\mathbf{x}_n \mathbf{x}_n^\top \right)^{-1} \sum_{n=1}^{N} y_n \mathbf{x}_n \right) = \left(\mathbf{X}^\top \mathbf{X} \right)^{-1} \mathbf{X}^\top \mathbf{y} \tag{5}$$

- Taking derivative of $L_{\text{emp}}(\mathbf{w})$ w.r.t. $\mathbf{w}$ and setting it to zero:

$$\hat{\mathbf{w}} = \left(\sum_{n=1}^{N} \left(\mathbf{x}_n \mathbf{x}_n^\top \right)^{-1} \sum_{n=1}^{N} y_n \mathbf{x}_n \right) = \left(\mathbf{X}^\top \mathbf{X} \right)^{-1} \mathbf{X}^\top \mathbf{y} \tag{5}$$

- Analytic, closed form solution, but has some issues

- Taking derivative of $L_{\text{emp}}(\mathbf{w})$ w.r.t. $\mathbf{w}$ and setting it to zero:

$$\hat{\mathbf{w}} = \left(\sum_{n=1}^{N} \left(\mathbf{x}_n \mathbf{x}_n^\top \right)^{-1} \sum_{n=1}^{N} y_n \mathbf{x}_n \right) = \left(\mathbf{X}^\top \mathbf{X} \right)^{-1} \mathbf{X}^\top \mathbf{y} \tag{5}$$

- Analytic, closed form solution, but has some issues
  - We didn't impose any regularization on $\mathbf{w}$ (thus prone to overfitting)

- Taking derivative of $L_{\text{emp}}(\mathbf{w})$ w.r.t. $\mathbf{w}$ and setting it to zero:

$$\hat{\mathbf{w}} = \left(\sum_{n=1}^{N} \left(\mathbf{x}_n \mathbf{x}_n^\top \right)^{-1} \sum_{n=1}^{N} y_n \mathbf{x}_n \right) = \left(\mathbf{X}^\top \mathbf{X} \right)^{-1} \mathbf{X}^\top \mathbf{y} \tag{5}$$

- Analytic, closed form solution, but has some issues
  - We didn't impose any regularization on $\mathbf{w}$ (thus prone to overfitting)
  - Have to invert a $D \times D$ matrix; prohibitive especially when D (and N) is large

- Taking derivative of $L_{\text{emp}}(\mathbf{w})$ w.r.t. $\mathbf{w}$ and setting it to zero:

$$\hat{\mathbf{w}} = \left(\sum_{n=1}^{N} \left(\mathbf{x}_n \mathbf{x}_n^\top \right)^{-1} \sum_{n=1}^{N} y_n \mathbf{x}_n \right) = \left(\mathbf{X}^\top \mathbf{X} \right)^{-1} \mathbf{X}^\top \mathbf{y} \tag{5}$$

- Analytic, closed form solution, but has some issues
  - We didn't impose any regularization on $\mathbf{w}$ (thus prone to overfitting)
  - Have to invert a $D \times D$ matrix; prohibitive especially when D (and N) is large
  - The matrix $\mathbf{X}^\top \mathbf{X}$ may not even be invertible (e.g., when $D > N$). Unique solution not guaranteed

- least squares require matrix inversion
  Least Square:
  $$\hat{\mathbf{w}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y} \tag{6}$$
- This can be computationally very expensive when D is very large
- We can instead solve for **w** more efficiently using generic/specialized optimization methods on the respective loss functions
- This is called gradient-descent procedure

Gradient descent works as follow:

Gradient descent works as follow:

- Start with an initial value of $\mathbf{w} = \mathbf{w}^{(0)}$

Gradient descent works as follow:

- Start with an initial value of $\mathbf{w} = \mathbf{w}^{(0)}$
- Update w by moving along the gradient of the loss function L (L_{emp} or L_{reg})

Gradient descent works as follow:

- Start with an initial value of $\mathbf{w} = \mathbf{w}^{(0)}$
- Update w by moving along the gradient of the loss function L (L_{emp} or L_{reg})

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta \frac{\partial L}{\partial \mathbf{w}}\bigg|_{\mathbf{w}=\mathbf{w}^{(t-1)}} \tag{7}$$

where η is the learning rate

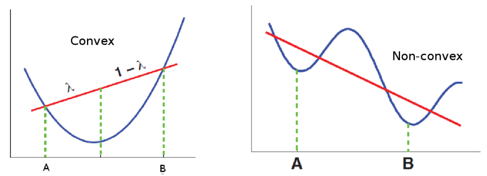
Gradient descent works as follow:

- Start with an initial value of $\mathbf{w} = \mathbf{w}^{(0)}$
- Update w by moving along the gradient of the loss function L (L_{emp} or L_{reg})

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta \frac{\partial L}{\partial \mathbf{w}}\bigg|_{\mathbf{w}=\mathbf{w}^{(t-1)}} \tag{7}$$

  where η is the learning rate
- Repeat until converge

- Guaranteed to converge to a local minima
- Converge to global minima if the function is convex



- Learning rate is important (should not be too large or too small)
- Can also use stochastic/online gradient descent for more speed-ups. Require computing the gradients using only one or a small number of examples

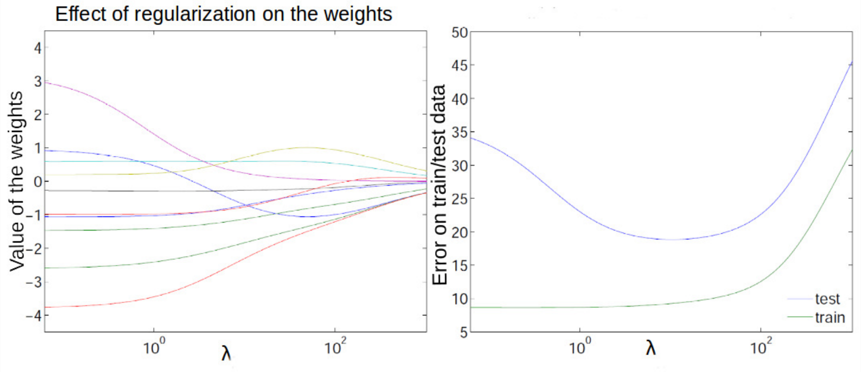
- on least square, no constraints/regularization on $\mathbf{w}$. Components $[w_1, w_2, ..., w_D]$ of $\mathbf{w}$ may become arbitrarily large. Why is this bad?
- Let's add squared ℓ_2 norm of $\mathbf{w}$ as a regularizer: $R(\mathbf{w}) = ||\mathbf{w}||^2$
- This is called "Ridge Regression" model

$$L_{\text{reg}}(\mathbf{w}) = \sum_{n=1}^{N} (y_n - \mathbf{w}^\top \mathbf{x}_n)^2 + \lambda ||\mathbf{w}||^2 \tag{8}$$

- The solution for L_{reg}:

$$\hat{\mathbf{w}} = \left(\sum_{n=1}^{N} \left(\mathbf{x}_n \mathbf{x}_n^\top + \lambda \mathbf{I}_D \right)^{-1} \sum_{n=1}^{N} y_n \mathbf{x}_n \right) = \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_D \right)^{-1} \mathbf{X}^\top \mathbf{y} \tag{9}$$

Consider ridge regression on some data with 10 features (thus the weight vector w has 10 components)



- Ridge regression is type of L2 regularization: prefers many small weights
- Lasso regression: L1 regularization prefers sparsity: many weights to be 0, others large

$$L_{\text{las}}(\mathbf{w}) = \sum_{n=1}^{N}(y_n - \mathbf{w}^\top \mathbf{x}_n)^2 + \lambda||\mathbf{w}|| \tag{10}$$

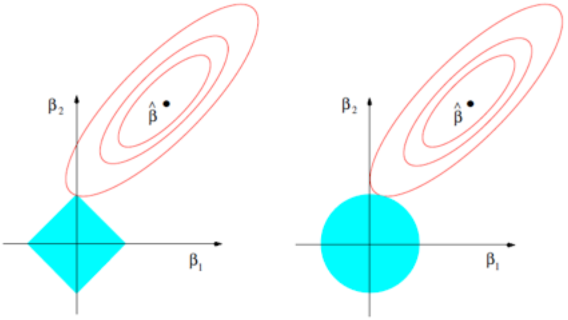
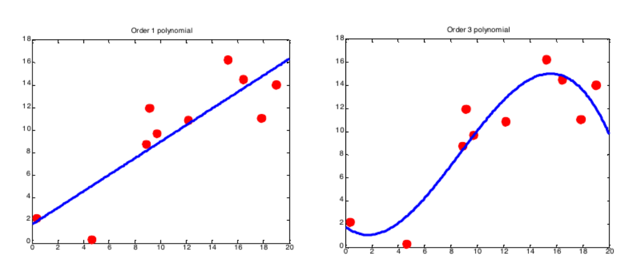


FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions* $|\beta_1| + |\beta_2| \leq t$ *and* $\beta_1^2 + \beta_2^2 \leq t^2$*, respectively, while the red ellipses are the contours of the least squares error function.*

- Adding more features or transforming features could improve the performance of ML systems
- Nonlinear lines could fit better
  - Ex: higher-order polynomials



- Sometimes useful to think of "feature transform"
- $D\{(X^{(i)}, y^{(i)})\} \Rightarrow D\{([X^{(i)}, (X^{(i)})^2, (X^{(i)})^3], y^{(i)})\}$
- $y = w_0 + w_1 x_1 \Rightarrow y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$ where $x_1 = X^{(i)}$, $x_2 = (X^{(i)})^2$, $x_3 = (X^{(i)})^3$
- Fit the same way

$$y_n \approx \mathbf{w}^\top \Phi(x_n) \tag{11}$$

- "Linear regression" = linear in the parameters
    - Features we can make as complex as we want!

- In general, can use any features we think are useful
- Other information about the problem
  - Sq. footage, location, age, ...
- Polynomial functions
  - Features $[1, x, x^2, x^3, ...]$
- Other functions: $1/x$, $\sqrt{x}$, $x_1 \times x_2, \dots$

- Investigate on using polynomial features on Boston data.
- Investigate and analyse how the value of alpha affect the performance using ridge regression
- Investigate the same thing on Lasso regression
- Comments on overfitting and underfitting