CS 1675 Intro to Machine Learning
Lecture 10

Linear regression

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

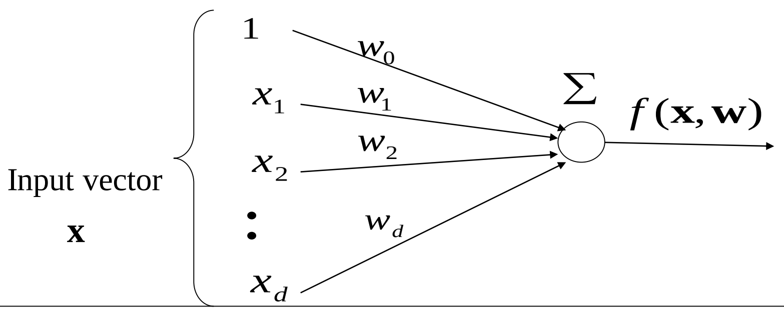
Linear regression

- **Shorter (vector) definition of the model**
  - Include bias constant in the input vector

$$\mathbf{x} = (1, x_1, x_2, \cdots x_d)$$

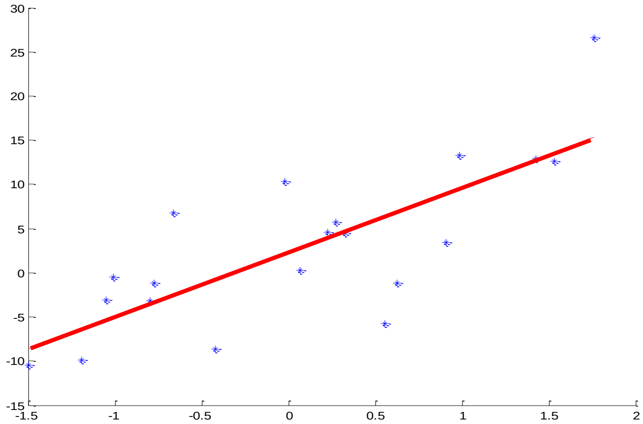
$$f(\mathbf{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots w_d x_d = \mathbf{w}^T \mathbf{x}$$

$$\mathbf{w} = [w_0, w_1, \ldots w_k]^T$$ **- parameters (weights)**



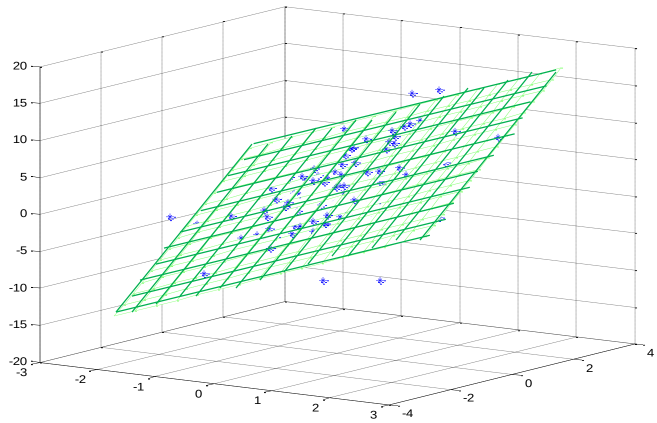
Linear regression. Example

- 1 dimensional input $\mathbf{x} = (x_1)$



Linear regression. Example.

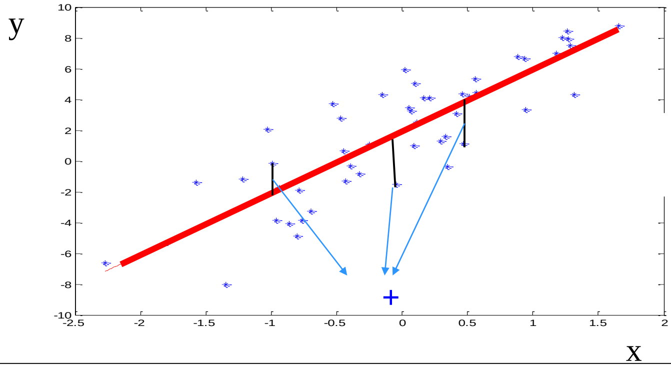
- 2 dimensional input $\mathbf{x} = (x_1, x_2)$



Linear regression: error

- **Data:** $D_i = < \mathbf{x}_i, y_i >$
- **Function:** $\mathbf{x}_i \rightarrow f(\mathbf{x}_i)$
- **Error**: a measure of misfit of the model and the data

$$J_n = \frac{1}{n} \sum_{i=1,..n} (y_i - f(\mathbf{x}_i, \mathbf{w}))^2$$



Solving linear regression

- The optimal set of weights satisfies:

$$\nabla_{\mathbf{w}} (J_n(\mathbf{w})) = -\frac{2}{n} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i = \overline{\mathbf{0}}$$

Leads to a **system of linear equations (SLE)** with $d+1$ unknowns of the form

$$\mathbf{A}\mathbf{w} = \mathbf{b}$$

$$w_0 \sum_{i=1}^{n} x_{i,0} x_{i,j} + w_1 \sum_{i=1}^{n} x_{i,1} x_{i,j} + \ldots + w_j \sum_{i=1}^{n} x_{i,j} x_{i,j} + \ldots + w_d \sum_{i=1}^{n} x_{i,d} x_{i,j} = \sum_{i=1}^{n} y_i x_{i,j}$$

Solution to SLE: $\boxed{\mathbf{w} = \mathbf{A}^{-1}\mathbf{b}}$

Assuming **X** is an *nxd* data matrix with rows corresponding to examples and columns to inputs, and ***y*** is nx1 vector of outputs, then $\mathbf{w} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y}$

Gradient descent solution

Objective: optimize the weights in the linear regression model

$$J_n = Error(\mathbf{w}) = \frac{1}{n} \sum_{i=1,..n} (y_i - f(\mathbf{x}_i, \mathbf{w}))^2$$

An alternative to SLE solution:

- **Gradient descent**

  Idea:
  - Adjust weights in the direction that improves the Error
  - The gradient tells us what is the right direction

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} Error_i(\mathbf{w})$$

$\alpha > 0$ - a **learning rate** (scales the gradient changes)

Batch vs online gradient algorithm

- The error function defined on the complete dataset *D*

$$J_n = Error(\mathbf{w}) = \frac{1}{n} \sum_{i=1,..n} (y_i - f(\mathbf{x}_i, \mathbf{w}))^2$$

- We say we are learning the model in **the batch mode**:
  - All examples are available at the time of learning
  - Weights are optimized with respect to all training examples
- An alternative is to learn the model in **the online mode**

$$J_{\text{online}} = Error_i(\mathbf{w}, \mathbf{x}_i) = \frac{1}{2}(y_i - f(\mathbf{x}_i, \mathbf{w}))^2$$

  - Examples are arriving sequentially
  - Model weights are updated after every example
  - If needed examples seen can be forgotten

Online gradient descent algorithm

Online-linear-regression (*stopping_criterion*)

  Initialize weights $\mathbf{w} = (w_0, w_1, w_2 \ldots w_d)$

  initialize i=1;

  while *stopping_criterion = FALSE*

    select the next data point $D_i = (\mathbf{x}_i, y_i)$

    set learning rate $\alpha(i)$

    update weight vector $\mathbf{w} \leftarrow \mathbf{w} + \alpha(i)(y_i - f(\mathbf{x}_i, \mathbf{w}))\mathbf{x}_i$

  end

return weights

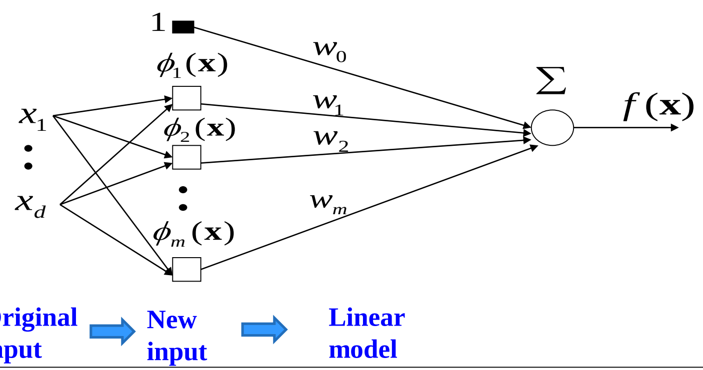
Advantages: very easy to implement, works on continuous data streams

Extensions of simple linear model

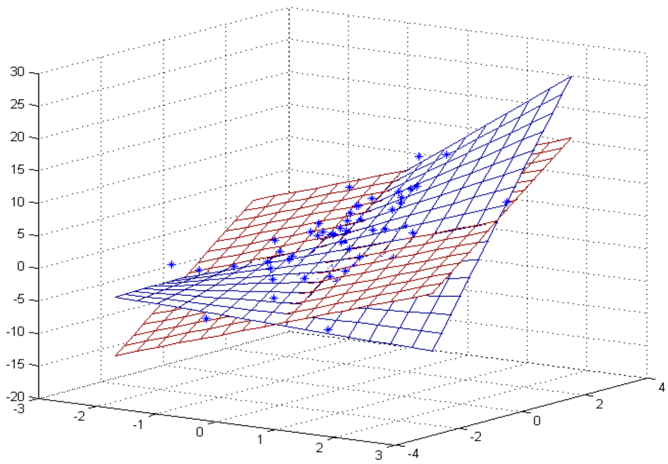
Replace inputs to linear units with *m* **feature (basis) functions** to model **nonlinearities**

$$f(\mathbf{x}) = w_0 + \sum_{j=1}^{m} w_j \phi_j(\mathbf{x})$$

$\phi_j(\mathbf{x})$ - an arbitrary function of $\mathbf{x}$

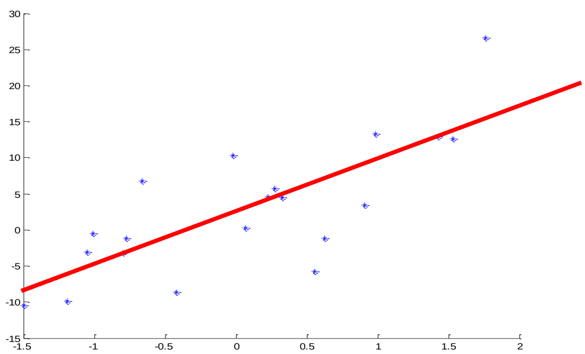


Non-linear (quadratic) model



Linear regression model

- **Linear model:** $y = f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x}$



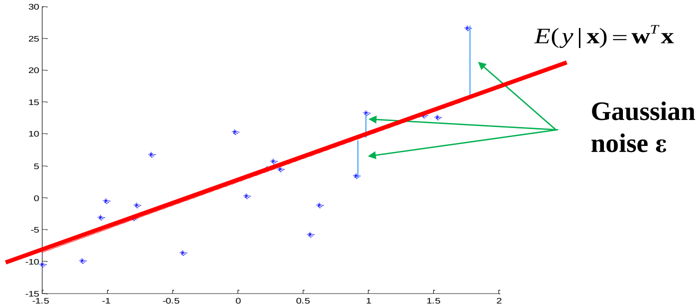
- **Notice: the above model does not explain variation in observed** ys **for the data**

Statistical model of regression

A statistical model of linear regression:

$$y = \mathbf{w}^T \mathbf{x} + \varepsilon \qquad\qquad \varepsilon \sim \mathrm{N}(0, \sigma^2)$$

ε is a random noise, represents deviations not captured with $\mathbf{w}^T\mathbf{x}$



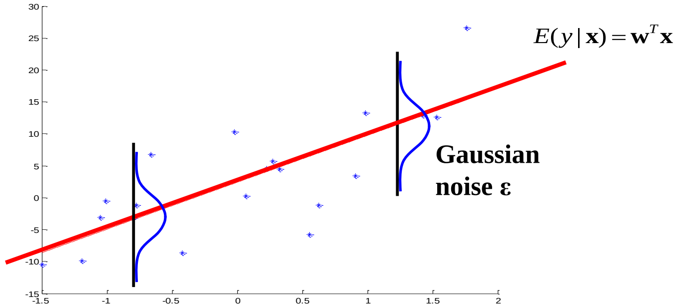
$$y \sim N(\mathbf{w}^T\mathbf{x}, \sigma^2)$$

Statistical model of regression

A statistical model of linear regression:

$$y = \mathbf{w}^T \mathbf{x} + \varepsilon \qquad\qquad \varepsilon \sim \mathrm{N}(0, \sigma^2)$$

ε is a random noise, represents deviations not captured with $\mathbf{w}^T\mathbf{x}$



$$y \sim N(\mathbf{w}^T\mathbf{x}, \sigma^2)$$

Statistical model of regression

A statistical model of linear regression:

$$y = \mathbf{w}^T \mathbf{x} + \varepsilon \qquad\qquad \varepsilon \sim \mathrm{N}(0, \sigma^2)$$

$$y \sim N(\mathbf{w}^T \mathbf{x}, \sigma^2)$$

- **The conditional distribution of y given x**

$$p(y \mid \mathbf{x}, \mathbf{w}, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2\sigma^2} (y - \mathbf{w}^T \mathbf{x})^2 \right]$$

$$E(y \mid \mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

ML estimation of the parameters

- **likelihood of predictions** = the probability of observing outputs y in D given $\mathbf{w}, \sigma$

$$L(D, \mathbf{w}, \sigma) = \prod_{i=1}^{n} p(y_i \mid \mathbf{x}_i, \mathbf{w}, \sigma)$$

- **Maximum likelihood estimation of parameters** w
  - parameters maximizing the likelihood of predictions

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \prod_{i=1}^{n} p(y_i \mid \mathbf{x}_i, \mathbf{w}, \sigma)$$

- **Log-likelihood** trick for the ML optimization
  - Maximizing the log-likelihood is equivalent to maximizing the likelihood

$$l(D, \mathbf{w}, \sigma) = \log(L(D, \mathbf{w}, \sigma)) = \log \prod_{i=1}^{n} p(y_i \mid \mathbf{x}_i, \mathbf{w}, \sigma)$$

ML estimation of the parameters

- **Using conditional density**

$$p(y \mid \mathbf{x}, \mathbf{w}, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp[-\frac{1}{2\sigma^2}(y - f(\mathbf{x}, \mathbf{w}))^2]$$

- **We can rewrite the log-likelihood as**

$$l(D, \mathbf{w}, \sigma) = \log(L(D, \mathbf{w}, \sigma)) = \log \prod_{i=1}^{n} p(y_i \mid \mathbf{x}_i, \mathbf{w}, \sigma)$$

$$= \sum_{i=1}^{n} \log p(y_i \mid \mathbf{x}_i, \mathbf{w}, \sigma) = \sum_{i=1}^{n} \left[-\frac{1}{2\sigma^2}(y_i - \mathbf{w}^T \mathbf{x}_i)^2 - c(\sigma) \right]$$

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

Did we see a similar expression before?

ML estimation of the parameters

- **Using conditional density**

$$p(y \mid \mathbf{x}, \mathbf{w}, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp[-\frac{1}{2\sigma^2}(y - f(\mathbf{x}, \mathbf{w}))^2]$$

- **We can rewrite the log-likelihood as**

$$l(D, \mathbf{w}, \sigma) = \log(L(D, \mathbf{w}, \sigma)) = \log \prod_{i=1}^{n} p(y_i \mid \mathbf{x}_i, \mathbf{w}, \sigma)$$

$$= \sum_{i=1}^{n} \log p(y_i \mid \mathbf{x}_i, \mathbf{w}, \sigma) = \sum_{i=1}^{n} \left\{ -\frac{1}{2\sigma^2}(y_i - \mathbf{w}^T \mathbf{x}_i)^2 - c(\sigma) \right\}$$

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

- **Maximizing the predictive log likelihood** with regard to **w**, is **equivalent to minimizing the mean squared error function**

ML estimation of parameters

- **Criteria based on the mean squares error function and the log likelihood of the output are related**

$$J_{online}(y_i, \mathbf{x}_i) = \frac{1}{2\sigma^2} \log\ p(y_i \mid \mathbf{x}_i, \mathbf{w}, \sigma) + c(\sigma)$$

- **We know how to optimize parameters *w***
  - the same approach as used for the least squares fit
- **But what is the ML estimate of the variance of the noise?**
- Maximize $l(D, \mathbf{w}, \sigma)$ with respect to variance

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i, \mathbf{w}^*))^2$$

= **mean square prediction error for the best predictor**

Regularized linear regression

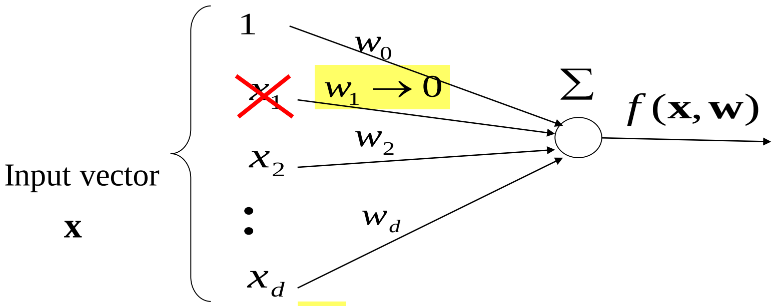
- If the number of parameters is large relative to the number of data points used to train the model, we face the threat of **overfitting** (generalization error of the model goes up)
- The prediction accuracy can be often improved by setting some coefficients (weights) of the model to zero
  - Increases the bias, reduces the variance of estimates
- **Solutions:**
  - **Subset selection**
  - **Ridge regression** ⬅
  - **Lasso regression** ⬅
  - **Principal component regression**

Regularization: motivation

- If the model is too complex and can cause overfitting, its prediction accuracy can be improved by **removing some inputs from the model = setting their coefficients to zero**

$$f(\mathbf{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + \dots w_d x_d = \mathbf{w}^T \mathbf{x}$$

$w_0, w_1, \dots w_k$ - **parameters (weights)**



$$f(\mathbf{x}) = w_0 x_0 + 0 x_1 + w_2 x_2 + w_3 x_3 + \dots w_d x_d = \mathbf{w}^T \mathbf{x}$$

Ridge regression

Question: how to force the weights to 0 ?

- Error function for the standard least squares estimates:

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

- **We seek:** $\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1,..n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$

- **Ridge regression:**

$$J_n(\mathbf{w}) = \boxed{\frac{1}{n} \sum_{i=1,..n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2} + \boxed{\lambda \|\mathbf{w}\|_{L2}^2}$$

Fit to data **Model complexity penalty**

- Where $\|\mathbf{w}\|_{L2}^2 = \sum_{i=0}^{d} w_i^2$ and $\lambda \geq 0$

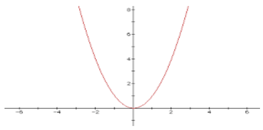
- What does the new error function do?

Ridge regression

Ridge regression:

$$J_n(\mathbf{w}) = \frac{1}{n}\sum_{i=1,..n}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 + \lambda\|\mathbf{w}\|_{L2}^2$$

Term $\|\mathbf{w}\|_{L2}^2 = \sum_{i=0}^{d} w_i^2$



- penalizes non-zero weights with the cost that is proportional to λ (a **shrinkage coefficient**)
- If an input attribute x_j has a small effect on improving the error function it is "shut down" (driven to 0) by the penalty term
- Inclusion of a shrinkage penalty is often referred to as **regularization.**

Regularized linear regression

How to solve the least squares problem if the error function is enriched by the regularization term $\lambda\|\mathbf{w}\|^2$?

Answer: The solution to the optimal set of weights w is obtained again by solving a set of linear equation.

Standard linear regression:

$$\nabla_{\mathbf{w}}(J_n(\mathbf{w})) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - \mathbf{w}^T\mathbf{x}_i)\mathbf{x}_i = \bar{\mathbf{0}}$$

Solution: $\mathbf{w}^* = (\mathbf{X}^\mathbf{T}\mathbf{X})^{-1}\mathbf{X}^\mathbf{T}\mathbf{y}$

where **X** is an *nxd* matrix with rows corresponding to examples and columns to inputs

Regularized linear regression:

$$\mathbf{w}^* = (\lambda\mathbf{I} + \mathbf{X}^\mathbf{T}\mathbf{X})^{-1}\mathbf{X}^\mathbf{T}\mathbf{y}$$

Lasso regression

- **Standard regression:**

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

- **Lasso regression/regularization:**

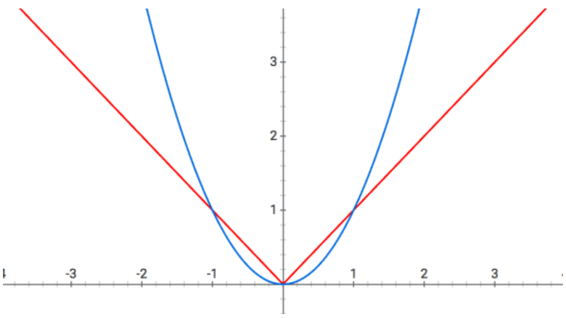
$$J_n(\mathbf{w}) = \underbrace{\boxed{\frac{1}{n} \sum_{i=1,..n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2}}_{\textbf{Fit to data}} + \underbrace{\boxed{\lambda \|\mathbf{w}\|_{L1}}}_{\textbf{Model complexity penalty}}$$

$$\|\mathbf{w}\|_{L1} = \sum_{i=0}^{d} | w_i | \qquad \text{and} \qquad \lambda \geq 0$$

- L1 is more aggressive pushing the weights to 0 compared to L2

Lasso vs Ridge penalty

- Lasso (L1) penalty $\|\mathbf{w}\|_{L1} = \sum_{i=0}^{d} | w_i |$
- Ridge (L2) penalty $\|\mathbf{w}\|_{L2}^2 = \sum_{i=0}^{d} w_i^2$



- L1 is more aggressive pushing the weights to 0 compared to L2