Machine Learning

CSE 6363 (Fall 2019)

Lecture 3 Classification, Linear Regression

Dajiang Zhu, Ph.D.

Department of Computer Science and Engineering

Slides of this course (CSE6363) courtesy: Dr. Heng Huang, Dr. Aarti Singh, Dr.**Tibshirani**

Supervised Learning

Data: $D = \{d_1, d_2, .., d_n\}$ **a set of *n* examples**

$d_i = < \mathbf{x}_i, y_i >$

$\mathbf{x}_i$ is input vector, and y is desired output (given by a teacher)

Objective: learn the mapping $f : X \rightarrow Y$

s.t. $y_i \approx f(x_i)$ for all $i = 1,.., n$

Two types of problems:

- **Regression:** X discrete or continuous $\longrightarrow$ Y is **continuous**
- **Classification:** X discrete or continuous $\longrightarrow$ Y is **discrete**

Ref: Milos Hauskrecht

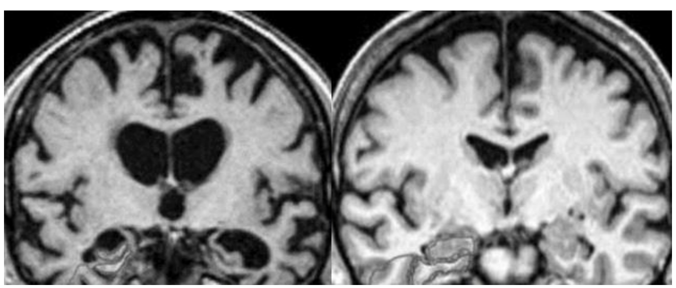
Discrete to Continuous Labels

- Classification

Science?
Sports?
News?
…

X = Document

Y = Topic



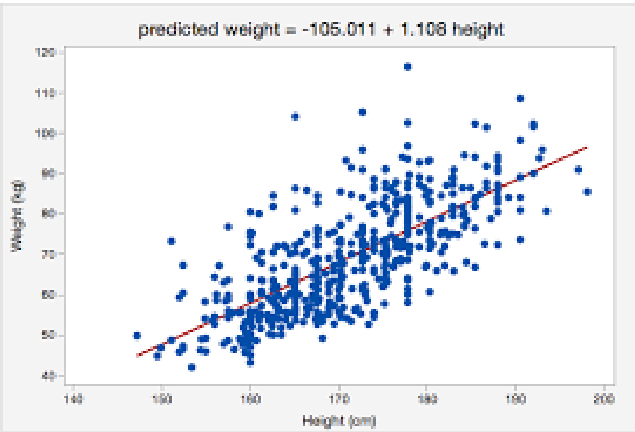
Alzhermer’s Disease (AD) patient?
Normal people?

X = Brain image

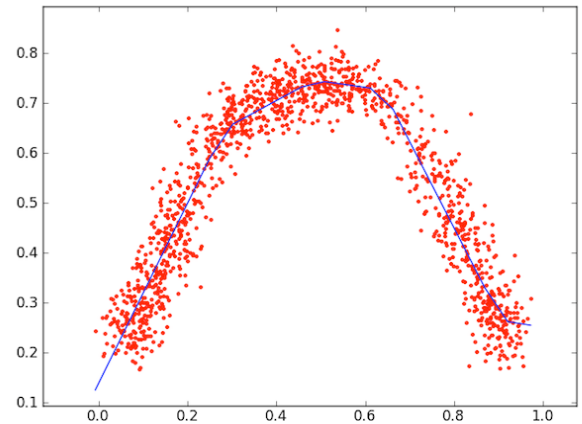
Y = Patien or not

Discrete to Continuous Labels

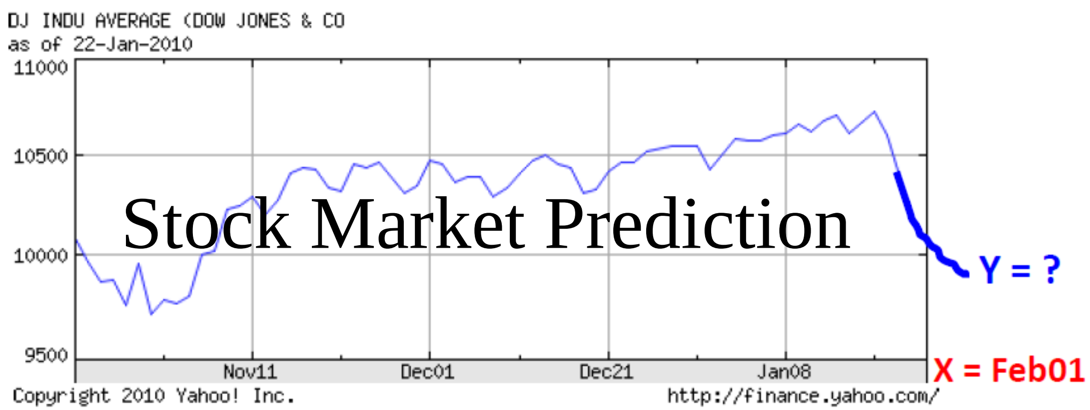
- Regression



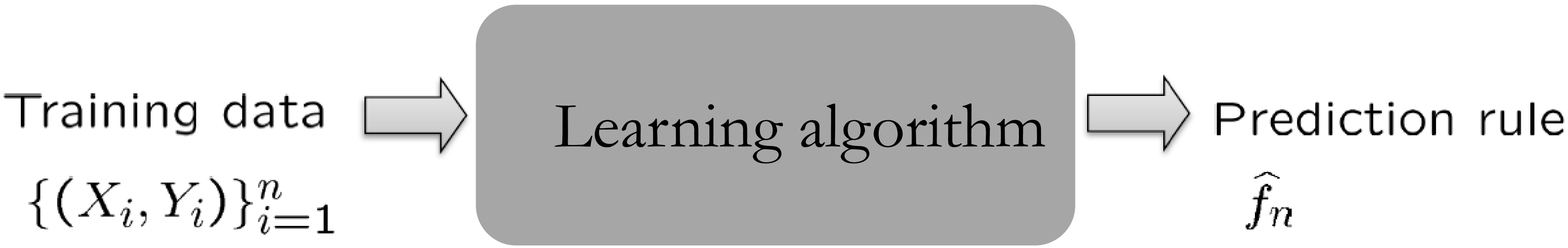
https://onlinecourses.science.psu.edu/stat200/node/81



https://blog.biolab.si/tag/regression/



Regression Algorithms



Linear Regression

Lasso, Ridge regression (Regularized Linear Regression)

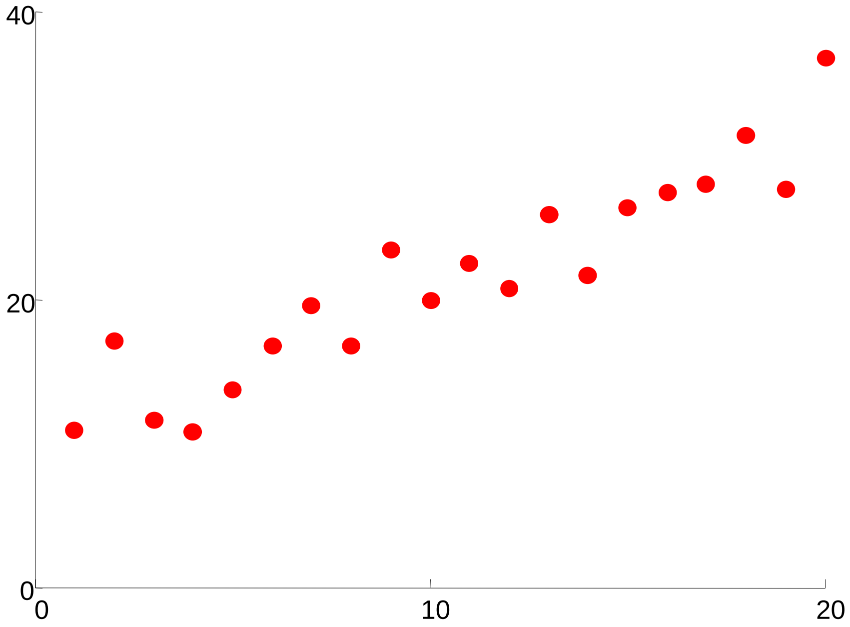
Nonlinear Regression

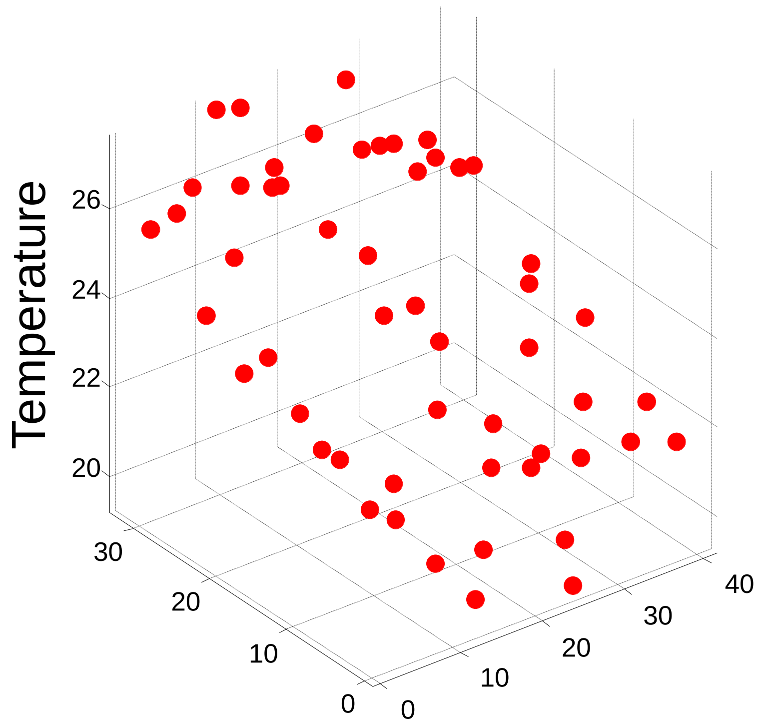
Kernel Regression

Regression Trees, Splines, Wavelet estimators, …

Ref: Milos Hauskrecht

Linear Regression

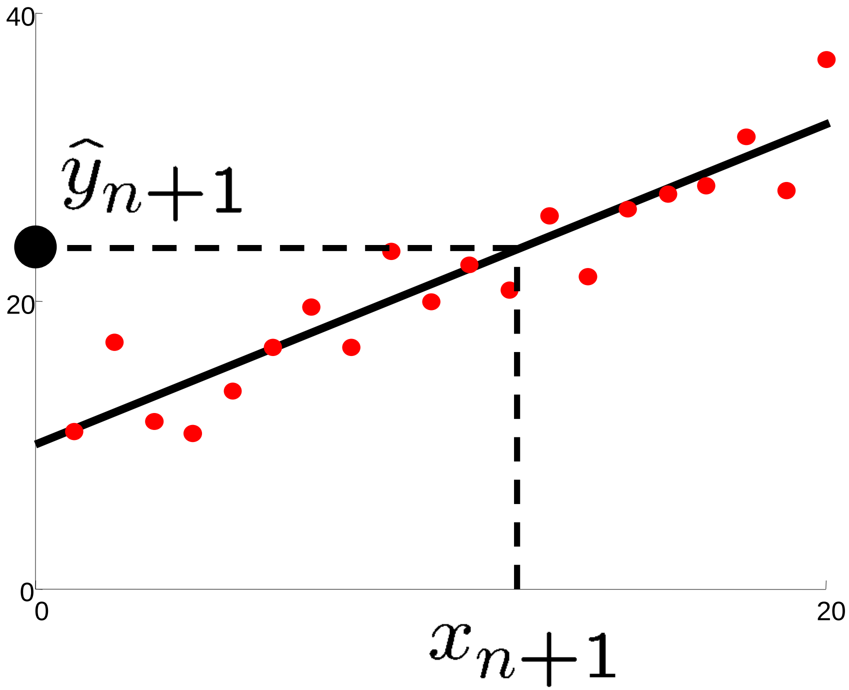


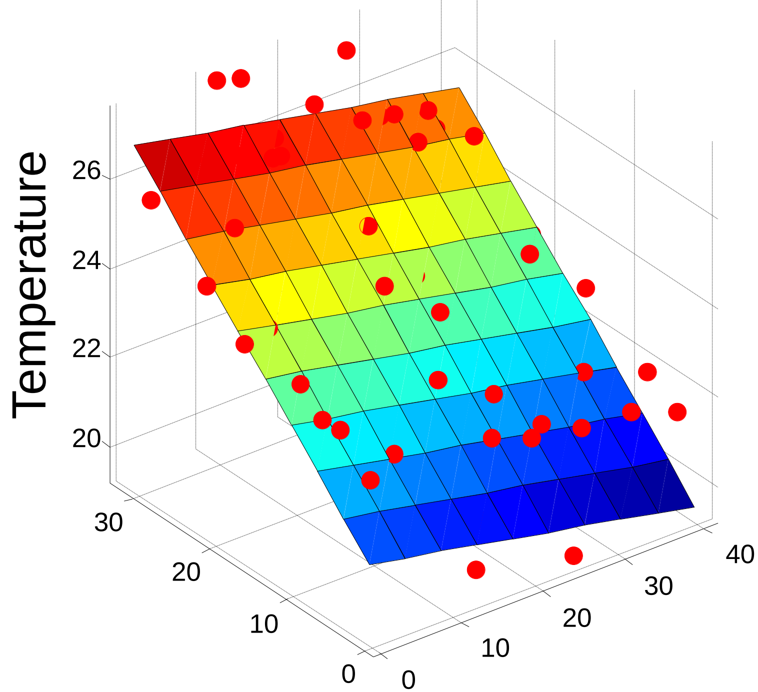


Given examples $(x_i, y_i)_{i=1\dots n}$

Predict y_{n+1} given a new point x_{n+1}

Linear Regression





Prediction

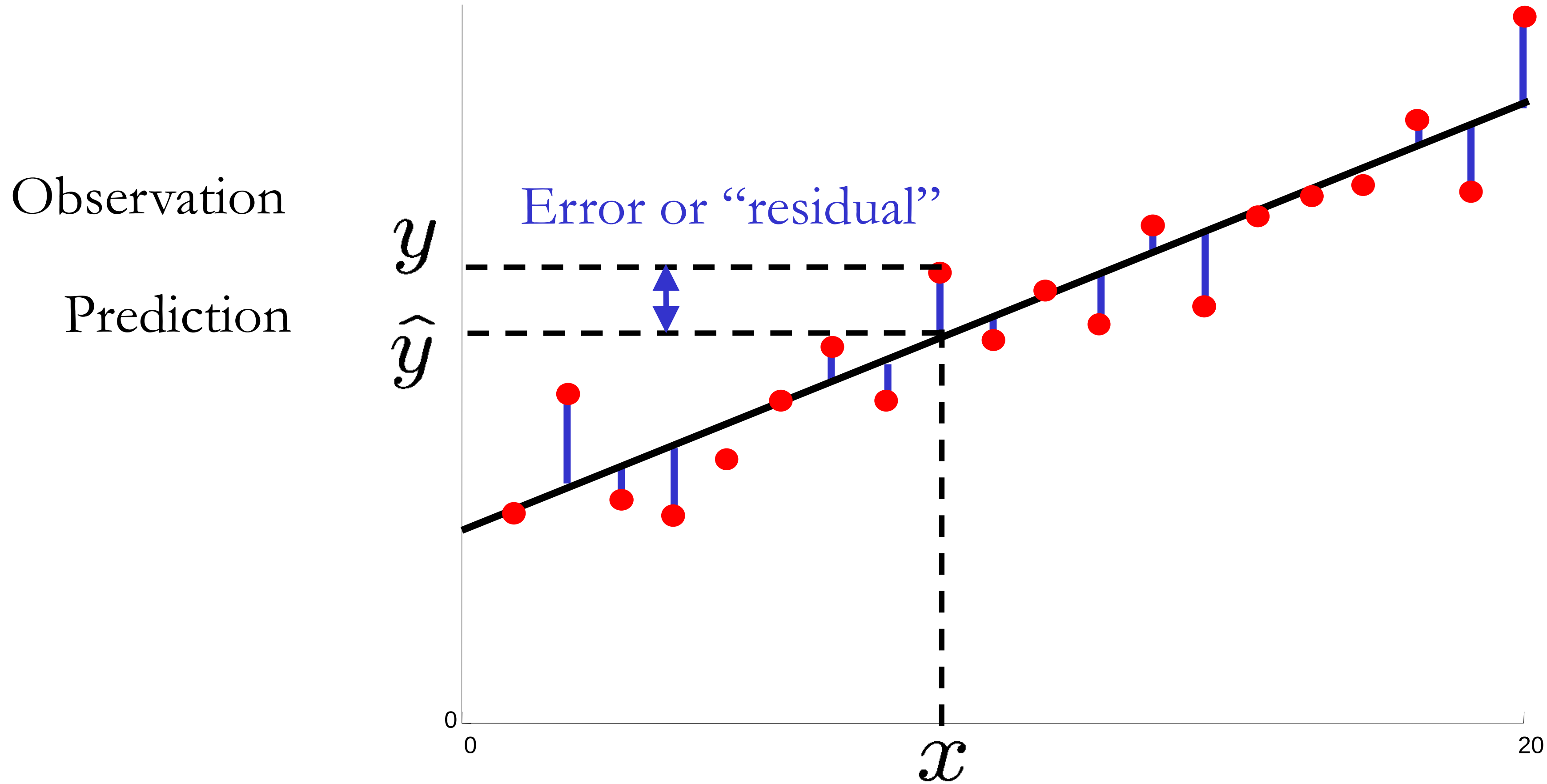
$$\hat{y}_i = w_0 + w_1 x_i$$

Pre

$$\hat{y}_i = \begin{pmatrix} 1 & x_{i,1} & x_{i,2} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} x_{i,2}$$

$$= X_i^\top w$$

Ordinary Least Squares (OLS)



Sum squared error $\sum_i (X_i^\top w - y_i)^2$

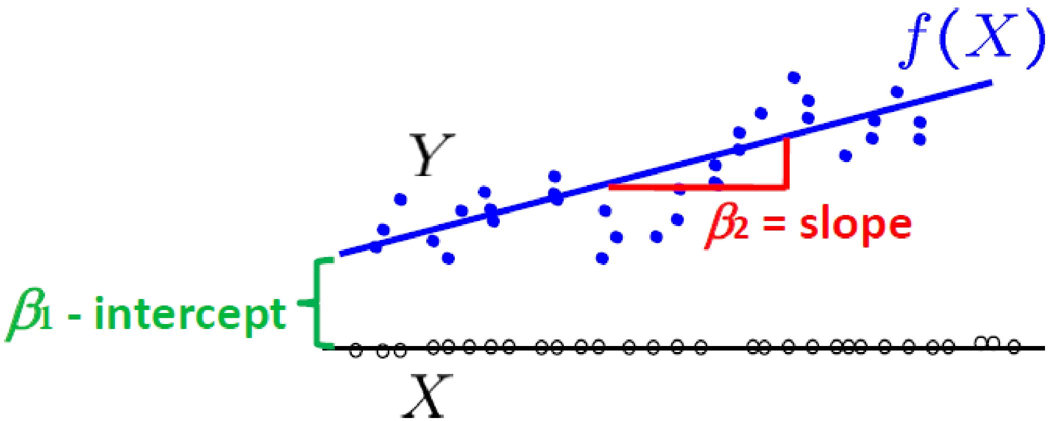
Least Squares Estimator

$$\widehat{f}_n^L = \arg\min_{f \in \mathcal{F}_L} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$$

$\mathcal{F}_L$ - Class of Linear functions

Uni-variate case:

$$\mathrm{Y} \quad = \beta_1 + \beta_2 X$$



Least Squares Estimator – Example

Goal: To minimize

$$D = \sum_{i=1}^{n} {d_i}^2 = \sum_{i=1}^{n} (y_i - \beta_1 - \beta_2 x_i)^2$$

Least Squares Estimator

Multi-variate case:

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} (X_i \beta - Y_i)^2$$

$$= \arg\min_{\beta} \frac{1}{n} (\mathbf{A}\beta - \mathbf{Y})^T (\mathbf{A}\beta - \mathbf{Y})$$

$$\mathbf{A} = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} X_1^{(1)} & \ldots & X_1^{(p)} \\ \vdots & \ddots & \vdots \\ X_n^{(1)} & \ldots & X_n^{(p)} \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} \mathbf{Y}_1 \\ \vdots \\ \mathbf{Y}_n \end{bmatrix}$$

Least Squares Estimator

Multi-variate case:

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}(\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y})$$

$$\mathrm{E}(\beta) = (\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y})$$

$$\Downarrow$$

$$\left.\frac{\partial \mathrm{E}(\beta)}{\partial \beta}\right|_{\widehat{\beta}} = 0$$

Least Squares Estimator

Multi-variate case:

$$(\mathbf{A}^T\mathbf{A})\widehat{\beta} = \mathbf{A}^T\mathbf{Y}$$

p xp   p x1   p x1

If $(A^T A)$ is invertable, we have:

$$\boxed{\widehat{\beta} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y}}$$

You need this for your project!

Project 1: Linear regression for classification

Data: http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica

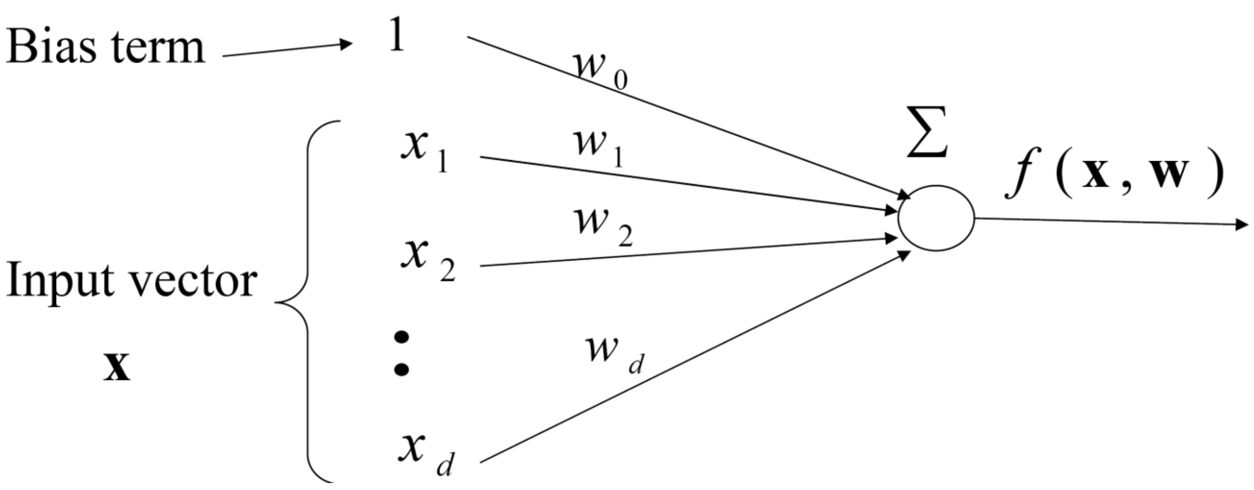
```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
```

Linear Regression

- **Function** $f : X \rightarrow Y$ is a linear combination of input components

$$f(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \ldots w_d x_d = w_0 + \sum_{j=1}^{d} w_j x_j$$

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



Ref: Milos Hauskrecht

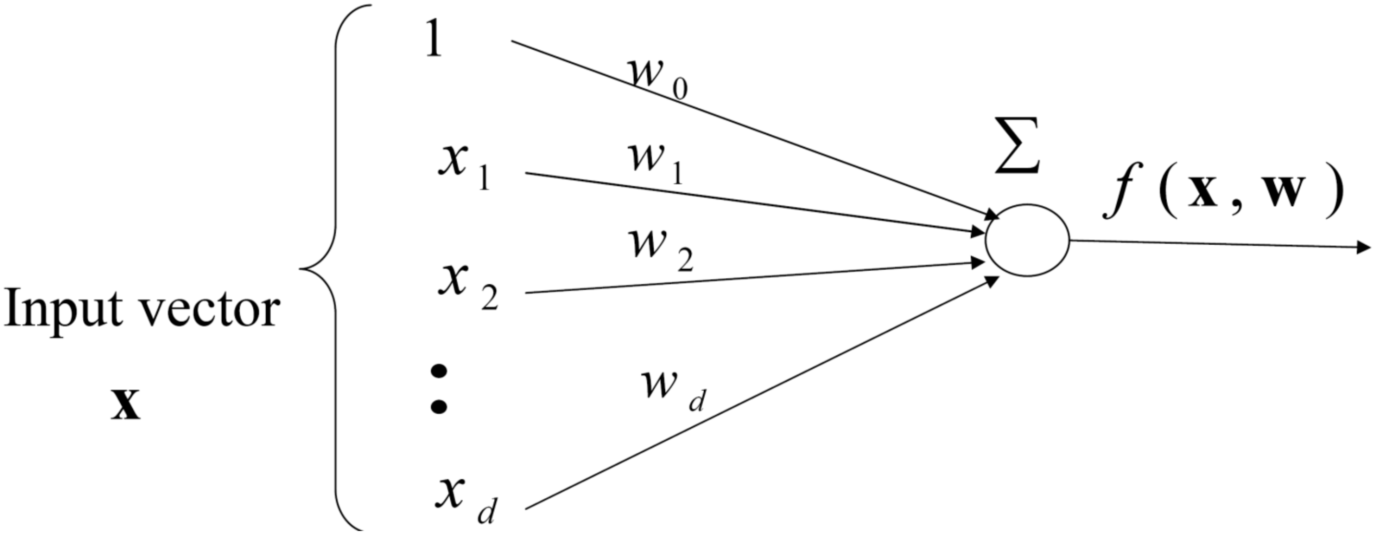
Linear Regression

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

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

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

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



Ref: Milos Hauskrecht

Examples

- Voltage -> Temperature
- Stock prediction -> Money
- Processes, memory -> Power consumption
- Protein structure -> Energy
- Robot arm controls -> Torque at effector
- Location, industry, past losses -> Premium

Linear Regression. Optimization.

- We want the **weights minimizing the error**

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

- For the optimal set of parameters, derivatives of the error with respect to each parameter must be 0

$$\frac{\partial}{\partial w_j} J_n(\mathbf{w}) = -\frac{2}{n} \sum_{i=1}^{n} (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d}) x_{i,j} = 0$$

- **Vector of derivatives:**

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

Ref: Milos Hauskrecht

Linear Regression. Optimization.

- $\text{grad}_{\mathbf{w}}(J_n(\mathbf{w})) = \bar{\mathbf{0}}$ defines a set of equations in **w**

$$\frac{\partial}{\partial w_0} J_n(\mathbf{w}) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d}) = 0$$

$$\frac{\partial}{\partial w_1} J_n(\mathbf{w}) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d}) x_{i,1} = 0$$

$$\ldots$$

$$\frac{\partial}{\partial w_j} J_n(\mathbf{w}) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d}) x_{i,j} = 0$$

$$\ldots$$

$$\frac{\partial}{\partial w_d} J_n(\mathbf{w}) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d}) x_{i,d} = 0$$

Ref: Milos Hauskrecht

Solving Linear Regression

$$\frac{\partial}{\partial w_j} J_n(\mathbf{w}) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d})x_{i,j} = 0$$

By rearranging the terms we get a **system of linear equations** with $d+1$ unknowns

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

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

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

$$\bullet\bullet\bullet$$

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

$$\bullet\bullet\bullet$$

Ref: Milos Hauskrecht

Solving Linear Regression

- The optimal set of weights satisfies:

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

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:

$$\boxed{\mathbf{w} = \mathbf{A}^{-1}\mathbf{b}}$$

Ref: Milos Hauskrecht

Gradient Descent Solution

Goal: the weight optimization 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

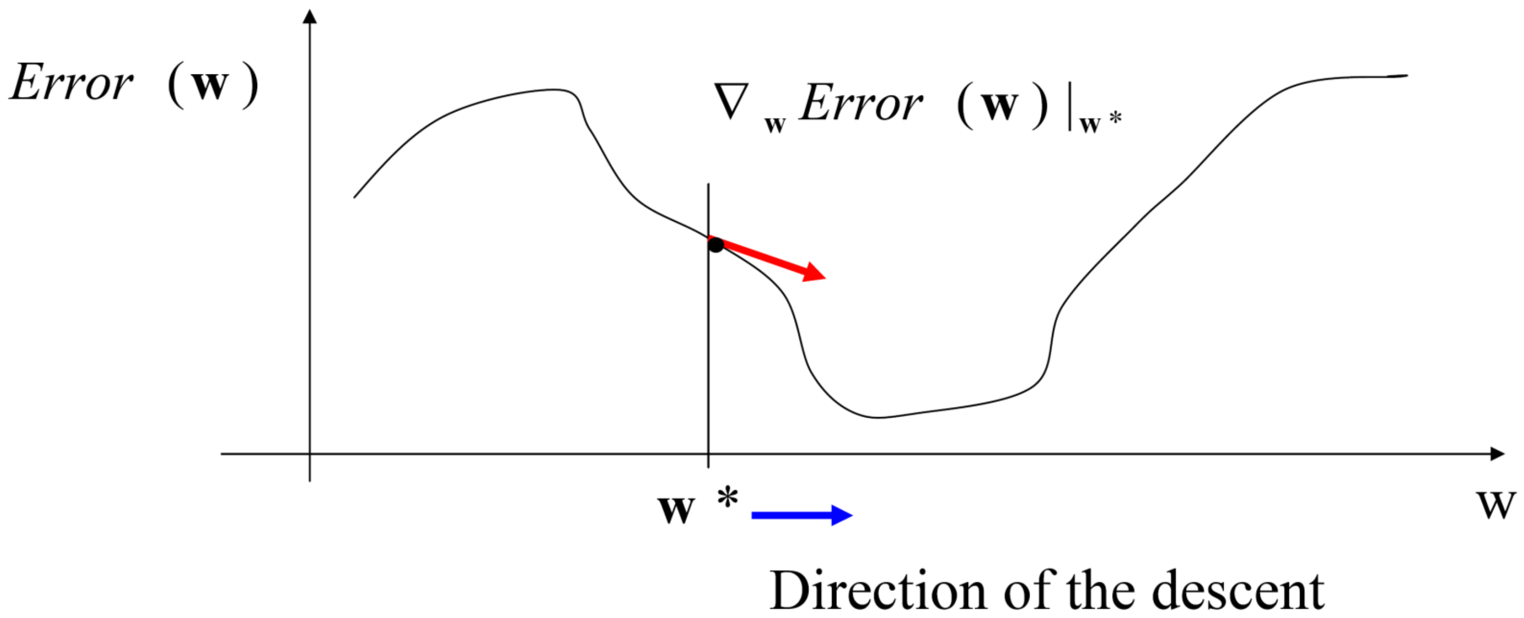
$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} Error_i(\mathbf{w})$$

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

Ref: Milos Hauskrecht

Gradient Descent Method

- Descend using the gradient information

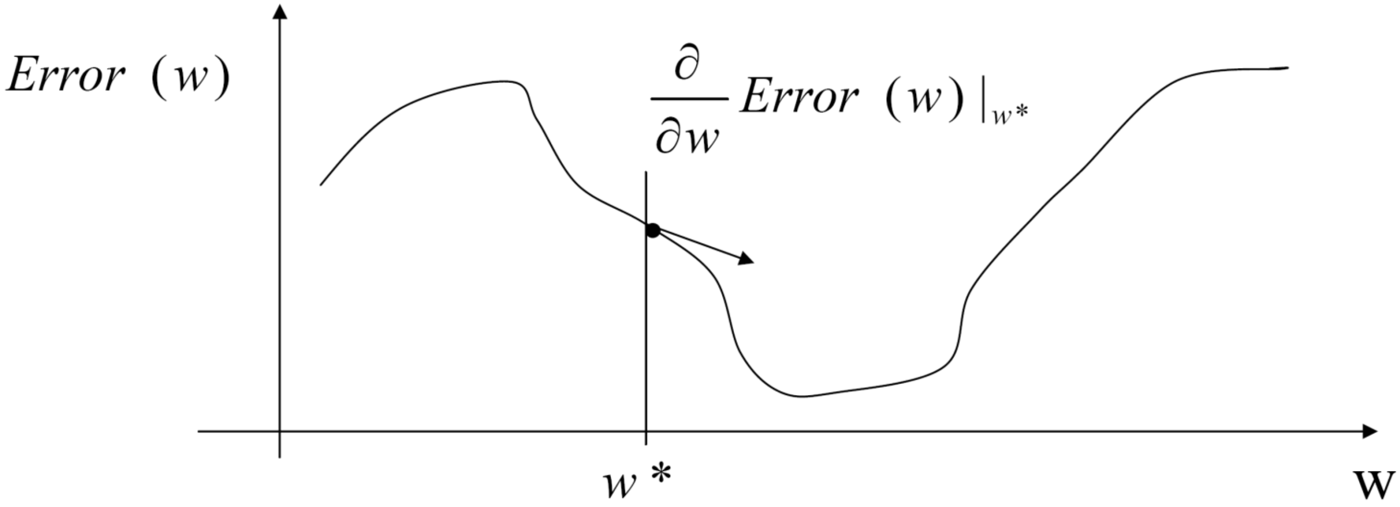


- Change the value of **w** according to the gradient

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

Ref: Milos Hauskrecht

Gradient Descent Method



- New value of the parameter

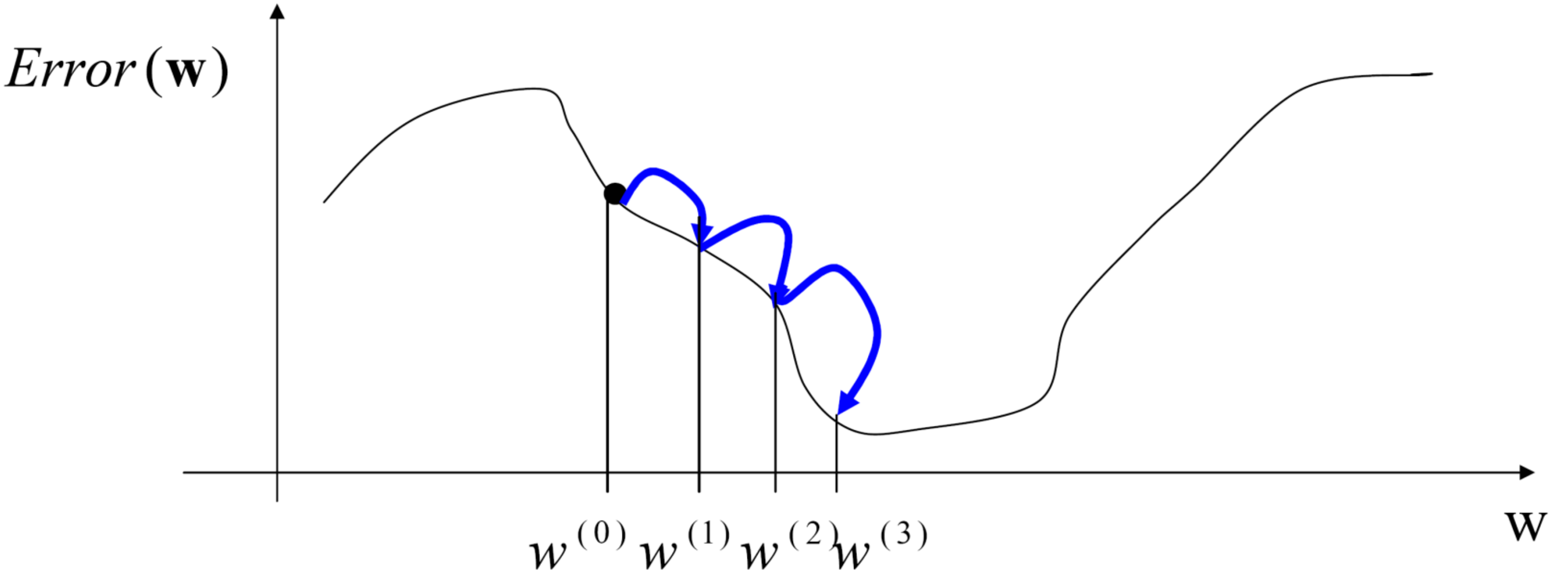
$$w_j \leftarrow w_j * -\alpha \frac{\partial}{\partial w_j} Error\ (w)|_{w*} \qquad \text{For all j}$$

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

Ref: Milos Hauskrecht

Gradient Descent Method

- Iteratively approaches the optimum of the Error function



Ref: Milos Hauskrecht

Online Gradient Algorithm

Linear model $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$

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

On-line algorithm: generates a sequence of online updates

(i)-th update step with : $D_i = < \mathbf{x}_i, y_i >$

j-th weight:

$$w_j^{(i)} \leftarrow w_j^{(i-1)} - \alpha(i) \frac{\partial Error_i(\mathbf{w})}{\partial w_j} \Big|_{\mathbf{w}^{(i-1)}}$$

$$\boxed{w_j^{(i)} \leftarrow w_j^{(i-1)} + \alpha(i)(y_i - f(\mathbf{x}_i, \mathbf{w}^{(i-1)}))x_{i,j}}$$

Fixed learning rate: $\alpha(i) = C$

- Use a small constant

Annealed learning rate: $\alpha(i) \approx \frac{1}{i}$

- Gradually rescales changes

Ref: Milos Hauskrecht

Online Regression Algorithm

Online-linear-regression (*D, number of iterations*)

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

for i=1:1: *number of iterations*

do **select** a data point $D_i = (\mathbf{x}_i, y_i)$ from D

set learning rate $\alpha(i)$

update weight vector

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

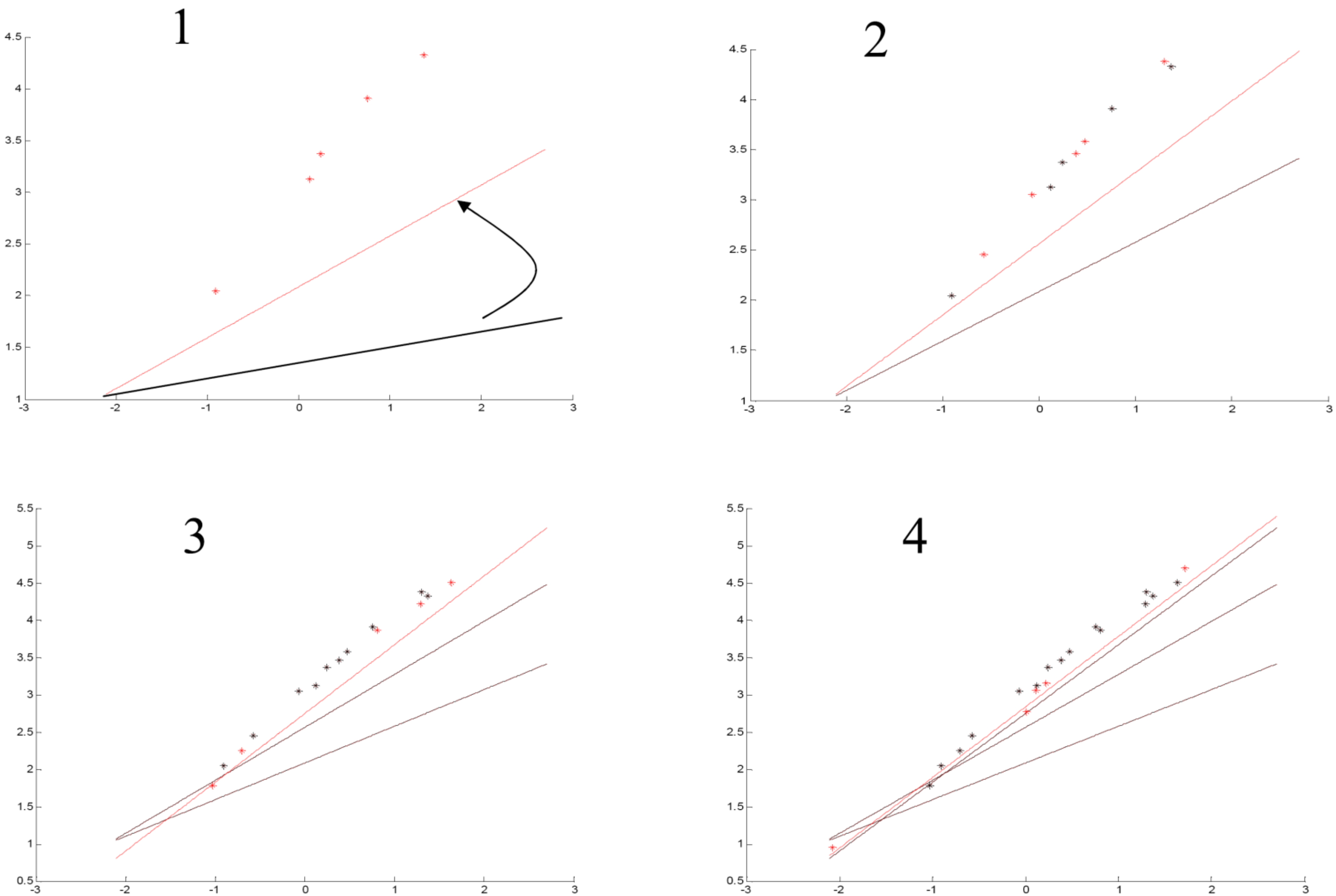
end for

return weights $\mathbf{w}$

- **Advantages:** very easy to implement, continuous data streams

Ref: Milos Hauskrecht

On-line Learning Example



Ref: Milos Hauskrecht

Linear Regression

Summary

$$\widehat{f}_n^L = \arg\min_{f \in \mathcal{F}_L} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$$

- is the orthogonal projection of Y onto the linear subspace spanned by the columns of A

Linear Regression - Example

Data pre-processing

Age

Sex

Data source

…

Using residual for analysis!

Ridge Regression

- Suppose we have an estimator $\hat{f}(\mathbf{z}) = \mathbf{z}^\top \hat{\beta}$
- To see if $\hat{f}(\mathbf{z}) = \mathbf{z}^\top \hat{\beta}$ is a good candidate, we can ask ourselves two questions:
  1.) Is $\hat{\beta}$ close to the true β?
  2.) Will $\hat{f}(\mathbf{z})$ fit future observations well?

Ridge Regression

- Just because $\hat{f}(\mathbf{z})$ fits our data well, this doesn't mean that it will be a good fit to new data
- In fact, suppose that we take new measurements y_i' at the same $\mathbf{z}_i$'s:

$$(\mathbf{z}_1, y_1'), (\mathbf{z}_2, y_2'), \ldots, (\mathbf{z}_n, y_n')$$

- So if $\hat{f}(\cdot)$ is a good model, then $\hat{f}(\mathbf{z}_i)$ should also be close to the new target y_i'

  This is the notion of **prediction error** (PE)

Ridge Regression

- If the β_j's are unconstrained...
  - They can explode
  - And hence are susceptible to very high variance
- To control variance, we might **regularize** the coefficients
  - E.g. control how large the coefficients grow

Ridge Regression

$$\text{minimize} \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^\top \mathbf{z}_i)^2 \text{ s.t. } \sum_{j=1}^{p} \beta_j^2 \leq t$$

$$\Leftrightarrow \quad \text{minimize } (y - \mathbf{Z}\boldsymbol{\beta})^\top (y - \mathbf{Z}\boldsymbol{\beta}) \text{ s.t. } \sum_{j=1}^{p} \beta_j^2 \leq t$$

By convention:

- Z is assumed to be standardized (mean 0, unit variance)
- y is assumed to be centered

Ridge Regression

- Can write the ridge constraint as the following **penalized** residual sum of squares (PRSS):

$$
\begin{aligned}
PRSS(\beta)_{\ell_2} &= \sum_{i=1}^{n}(y_i - \mathbf{z}_i^\top \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \\
&= (\mathbf{y} - \mathbf{Z}\beta)^\top (\mathbf{y} - \mathbf{Z}\beta) + \lambda ||\beta||_2^2
\end{aligned}
$$

- Its solution may have smaller average PE than $\hat{\beta}^{\text{ls}}$
- $PRSS(\beta)_{\ell_2}$ is convex, and hence has a unique solution
- Taking derivatives, we obtain:

$$
\frac{\partial PRSS(\beta)_{\ell_2}}{\partial \beta} = -2\mathbf{Z}^\top (y - \mathbf{Z}\beta) + 2\lambda\beta
$$

Ridge Regression

- The solution to $PRSS(\hat{\beta})_{\ell_2}$ is now seen to be:

$$\hat{\beta}_{\lambda}^{\text{ridge}} = (\mathbf{Z}^{\top}\mathbf{Z} + \lambda\mathbf{I}_p)^{-1}\mathbf{Z}^{\top}\mathbf{y}$$

- Solution is indexed by the tuning parameter λ (more on this later)

- So for each λ, we have a solution
  Hence, the λ's trace out a path of solutions

- λ is the shrinkage parameter

  λ controls the size of the coefficients
  λ controls amount of **regularization**
  As $\lambda \downarrow 0$, we obtain the least squares solutions
  As $\lambda \uparrow \infty$, we have $\hat{\beta}_{\lambda=\infty}^{\text{ridge}} = 0$ (intercept-only model)

Ridge Regression

- Need disciplined way of selecting λ:

  That is, we need to "tune" the value of λ
- In their original paper, Hoerl and Kennard introduced **ridge traces**:

  Plot the components of $\hat{\beta}_{\lambda}^{\text{ridge}}$ against λ
  Choose λ for which the coefficients are not rapidly changing and have "sensible" signs
  No objective basis; heavily criticized by many
- Standard practice now is to use cross-validation

Cross Validation

- We need a disciplined way of choosing λ
- Obviously want to choose λ that minimizes the mean squared error
- Issue is part of the bigger problem of **model selection**

Cross Validation

- If we have a good model, it should predict well when we have new data
- In machine learning terms, we compute our statistical model $\hat{f}(\cdot)$ from the **training set**
- A good estimator $\hat{f}(\cdot)$ should then perform well on a new, independent set of data
- We "test" or assess how well $\hat{f}(\cdot)$ performs on the new data, which we call the **test set**

Cross Validation

- Ideally, we would separate our available data into both training and test sets

In reality, it is very difficult!

- Hope to come up with the best-trained algorithm that will stand up to the test

Cross Validation

Most common approach is K-**fold cross validation**:

- Partition the training data T into K separate sets of equal size

  Suppose $T = (T_1, T_2, \ldots, T_K)$

  Commonly chosen K's are $K = 5$ and $K = 10$
- For each $k = 1, 2, \ldots, K$, fit the model $\hat{f}_{-k}^{(\lambda)}(\mathbf{z})$ to the training set excluding the kth-fold T_k
- Compute the fitted values for the observations in T_k, based on the training data that excluded this fold
- Compute the cross-validation (CV) error for the k-th fold:

$$(\text{CV Error})_k^{(\lambda)} = |T_k|^{-1} \sum_{(\mathbf{z},y)\in T_k} (y - \hat{f}_{-k}^{(\lambda)}(\mathbf{z}))^2$$

Cross Validation

- The model then has overall cross-validation error:

$$(\text{CV Error})^{(\lambda)} = K^{-1} \sum_{k=1}^{K} (\text{CV Error})_k^{(\lambda)}$$

- Select λ^* as the one with minimum $(\text{CV Error})^{(\lambda)}$

Cross Validation

What happens when $\boxed{K = 1}$?

This is called **leave-one-out cross validation**

For squared error loss, there is a convenient approximation to CV(1), which is the leave one-out CV error

Linear Regression Summary

$$\min_{\beta}(\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y}) + \lambda \mathrm{pen}(\beta) = \min_{\beta} J(\beta) + \lambda \mathrm{pen}(\beta)$$

Least Square Solution

$$\mathrm{pen}(\beta) = \|\beta\|_2^2$$

Ridge Regression

$$\mathrm{pen}(\beta) = \|\beta\|_1$$

Lasso Regression

Lasso (L1 penalty) results in sparse solutions –vector with more zero coordinates. Will come to Lasso later!