FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

NATURWISSENSCHAFTLICHE
FAKULTÄT

Linear Regression

Lecture “Mathematical Data Science” 2021/2022

Frauke Liers
Friedrich-Alexander-Universität Erlangen-Nürnberg
16.11.2021

DISCRETE
OPTIMIZATION

Linear Regression

Today, we will see two methods for prediction:

- a linear model (linear regression)
- nearest neighbors

- regression: targets are numeric values (quantitative)
- classification problem: targets are categorial (discrete, e.g., a color, an object, etc.). Can also be *ordered categorial*, e.g., small, medium, large. Typically represented by discrete numbers 0,1,...
- Recall: polynomial function

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

M order ot polynomial, real coefficients w_j is only *linear* in unknown coefficients w.
monomials x, x^2, x^3, are also called *basis functions*.

Linear Models and Least Squares

(see book Hastie et al. Elements of Statistical Learning, chapters 2,3.)
Let input variable be denoted by X (can also be a vector of appropriate dimension), and Y (quantitative) output, i.e., target.
Let $X^T = (X_1, \dots, X_M)$ M-dimensional vector of inputs predict the output (target) Y (in dimension K)

$$(Y =) f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

β_0: intercept, *bias* in machine learning
can also model polynomials via $X_2 = X_1^2, X_3 = X_1^3$, etc.
In order to write linear model compactly: include constant variable 1 in X, include β_0 in vector of coefficients β , and write linear model in vector form

$$Y = X^\top \beta$$

In M-dimensional input-output space, (X, Y) is a hyperplane. As β_0 is included in X, the hyperplane includes the origin and is a subspace.

Linear Models and Least Squares

Typical fit of linear model to a set of training data via *method of least squares* (cmp last week): Determine β that minimizes residual sum of squares RSS

$$RSS(\beta) = \sum_{i=1}^{N}(y_i - f(x_i))^2 = \sum_{i=1}^{N}(y_i - x_i^\top \beta)^2$$

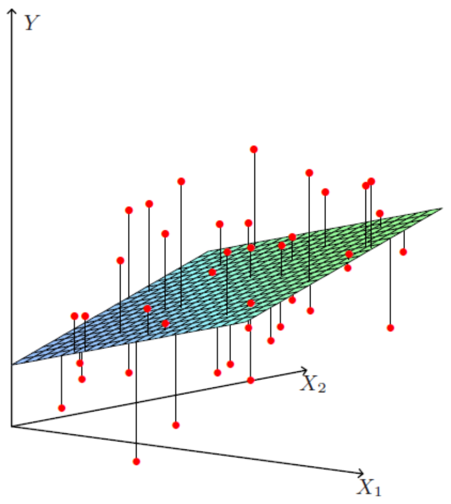

FIGURE 3.1. *Linear least squares fitting with $X \in \mathbb{R}^2$. We seek the linear function of X that minimizes the sum of squared residuals from Y.*

Linear Models and Least Squares

L o o k f o r t h e d e r i v a t i o n .

P e r f o r m i n N o t e b o o k .

$RSS(\beta)$ is a quadratic function in $M + 1$ parameters that can be written in matrix form as

$$RSS(\beta) = (y - \mathbf{X}\beta)^\top (y - \mathbf{X}\beta).$$

Taking partial derivative w.r.t. β yields: $\frac{\partial RSS}{\partial \beta} = -2\mathbf{X}^\top (y - \mathbf{X}\beta)$
We calculate the Hessian: $\frac{\partial^2 RSS}{\partial\beta\partial\beta^\top} = 2\mathbf{X}^\top\mathbf{X}$. We assume for the moment that $\mathbf{X}$ has full column rank. Hence, $\mathbf{X}^\top X$ is positive definite.
We determine critical points: $\mathbf{X}^\top(y - \mathbf{X}\beta) = 0$

- From inserting the data in matrix form $\mathbf{X}$, we obtain unique solution $\hat{\beta} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top y$.
- Fitted surface $f(X) = X^\top\hat{\beta}$ is fully characterized by $\hat{\beta}$.
- If new data point x comes in, target / output Y is predicted via $x^\top\hat{\beta}$.

Linear Models and Least Squares

suppose data is such that columns of $\mathbf{X}$ are not linearly independent and so $\mathbf{X}$ does not have full rank, e.g., because of correlations in input data. Non-full-rank occurs often due to redundant coding in redundant fashion.
Usually: preprocessing such that redundant columns in $\mathbf{X}$ are deleted. I.e., we can assume that full rank is given.
The method can easily be generalized to multidimensional target (output) vectors.

Example Linear Regression

training data on inputs (X_1, X_2): target G takes either value $Y = 0$ for value blue or $Y = 1$ for value orange. Targets denoted by $\hat{Y}$.

$$G = \begin{cases} \text{orange} & \text{, if } Y > 0.5 \\ \text{blue} & \text{, if } Y \leq 0.5 \end{cases}$$

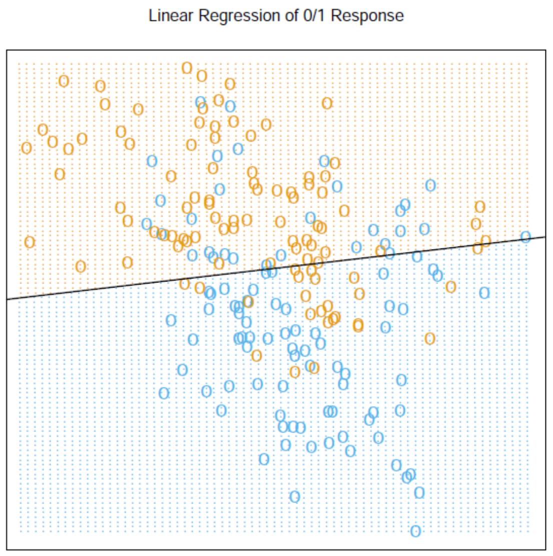

FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (BLUE $= 0$, ORANGE $= 1$), and then fit by linear regression. The line is the decision boundary defined by $x^T\hat{\beta} = 0.5$. The orange shaded region denotes that part of input space classified as ORANGE, while the blue region is classified as BLUE.*

Example Linear Regression

Read the Screnarios in which linear model will work.

- set of points in $\mathbb{R}^2$ with $\{x \mid x^\top \hat{\beta} > 0.5\}$ are classified as orange
- $\{x \mid x^\top \hat{\beta} = 0.5\}$ is (here linear) decision boundary

We see several misclassifications on each side.

Can this be avoided?
Assume data was generated by one of the two scenarios:

- scenario 1: in each class, training data was generated from Gaussian distribution, uncorrelated components, different means
- scenario 2: data comes from mixture of 10 low-variance Gaussians, means themselves distributed as Gaussian.

in case of scenario 1: linear decision boundary is best we can do, estimate is almost optimal (see later...), region of overlap is inevitable.
in case of scenario 2: linear boundary is unlikely to be optimal, should be nonlinear & disjoint and more difficult.
nearest-neighbor method is better suited for scenario 2, see next.

Nearest-Neighbor Methods

We use a similar idea as in unsupervised learning for k-means clustering.
Here: For estimating the targets, use observations in training set that are 'closest' in input space, measured in some metric, e.g., in Euclidean distance.
Let $N_k(x)$ neighborhood of x defined by k closest points x_i in training set.
k-nearest neighbor fit to estimate target $\hat{Y}$:

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

$$G = \begin{cases} \text{orange} & , \textit{if } \hat{Y} > 0.5 \\ \text{blue} & , \textit{if } \hat{Y} \leq 0.5 \end{cases}$$

Nearest-Neighbor Methods

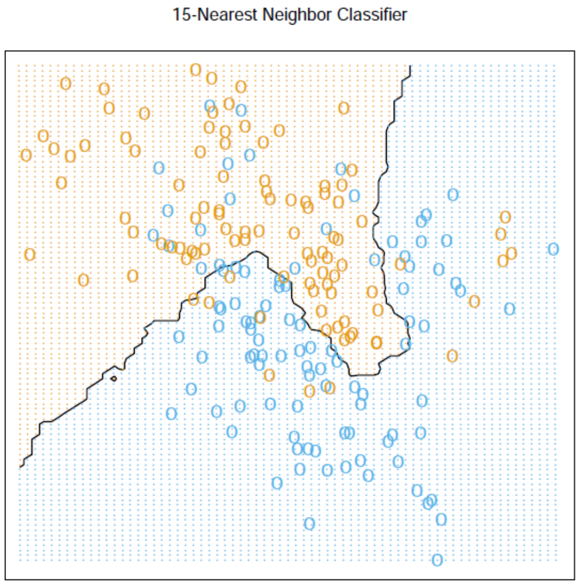

FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

15-nearest-neighbor averaging: irregular boundary

Nearest-Neighbor Methods

however: for $k = 1$, error (i.e., sum of Euclidean distances) will always be zero,

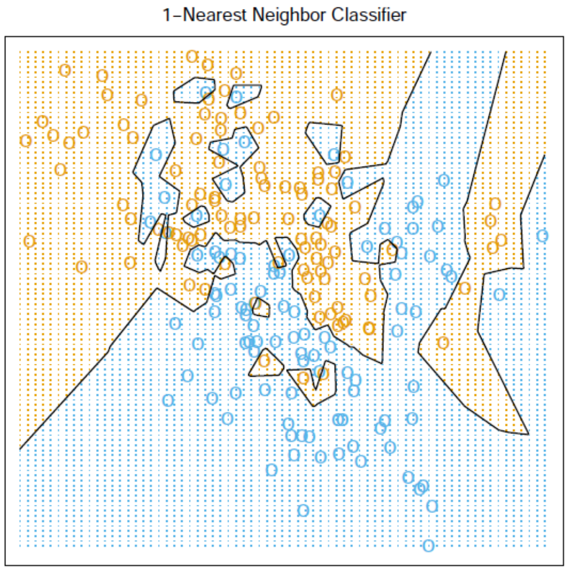

FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by 1-nearest-neighbor classification.*

Look out the notes from Applied AI, like for k=1, k=2 what will happen.

Smaller the K=> higher the overfitting.

see figure.
error will increase with value of k.

Least Squares and Nearest Neighbors

least squares: linear decision boundary is smooth and stable, relies on assumption that linear boundary is appropriate
nearest neighbors: does not assume anything on underlying data, can adapt. however, is unstable
each method has its situations in which it works best. linear regression for data resemble scenario 1, nearest neighbor scenario 2.

More Explanations via Statistical Decision Theory

Assume quantiative output. Let $X \in \mathbb{R}^p$ be random input vector $Y \in \mathbb{R}$ random output variable, joint distribution $Pr(X, Y)$ (i.e. distribution of observing X together with Y)
Seek function $f(X)$ for predicting Y given values of input X.
loss function $L(Y, f(X))$ penalizes prediction errors. Often used: squared error loss

$$L(Y, f(X))(Y - f(X))^2$$

.
expected squared prediction error

$$EPE(f) = E(Y - f(X))^2 = \int [y - f(x)]^2 Pr(dx, dy)$$

Write joint density as $Pr(X, Y) = Pr(Y|X)Pr(X)$ (see Bayes' Theorem).
Thus, write EPE as

$$EPE(f) = E_X E_{Y|X}([Y - f(X)]^2|X)$$

More Explanations via Statistical Decision Theory

It suffices to minimize EPE pointwise:

$$f(x) = \operatorname*{argmin}_c E_{Y|X}([Y - c]^2) \mid X = x).$$

Solution is

$$f(x) = E(Y \mid X = x)$$

Thus, best prediction of Y at any point $X = x$ is conditional mean, when measured by average squared error.
nearest-neighbor method attempts to implement this formula on the training data:
Conditional mean, i.e., at each point x we ask for average of all those y_i with input $x_i = x$. Typically there is at most one observation at any point x.

More Explanations via Statistical Decision Theory

Denote by *Ave* the average. Thus, for

$$\hat{f}(x) = Ave(y_i \mid x_i \in N_k(x). \tag{1}$$

we have performed the following approximations:

- expectation is done by averaging over training data
- conditioning is relaxed to conditioning on some 'neighbor' region 'close' to target

How good is this?
for large data sets N, points in neighborhood tend to be close to x. As k gets large, average will get more stable
Do we need anything more? Yes, N is not often large. Thus, if the linear model is appropriate, this is usually more stable than k-nearest neighbors. For large M, running timea tend to get large for larger values of k.

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
NATURWISSENSCHAFTLICHE FAKULTÄT

More information

linear regression: assumption that regression function $f(x)$ is approximately linear in its arguments $f(x) \approx x^\top \beta$.
Insert $f(x)$ into

$$EPE(f) = E_X E_{Y|X}([Y - f(X)]^2|X),$$

determine critical points, finds

$$\beta = [E(XX^\top)]^-1 E(XY).$$

Compare with least-square solution that we have calculated earlier from data $\hat{\beta} = (XX^\top)^-1 X^\top y$:
We replace the expectation by averages over training data
k-nearest neighbors and least squares both replace conditional expectations by averages over data.
differences:

- least squares assumes $f(x)$ is well approximated by globally linear function (model-based approach)
- nearest neighbor assumes $f(x)$ is approximately locally constant, see (1).

More generally: Linear Regression as Function Approximation

determine useful approximation of a function $f(x)$ for all x in some region of $\mathbb{R}^M$
until now: linear basis functions. Now slight generalization via linear basis expansions

$$f_\theta(x) = \sum_{k=1}^{K} h_k(x)\theta_k$$

with suitable functions h_k, e.g., polynomial, trigonometric: $x_1, x_1 x_2^2, \cos(x_1)\dots$
another example: sigmoid transformation (see neural networks later)
$h_k(x) = \frac{1}{1+\exp(-x^\top \beta_k)}$
Similarly, calculate residual sum-of-squares

$$RSS(\theta) = \sum_{i=1}^{N} (y_i - f_\theta(x_i))^2.$$

An alternative: maximum likelihood estimation: suppose have random sample $y_i, i = 1, \dots, N$ from density $Pr_\theta(y)$ indexed by θ.

More information

Log-probability of observed sample is

R e a d   a b o u t   M a x i m u m
L i k e l i h o o d   e s t i m a t i o n .

$$L(\theta) = \sum_{i=1}^{N} \log Pr_\theta(y_i)$$

principle of maximum likelihood: assume that most reasonable values for θ are those for which probability of observed sample is largest.
Least squares for error model $Y = f_\theta(X) + \epsilon$, with Gaussian distributed error $\epsilon \sim N(0, \sigma^2)$, is equivalent to maximum likelihood

$$Pr(Y \mid X, \theta) = N(f_\theta(X), \sigma^2)$$

insert in loss function:

$$L(\theta) = -\frac{N}{2}\log(2\pi) - N\log\sigma - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - f_\theta(x_i))^2$$

only term involving θ is the last one which is $RSS(\theta)$. Thus, linear regression is similar...!