

Linear models: regression and logistic regression

ENSE3 / Grenoble-INP

Parcours Numérique 1A

Florent Chatelain* Olivier Michel*

* GIPSA-lab, Univ. Grenoble Alpes,

2019-2020

Linear model: Keep it simple!

Simple linear approach may seem overly simplistic

- true prediction functions are never linear
- + extremely useful, both conceptually and practically

Practically

Gorge Box, 60': "Essentially, all models are wrong, but some are very useful"

- 👉 *Simple is actually very good*: works very well in a lot of situations by capturing the main effects (which are generally the most interesting)

Conceptually

Many concepts developed for the linear problem are important for a lot of the supervised learning techniques

- 👉 Although it is never correct, a linear model serves as a good and interpretable approximation of the unknown true function $f(X)$

Model based approaches

Reminder on Supervised Learning

- ▶ input data $X \in \mathbb{R}^p$
- ▶ response Y to be predicted
- ▶ training set $(X_1, Y_1), \dots, (X_n, Y_n)$

In a model based approach, we seek an explicit relation between the (input) data X and the response Y . We focus here on *Discriminative models*, where we just model explicitly the conditional distribution $P(Y|X)$ rather than the joint distribution $P(X, Y)$.

Discriminative models

Direct learning of $P(Y|X)$, e.g.

- ▶ Linear regression
- ▶ Logistic regression (\leftarrow generalized linear model for **classification** tasks)
- ▶ ...

Outline

Linear regression

Stochastic Gradient Descent

Logistic regression

Linear Regression Problem

- ▶ $X_i = (X_{i,1}, \dots, X_{i,p})^T \in \mathbb{R}^p$,
 - ▶ $Y_i \in \mathbb{R}$,
- for $i = 1, \dots, n$ (sized n training set)

Linear Regression Model

$$Y_i = \beta_0 + \sum_{j=1}^p \beta_j X_{i,j} + \sigma \epsilon_i, \quad \text{for } i = 1, \dots, n,$$

- ▶ ϵ_i is a centered with unit variance ($E[\epsilon_i] = 0$, $\text{var}(\epsilon_i) = 1$) white noise
- ▶ β_0 is the “intercept” (reduces to the ordinate at the origin when $p = 1$)
- ▶ $\beta \equiv (\beta_0, \dots, \beta_p) \in \mathbb{R}^{p+1}$ is the **coefficient vector**

Objective: estimation of β using the samples in the training set \leftarrow supervised learning problem

Linear Regression Problem (Cont'd)

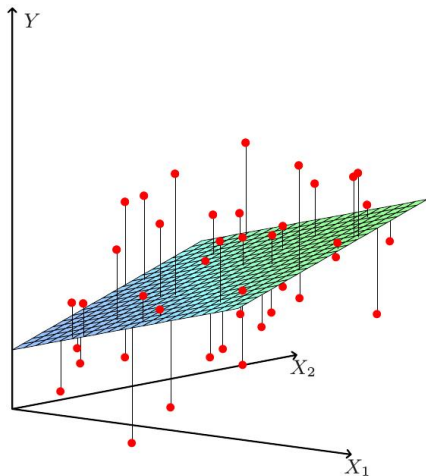
Linear Regression Model

$$Y_i = \beta_0 + \sum_{j=1}^p \beta_j X_{i,j} + \sigma \epsilon_i, \quad \text{for } i = 1, \dots, n,$$

Remark: model linear w.r.t. $\beta \equiv (\beta_0, \dots, \beta_p) \in \mathbb{R}^{p+1}$, but not necessarily linear w.r.t.

- ▶ the inputs X_i : we can add non linear predictors $h(X_1, \dots, X_p)$ in the model, e.g. X_i^2 , $X_i X_j \dots$
- ▶ the outputs Y_i : we can introduce a non linear link function \leftarrow generalized linear model, e.g. logistic regression

Least Squares (LS) Estimator



Linear least squares fitting with $X \in \mathbb{R}^2$

LS estimate defined by minimizing the **Residual Sum of Squares (RSS)**

$$\hat{\beta} = \arg \min_{\beta} \underbrace{\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{i,j} \right)^2}_{\text{RSS}(\beta)}$$

► $\text{RSS}(\beta) \propto$ training error rate for quadratic loss

Least Squares Estimator (Cont'd)

$$\hat{\beta} = \arg \min_{\beta} \text{RSS}(\beta), \quad \text{where } \text{RSS}(\beta) = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{i,j} \right)^2$$

Matrix expression of RSS

$$\text{RSS}(\beta) = \|Y - X\beta\|_2^2,$$

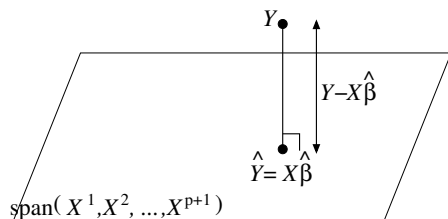
$$\text{where } Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^n, \quad X = \begin{pmatrix} 1 & x_{1,1} & \dots & x_{1,p} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n,1} & \dots & x_{n,p} \end{pmatrix} \in \mathbb{R}^{n \times (p+1)}$$

LS Estimator derivation

$\hat{Y} = X\hat{\beta}$ is the prediction in the space spanned by the column vectors of X such that the euclidean error norm $\|Y - X\hat{\beta}\|_2$ is minimized

Orthogonality principle

Let X^j be the j th column of X



for $j = 1, \dots, p + 1$

$$\begin{aligned} \langle X^j, Y - X\hat{\beta} \rangle &= (X^j)^T (Y - X\hat{\beta}) = 0, \\ \Leftrightarrow X^T (Y - X\hat{\beta}) &= 0, \\ \Leftrightarrow (X^T X) \hat{\beta} &= X^T Y \end{aligned}$$

Rk: This condition can also be derived by setting the gradient of $\text{RSS}(\beta) = (Y - X\beta)^T (Y - X\beta)$ to 0.

LS Estimator computation

Assumption: $\text{rank } X = p + 1$, hence $X^T X$ is invertible

Analytical expression

Because $(X^T X) \hat{\beta} = X^T Y$,

$$\hat{\beta} = (X^T X)^{-1} X^T Y,$$

Notebook `N1_polynomial_regression.ipynb`

Numerical computation in high dimension

Pb: When $p > 10^3$ or $p > 10^4$, too expensive to compute $(X^T X)^{-1} \dots$

👉 more efficient to use a numerical procedure to minimize the RSS, e.g. [steepest descent](#)

Outline

Linear regression

Stochastic Gradient Descent

Logistic regression

Steepest descent, aka gradient descent

We can define the criterion to be minimized as $J(\beta) \equiv \frac{1}{2}\text{RSS}(\beta)$

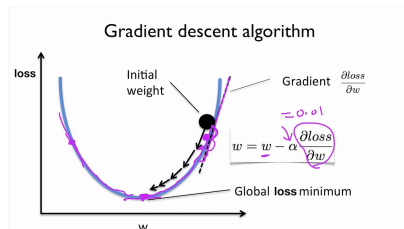
Steepest descent

Ubiquitous iterative procedure based on the observation that $J(\mathbf{w})$ decreases fastest if one goes from \mathbf{w} in the direction of the negative gradient of $J(\cdot)$ at \mathbf{w} , i.e. $-\nabla J(\mathbf{w})$. Here one iteration consists in

$$\beta_{k+1} = \beta_k - \alpha_k \nabla_{\beta} J(\beta_k),$$




where

- ▶ $\alpha_k \in \mathbb{R}$ is the **learning rate**
- ▶ $\nabla_{\beta} J(\beta) = X^T X \beta - X^T Y$ is the **batch gradient** (computed over the whole training set)



Steepest descent, aka gradient descent

Supplementary materials

-  Coursera MOOC short (11mn) and educational video <https://www.coursera.org/lecture/machine-learning/gradient-descent-intuition-GFFPB>
-  Wikipedia page https://en.wikipedia.org/wiki/Gradient_descent
-  Scikit-learn documentation (with description of learning rate strategies)
<https://scikit-learn.org/stable/modules/sgd.html#mathematical-formulation>

Stochastic Gradient Descent (SGD)

Remember: batch gradient $\nabla_{\beta} J(\beta) = X^T X \beta - X^T Y$

Pb: For large and high-dimensional datasets, still too expensive to compute the batch gradient (requires to store and compute $X^T X \dots$)

- 👉 stochastic approximation of the batch gradient to decrease the computational burden

Stochastic gradient

Descent direction is computed as $\nabla_{\beta} J(\beta) \approx X_i^T X_i \beta - X_i^T Y_i$, for a given sample $i \in \{1, \dots, n\}$ in the training set, where X_i is the i th *line vector* of X .

- ▶ cheaper than batch one for a single iteration, can be much more efficient
- ▶ one loop over all the $i = 1, \dots, n$ training samples is called an **epoch**

Notebook `N2_learning_rate_SGD.ipynb`

Mini-batch SGD

Tradeoff between batch and stochastic gradients:

- ▶ gradient computed on a small subset (**mini-batch**) of the training set,
- ▶ one loop over all the mini-batches (thus over the whole training set) is an **epoch**
- ▶ one epoch is one iteration of the gradient procedure, which is repeated many times to achieve a good minimization

Properties

- ▶ smoother convergence than pure SGD
- ▶ more computationally efficient than batch gradient
- ▶ size b of the mini-batch drives the trade-off ($b = 1$ is pure SGD, $b = n$ is batch gradient). Basically $b = 32, 64$ or 128 .
- 👉 standard optimization procedure for many ML methods (e.g. deep neural nets)

Outline

Linear regression

Stochastic Gradient Descent

Logistic regression

Discriminative model for classification: $Y \in \mathcal{Y} \leftarrow$ discrete set

Discriminative model

For a given $X = x$, we want to model directly

$$\Pr(Y = k | X = x)$$

for each value of the class label $k \in \mathcal{Y}$

- ▶ do not require to specify the marginal distribution of the inputs X

Model-based classification rule

We predict the class with the **highest probability**

$$\hat{Y} = \arg \max_k \Pr(Y = k | X = x)$$

- ▶ this is the optimal rule for misclassification rate referred to as *Bayes Classifier*... if the model is true (of course this is not the case, but it may be useful)!

How can we use linear regression to model a probability $\Pr(Y = k | X = x)$?

Linear model for classification: Logistic regression (LR)

Classification problem $Y \in \mathcal{Y} \leftarrow$ discrete set

Binary classification problem: $\mathcal{Y} = \{1, 2\}$

Consider the following model

$$\Pr(Y_i = 1 | X_i = x_i) = \phi(x_i^T \beta) = \frac{\exp(x_i^T \beta)}{1 + \exp(x_i^T \beta)},$$

where

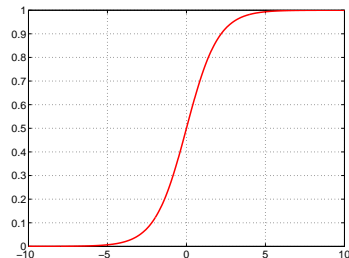
- ▶ $x_i = (\mathbf{1}, x_{i,1}, \dots, x_{i,p})^T \in \mathbb{R}^{p+1} \leftarrow$ **intercept** term included by default,
- ▶ ϕ is the **logistic** function: maps a real value to a probability

Multiclass problem: $\mathcal{Y} = \{1, 2, \dots, K\}$

logistic model can be easily extended to the multiclass problem: **multinomial** logistic regression

$$\phi : \mathbb{R} \rightarrow (0, 1)$$

$$u \mapsto \frac{\exp u}{1 + \exp u} = \frac{1}{1 + \exp(-u)}.$$



LR is a generalized linear model

Consider

- ▶ $p_i \equiv \Pr(Y_i = 1|X_i = x_i) = \phi(x_i^T \beta)$
- ▶ $\phi^{-1} : p \in (0, 1) \mapsto \log \frac{p}{1-p} \in \mathbb{R}$ is the **logit** function

Generalized linear model

- ▶ **Linear** equation w.r.t. β : $\text{logit}(p_i) = x_i^T \beta$,
- ▶ + additional nonlinear constraint (proba sum to 1): $\Pr(Y_i = 2|X_i = x_i) = 1 - p_i = \frac{1}{1 + \exp(x_i^T \beta)}$

Maximum Likelihood Estimates

$$\hat{\beta} = \arg \min_{\beta} -\ell(\beta)$$

where $\ell(\beta)$ is the logistic log-likelihood (normal model yields standard LSE for linear regression)

- ▶ no analytical expression of $\hat{\beta}$ which is computed by an optimization procedure, e.g. (Stochastic) Gradient Descent

Conclusions

Generalized Linear Models

Learning of the prediction rule based on a model of Y given X

- 👉 Linear regression, Logistic regression

Properties

- ▶ Simplicity: useful to capture the main effects
- ▶ Interpretability
- ▶ Efficient numerical procedures for large or high-dimensional data

Perspectives

- ▶ Regularization: Shrinkage and Selection procedures