Introduction to ML and Regression

Statistical Natural Language Processing

Çağrı Çöltekin

University of Tübingen
Seminar für Sprachwissenschaft

Summer Semester 2021

Why machine learning?

- Majority of the modern computational linguistic tasks and applications are based on machine learning
  - Tokenization
  - Part of speech tagging
  - Parsing
  - ...
  - Speech recognition
  - Named Entity recognition
  - Document classification
  - Question answering
  - Machine translation
  - ...

Machine learning is ...

The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience. —Mitchell (1997)

Machine Learning is the study of data-driven methods capable of mimicking, understanding and aiding human and biological information processing tasks. —Barber (2012)

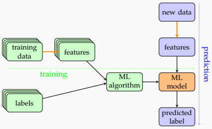
Statistical learning refers to a vast set of tools for understanding data. —James et al. (2013)

Supervised or unsupervised

- Machine learning methods are often divided into two broad categories: *supervised* and *unsupervised*
- Supervised methods rely on *labeled* (or *annotated*) data
- Unsupervised methods try to find regularities in the data without any (direct) supervision
- Some methods do not fit any (or fit both):
  - *Semi-supervised* methods use a mixture of both
  - *Reinforcement learning* refers to the methods where supervision is indirect and/or delayed

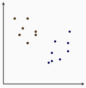
In this course, we will mostly discuss/use supervised methods.

Supervised learning



Unsupervised learning

- In unsupervised learning we do not have any labels
- The aim is discovering some 'latent' structure in the data
- Common examples include
  - Clustering
  - Density estimation
  - Dimensionality reduction
- The methods that do not require (manual) annotation are sometimes called unsupervised



Supervised learning

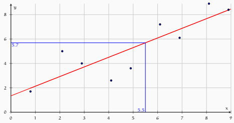
two common settings

A supervised ML method is called

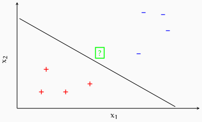
regression if the outcome to be predicted is a numeric (continuous) variable

classification if the outcome to be predicted is a categorical variable

Regression



Classification



ML topics we will cover in this course

- (Linear) Regression (today)
- Classification (perceptron, logistic regression, ANNs)
- Evaluating ML methods / algorithms
- Unsupervised learning
- Sequence learning

Machine learning and statistics

- The methods largely overlap (it was even suggested that both should be collectively called 'data science')
- Aims differ
  - In statistics (used as in experimental sciences) aim is making inferences using the models
  - In machine learning correct predictions are at the focus
- A more diverse set of models/methods are used in ML

Machine learning and models

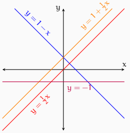
- A machine learning method makes its predictions based on a model
- The models are often parametrized: a set of parameters defines a model
- Learning can be viewed as finding the 'best' model among a family of models (that differ based on their parameters)

The linear equation: the regression model

$$y = a + bx$$

a (intercept) is where the line crosses the y axis.

b (slope) is the change in y as x is increased one unit.



The simple linear model

some terminology

$$y_i = a + bx_i$$

y is the *outcome* (or response, or dependent) variable. The index i represents each unit observation/measurement (sometimes called a 'case')

x is the *predictor* (or explanatory, or independent) variable

a is the *intercept* (called *bias* in the NN literature)

b is the *slope* of the regression line.

a and b are called *coefficients* or *parameters*

$a + bx$ is the model's prediction of y ($\hat{y}$), given x

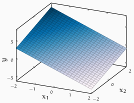
Notation differences for the regression equation

$$y_i = \boldsymbol{w}\boldsymbol{x}_i$$

- Sometimes, Greek letters α and β are used for intercept and the slope, respectively
- Another common notation to use only b, β, θ or w, but use subscripts, 0 indicating the intercept and 1 indicating the slope
- In machine learning it is common to use w for all coefficients (sometimes you may see b used instead of w_0)
- Sometimes coefficients wear hats, to emphasize that they are estimates
- Often, we use the vector notation for both input(s) and coefficients: $\boldsymbol{w} = (w_0, w_1)$ and $\boldsymbol{x}_i = (1, x_i)$

Regression models with multiple predictors

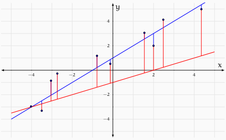
- The equation defines a (hyper)plane
- With 2 predictors: $y = w_0 + w_1x_1 + w_2x_2$
- With more predictors it is more convenient to use the vector notation: $y = \boldsymbol{w}\boldsymbol{x}$



Parameter estimation

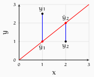
- In ML, we are interested in finding the best model based on data
- Learning is selecting a model from a family of models that differ in their parameters
- Typically, we seek the parameters that reduce the prediction error on a training set
- Ultimately, we seek models that do not only do well on the training data, but also new, unseen instances

Parameter estimation for regression



Estimating regression parameters

- We view learning as a search for the regression equation with least error
- The error terms are also called *residuals*
- We want error to be low for the whole training set: average (or sum) of the error has to be reduced
- Can we minimize the sum of the errors?



$$y_i = \underbrace{w_0 + w_1x_i}_{\hat{y}_i} + \epsilon_i$$

Least-squares regression

- Find w_0 and w_1, that minimize the *sum of the squared errors* (SSE)

$$E(\boldsymbol{w}) = \sum_i \epsilon_i^2 = \sum_i (y_i - \hat{y}_i)^2 = \sum_i (y_i - (w_0 + w_1x_i))^2$$

- We can minimize $E(\boldsymbol{w})$ analytically

$$w_1 = \frac{\sigma_{xy}}{\sigma_x^2} = r\frac{sd_y}{sd_x} \qquad w_0 = \bar{y} - w_1\bar{x}$$

Short digression: minimizing functions

In least squares regression, we want to find w_0 and w_1 values that minimize

$$E(\boldsymbol{w}) = \sum_i (y_i - (w_0 + w_1x_i))^2$$

- Note that $E(\boldsymbol{w})$ is a *quadratic* function of $\boldsymbol{w} = (w_0, w_1)$
- As a result, $E(\boldsymbol{w})$ is *convex* and have a single extreme value
  - there is a unique solution for our minimization problem
- In case of least squares regression, there is an analytic solution
- Even if we do not have an analytic solution, if the error function is convex, a search procedure like *gradient descent* can still find the *global minimum*

What is special about least-squares?

- Minimizing MSE (or SS_R) is equivalent to MLE estimate under the assumption $\epsilon \sim \mathcal{N}(0, \sigma^2)$
- Working with 'minus log likelihood' is more convenient

$$E(w) = -\log \mathcal{L}(\boldsymbol{w}) = -\log \prod_i \frac{e^{-\frac{(y_i - \hat{y}_i)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$$

$$\hat{\boldsymbol{w}} = \arg\min_w (-\log \mathcal{L}(\boldsymbol{w})) = \arg\min_w \sum_i (y_i - \hat{y}_i)^2$$

- There are other error functions, e.g., absolute value of the errors, that can be used (and used in practice)
- One can also estimate regression parameters using Bayesian estimation

Regression with multiple predictors

$$y_i = \underbrace{w_0 + w_1x_{i,1} + w_2x_{i,2} + \ldots + w_kx_{i,k}}_{\hat{y}} + \epsilon_i = \boldsymbol{w}\boldsymbol{x}_i + \epsilon_i$$

w_0 is the intercept (as before).

$w_{1..k}$ are the coefficients of the respective predictors.

ϵ is the error term (residual).

- using the vector notation the equation becomes:

$$y_i = \boldsymbol{w}\boldsymbol{x}_i + \epsilon_i$$

where $\boldsymbol{w} = (w_0, w_1, \ldots, w_k)$ and $\boldsymbol{x}_i = (1, x_{i,1}, \ldots, x_{i,k})$

It is a generalization of simple regression with some additional power and complexity.

Evaluating machine learning systems

- Any (machine learning) system needs a way to measure its success
- For measuring success (or failure) in a machine learning system we need quantitative measures
- Remember that we need to measure the success outside the training data

Measuring success in Regression

- *Root-mean-square error* (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2}$$

measures average error in the units compatible with the outcome variable.

- Another well-known measure is the *coefficient of determination*

$$R^2 = \frac{\sum_i^n(\hat{y}_i - \bar{y})^2}{\sum_i^n(y_i - \bar{y})^2} = 1 - \left(\frac{\text{RMSE}}{\sigma_y}\right)^2$$

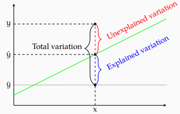
Assessing the model fit: R^2

We can express the variation explained by a regression model as:

$$\frac{\text{Explained variation}}{\text{Total variation}} = \frac{\sum_i^n(\hat{y}_i - \bar{y})^2}{\sum_i^n(y_i - \bar{y})^2}$$

- In simple regression, it is the square of the correlation coefficient between the outcome and the predictor
- The range of R^2 is $[0, 1]$
- $100 \times R^2$ is interpreted as 'the percentage of variance explained by the model'
- R^2 shows how well the model fits to the data: closer the data points to the regression line, higher the value of R^2

Explained variation



Total variation	=	Unexplained variation	+	Explained variation
$y - \bar{y}$	=	$y - \hat{y}$	+	$\hat{y} - \bar{y}$

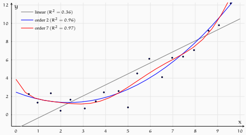
Dealing with non-linearity

- Least-squares estimation works because the regression equation is linear with respect to parameters $\boldsymbol{w}$ (error function is quadratic)
- Introducing non-linear combinations of inputs does not affect the estimation procedure. The following are still linear models

$$y = w_0 + w_1 x^2 + \epsilon$$
$$y = w_0 + w_1 \log(x) + \epsilon$$
$$y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + \epsilon$$

- In general, we can replace input x by a function of the input(s) $\Phi(x)$. $\Phi()$ is called a *basis function*
- Basis functions allow linear models to model non-linear relations by *transforming* the input variables

Example: polynomial basis functions

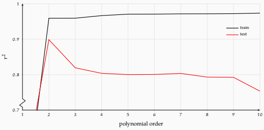


Overfitting

- *Overfitting* is an important problem in ML, happens when the model learns peculiarities/noise in the training data
- An overfitted model will perform well on training data, but worse on new/unseen data
- Typically 'more complex' models are more likely to overfit

Overfitting

demonstration through polynomial regression



Preventing overfitting

- A straightforward approach is to chose a simpler model (family), e.g., by reducing the number of predictors
- More training data helps: it is less likely to overfit if number of training instances are (much) larger than the paramters
- There are other methods (one is coming on the next slide)
- We will return to this topic frequently during later lectures

Regularized parameter estimation

- *Regularization* is a general method for avoiding overfitting
- The idea is to constrain the parameter values in addition to minimizing the training error
- For example, the regression estimation becomes:

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \sum_i (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{k} w_j^2$$

- The new part is called the regularization term,
- λ is a *hyperparameter* that determines the strength of the regularization
- In effect, we are preferring small values for the coefficients
- Note that we do not include w_0 in the regularization term

L2 regularization

The form of regularization, where we minimize the regularized cost function,

$$J(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|_2$$

is called L2 regularization.

- Note that we are minimizing the L2-norm of the weight vector
- In statistic literature L2-regularized regression is called *ridge regression*
- The method is general: it can be applied to other ML methods as well
- The choice of λ is important
- Note that the scale of the input also becomes important

L1 regularization

In L1 regularization we minimize

$$J(\boldsymbol{w}) + \lambda \sum_{j=1}^{k} |w_j|$$

- The additional term is the L1-norm of the weight vector (excluding w_0)
- In statistics literature the L1-regularized regression is called *lasso*
- The main difference from L2 regularization is that L1 regularization forces some values to be 0 – the resulting model is said to be 'sparse'

Regularization as constrained optimization

L1 and L2 regularization can be viewed as minimization with constraints

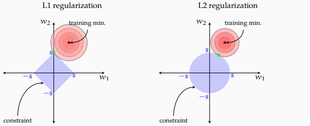
L2 regularization

$$\text{Minimize} \quad J(w) \quad \text{with constraint} \quad \|\boldsymbol{w}\| < s$$

L1 regularization

$$\text{Minimize} \quad J(w) \quad \text{with constraint} \quad \|\boldsymbol{w}\|_1 < s$$

Visualization of regularization constraints



Regularization: some remarks

- Regularization prevents overfitting
- The *hyperparameter* λ needs to be determined
  - best value is found typically using a *grid search*, or a *random search*
  - it is tuned on an additional partition of the data, *development* set
  - development set cannot overlap with training or test set
- The regularization terms can be interpreted as *priors* in a Bayesian setting
- Particularly, L2 regularization is equivalent to a normal prior with zero mean

Gradient descent for parameter estimation

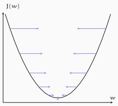
- In many ML problems, we do not have a closed form solution for finding the minimum of the error function
- In these cases, we use a search strategy
- *Gradient descent* is a search method for finding a minimum of a (error) function
- The general idea is to approach a minimum of the error function in small steps

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla J(\boldsymbol{w})$$

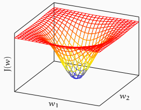
∇J is the gradient of the loss function, it points to the direction of the maximum increase
η is the learning rate

Gradient descent with single parameter

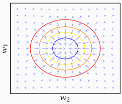
- For a single parameter, gradient is a one-dimensional vector
- The direction of gradient is towards the maximum increase
- We take steps proportional to $-\nabla J(w)$
- Steeper the curve, the larger the parameter update



Gradient descent with single parameter



Objective function



Negative gradients

Categorical predictors

- Categorical predictors are represented as multiple binary coded input variables
- For a binary predictor, we use a single binary input. For example, (1 for one of the values, and 0 for the other)

$$x = \begin{cases} 0 & \text{for male} \\ 1 & \text{for female} \end{cases}$$

- For a categorical predictor with k values, we use one-hot encoding (other coding schemes are possible)

$$\boldsymbol{x} = \begin{cases} (0,0,1) & \text{neutral} \\ (0,1,0) & \text{negative} \\ (1,0,0) & \text{positive} \end{cases}$$

Summary

What to remember:

- Supervised vs. unsupervised learning
- Regression vs. classification
- Linear regression equation
- Least-square estimate
- MSE, R^2
- non-linearity & basis functions
- L1 & L2 regularization (lasso and ridge)

Next:

Wed classification
Fri first lab session
Mon classification

Additional reading, references, credits

- Hastie, Tibshirani, and Friedman (2009) discuss introductory bits in chapter 1, and regression on chapter 3 (sections 3.2 and 3.4 are most relevant to this lecture)
- Jurafsky and Martin (2009) has a short section (6.6.1) on regression
- You can also consult any machine learning book (including the ones listed below)

Barber, David (2012). *Bayesian Reasoning and Machine Learning*. Cambridge University Press. ISBN: 9780521518147.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Second. Springer series in statistics. Springer-Verlag New York. ISBN: 9780387848587. URL: `http://web.stanford.edu/~hastie/ElemStatLearn/`.

James, G., D. Witten, T. Hastie, and R. Tibshirani (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics. Springer New York. ISBN: 9781461471387. URL: `http://www-bcf.usc.edu/~gareth/ISL/`.

Jurafsky, Daniel and James H. Martin (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. second. Pearson Prentice Hall. ISBN: 978-0-13-504196-3.

Mitchell, Thomas (1997). *Machine Learning*. 1st. McGraw Hill Higher Education. ISBN: 0071154671,0070428077,9780071154673,9780070428072.

A hands-on exercise

Draw a regression line over the plot