Statistical NLP: course notes

Çağrı Çöltekin — SfS / University of Tübingen

2020-05-25

These notes are prepared for the class *Statistical Natural Language Processing* taught in Seminar für Sprachwissenschaft, University of Tübingen.

This work is licensed under a Creative Commons “Attribution 3.0 Unported” license.

Draft lecture notes. Version: f258c55@2020-05-03; ml-basics.tex

4 Machine learning basics

Statistical methods, particularly methods from machine learning, has been the most successful solutions for many problems in natural languages processing. The methods from machine learning dominates the filed so much that many people consider NLP as a branch of machine learning.

Machine learning is about learning from data. Instead of writing a specialized program based on expert knowledge for solving a problem, we rely on generic 'programs', or models, which learn from data. As noted above, this has proven useful in many applications. However, machine learning also offers us ways to analyze the data at hand and arrive at generalizations that are sometimes impossible without use of these techniques.

This lecture introduces some of the basic ideas behind machine learning, alongside linear regression, a simple but fundamental model for learning from data.

4.1 Machine learning: broad categorization of methods

Machine learning methods are categorized into a number of broad categories in the literature. Most commonly, the methods are categorized based on the amount of supervision they need. On the one hand, a *supervised* method requires labeled data. That is, every object we want to classify (e.g., a document) has to be annotated with target information want to predict (e.g., the author of the document) in the training data. However, the aim of the method is to make predictions outside its training data. We want our models to generalize, not to memorize. On the other hand, an *unsupervised* method does not require any target label or information. The aim is to use the differences and similarities between the data points for finding useful patterns. The methods that exploit both annotated data (with target label/information) and unannotated data are called *semi-supervised*. Another interesting class of methods where success and failure is not associated with individual predictions but a collection of them is called *reinforcement learning*.

In this course we mostly focus on supervised methods, but also cover some of the unsupervised methods commonly used in the field.

IN SUPERVISED LEARNING the training data contains what we want to predict. The task of the system is, then, to learn this predictions

🅭🅯

from the training data in a way that it is useful for making predictions for new, unseen instances. An overall picture of supervised learning is provided in Figure 4.1.

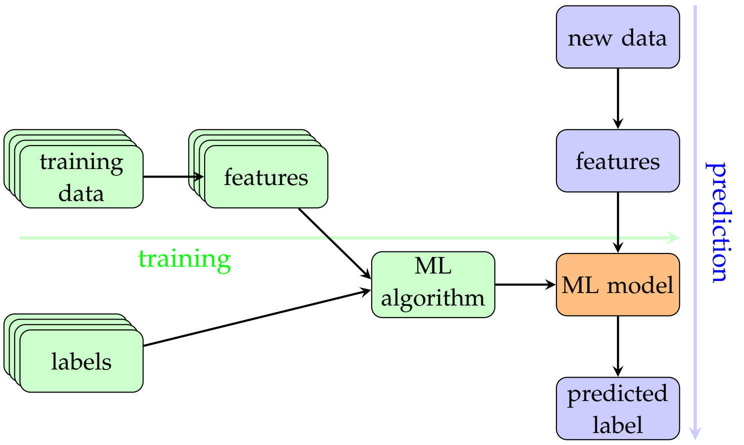


Figure 4.1: A picture of supervised learning.

During training, we need both our training data and the associated predictions (indicated as 'labels' in Figure 4.1). Typically, the objects we want to work with cannot be used directly with a machine learning algorithm. We need to first extract features that are useful for prediction and represent them in a form the machine learning algorithms can work with. For example, to classify documents, we may use the length of the documents, or number of times a particular word occurs in the document as features. The role of the machine learning algorithm is to find the best model (among a family of models) based on the training data. During prediction time, we first extract the features from a new data instance the same way we did during training, and predict the outcome using the model. An important point that is worth repeating is we want our model to perform well on the new data.[1]

If a supervised machine learning model predicts a numeric value it is called a *regression* model. A simple regression model predicting one numeric variable (y) from a single numeric predictor (x) is demonstrated in Figure 4.2. The small circles on the plot represent the pairs of x,y values observed. The red line is the model after observing this 'training' data. Once we have the model, our predictions for the new data points will be based on this line. The blue lines on the figure demonstrate the prediction of the model for $x = 5.5$, which turns out to be 5.7 according to this model.

If the model predicts a category or a label, it is called a *classification* model. Figure 4.3 demonstrates an example setting for classification, where the points marked with $+$ and $-$ are the data points expressed in a two dimensional feature space. Unlike in regression example above, both axes correspond to features in this example. The outcome, the category, is represented by the shape of the data point. Similar to regression, we estimate a model on the training data. The aim is to predict the class ($+$ or $-$) of an unseen data point.

In NLP we use classification more often than regression, since

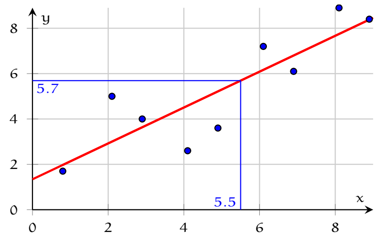


Figure 4.2: A demonstration of simple linear regression.

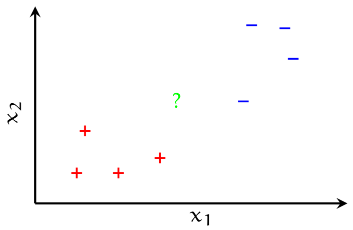


Figure 4.3: A demonstration of classification problem.

---

[1] And, we will repeat this many times in this class.

many properties of natural language data we want to predict are categorical. However, besides occasional practical use, understanding regression will also help understanding other machine learning methods in general. In this lecture we will introduce some of the basic concepts and issues in machine learning through regression, and return to classification next.

UNSUPERVISED LEARNING refers to a set of methods that allow finding interesting or useful patterns that are not explicitly marked in the data. Figure 4.4 show a set of data instances plotted on a two-dimensional space (based on two features). For example, these dots could represent the instances of speech sounds, while the features (axes) could be frequency and duration of each instance. Not having any labels (e.g., phonemes, or speakers these points belong to), we cannot use a supervised learning algorithm. However, it is easy for human eye to pick two groups in the data presented in Figure 4.4. Such methods allow exploring the data in insightful ways. However, once we build a model based on the extracted pattern, we can also assign group, or *cluster* memberships to new data items. Although we cannot assign a meaningful name to the clusters automatically, being able assign data points to clusters is also useful for supplementing supervised methods.

Most commonly used unsupervised methods include *clustering*, *density estimation* and *dimensionality reduction*. Clustering refers to the process we described above: given a set of unlabeled data points, the aim is to find a 'natural grouping' within the data. Density estimation is similar to clustering, but we assume data comes from a mixture of probability densities. As a result, each data point receives a probability (or likelihood) of coming from one of these probability distributions. In a way, density estimation makes 'soft assignments' to each density, or cluster. Dimensionality reduction aims to reduce a data set defined in a high-dimensional feature space into a lower dimensional space while retaining most of the information the data. We will revisit all these methods and discuss them in more detail later in this class.

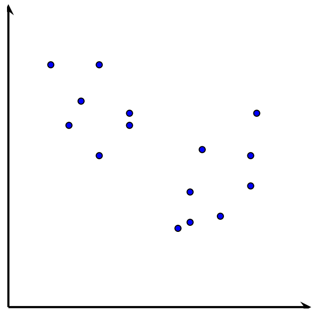


Figure 4.4: A set of unlabeled data points in a two-dimensional feature space.

4.2 *The linear regression model*

The linear regression is a simple, yet a very fundamental method in statistics (and machine learning). A simple linear regression model predicts value of a numeric variable, conventionally denoted y, from a set of predictors, denoted x.[2] In the simplest case of a single predictor, the model is expressed by Equation 4.1, which corresponds to a line in x–y plane.

$$y = a + bx \tag{4.1}$$

where y is the outcome variable we want to predict, x is our single predictor, and a and b are the parameters of the model, which are called *intercept* and *slope* respectively.[3] The intercept is the value at

[2] Note that x is a vector. In case of a single predictor, we also use the symbol x (a scalar, not a vector).

[3] The symbols a and b for intercept and slope are widely used conventions (especially in statistics). However, alternative notations instead of a and b include

- α and β
- θ_0 and θ_1
- w_0 and w_1

The indexed notations help when we extend this single-predictor model to multiple predictors. In some neural network literature intercept is sometimes denoted with letter b, as it is also called the *bias term*.

which the line 'intercepts' the y axis, and the slope is the slope of the line representing the linear equation on Euclidean space. Slope indicates the amount of change in y for each unit change in x. Figure 4.5 demonstrates a few examples of linear equations with different slope and intercept values.

A positive slope means the outcome y increases as x increases, while a negative slope indicates a decrease in y value as x increases. A slope of 0 simply means y is constant, it is not affected by the values of x, or in other words, the outcome and predictors are (linearly) independent.

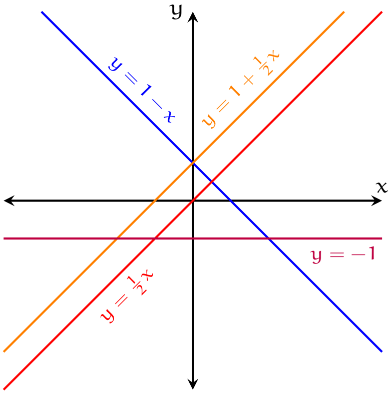


Figure 4.5: Example instances of Equation 4.1.

The equation generalizes to multiple predictors trivially. For k predictors, we have

$$y = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_k x_k \,. \tag{4.2}$$

We can simplify the notation by specifying the weight vector as $\boldsymbol{w} = (w_0, \ldots, w_k)$ and the input vector as $\boldsymbol{x} = (1, x_1, \ldots, w_k)$.[4] Then Equation 4.2 becomes

$$y = \boldsymbol{w}\boldsymbol{x}.$$

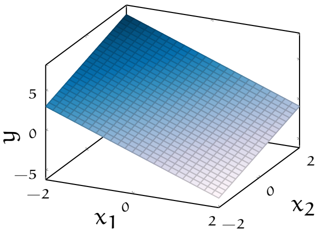


Figure 4.6: Visualization of a linear of a linear equation with two predictors: $y = 1 - 2x_2 + x_1$.

With multiple predictors, the equation defines a (hyper)plane. Figure 4.6 visualizes an example linear model with two predictors. With multiple predictors, we have multiple coefficients indicating the slope for each predictor. They still indicate the amount of change in the outcome variable for unit change in the corresponding predictor while all other predictors are kept constant. Effects of all predictors are additive, and independent of the effects of the other predictors. In the example in Figure 4.6, negative slope of x_1 means that y decreases as x_1 increase, while positive slope for x_2 means that increasing x_2 increases y. The value of y, however, is determined based on the linear combination of both. Beyond 2 predictors (three dimensions including the outcome variable), the visualization becomes impossible. However, the idea of a relationship, determined by a hyperplane generalizes to higher dimensions as well.

4.3 *Estimating parameters*

The model we briefly discussed above is useful for modeling a vast amount of phenomena. The linear model is most likely the most common tool used across all modern sciences. Equation 4.2 defines a 'model family'. Each choice of intercept and slope values defines another model. For some problems, these values are fixed, and one can find the values of the parameters with an analytic method of some sort. However, the aim in machine learning (and statistics) is to learn these parameters from data. We now turn to the question of how to find the 'best' parameters given a data set with observations of both predictors and the outcome variable.

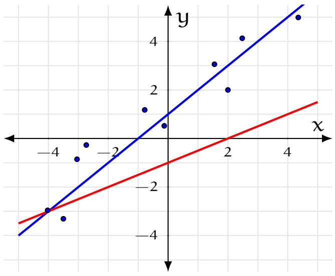


Figure 4.7: A typical data set for regression (dots). And possible linear regression models (blue and red lines).

In case of regression, the data we use looks like the one presented in Figure 4.7. We have a continuous predictor x, and we want to

[4] Note that x_0 is always 1.

predict the value y, where we have 10 observations (or data points, or training instances) represented by the dots in the figure. Our aim is to find a linear equation, a line like the ones presented in Figure 4.6, that allows us to predict y values for the future observations that are similar to the ones in the data set. We can view learning as choosing the best line among all possible lines. Figure 4.7 presents two candidate models with blue and red lines. Intuitively, the blue line is better than the red one. However, our aim is to formalize which models are better than the others, and find the best one given the data at hand.

The most common approach for estimating model parameters is to define an *error function* and find the parameter values that minimize the error on the training data. Figure 4.8 demonstrates the errors made by the two alternative models on the data presented in Figure 4.7. It is clear that the sum of the errors (the vertical lines) for the model represented with the blue line is smaller, and we should prefer this one instead of the red one.

So, to find the best linear regression line, we may look for the model parameters that minimize the sum of the lengths of the vertical line segments depicted in Figure 4.8. For the i^{th} data point (x_i, y_i) the error is simply the difference between the observed y-value, y_i and the model's prediction for x_i, which is simply $a + bx_i$. A typical notation for estimated values (in contrast to than real/observed ones) is to indicate it on the variable with a hat. Hence, we indicate the prediction of the model for data point i, as $\hat{y}_i = a + bx_i$, and the error (or *residual*) in this case would be $y_i - \hat{y}_i$. Since we want this value to be lower for the whole data set, we want to minimize the sum of this error over all data points. However, error as formalized above will be negative for some of the training examples, and positive for others. As a result, minimizing the sum of this error is not useful. A reasonable quantity to minimize is the absolute value of the error, $|y_i - \hat{y}_i|$. However, the absolute value function does not have some of the properties nice algebraic properties of squared differences.[5] The most commonly used error function for linear regression is the sum of squared errors, $(y_i - \hat{y}_i)^2$. Which is a convenient function to minimize, and as we will revisit later, it yields the model that assigns maximum likelihood to the data under the assumption that the errors are normally distributed.

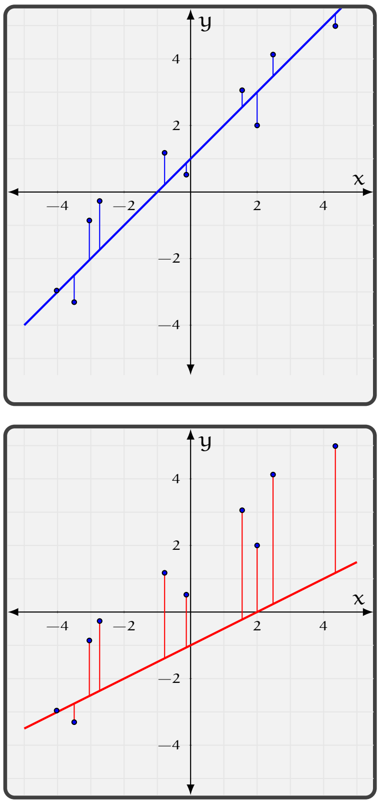


Figure 4.8: Demonstration of the errors made by the models represented by blue (top) and the red (bottom) lines in Figure 4.7.

In summary, a linear regression model is typically estimated from data by minimizing the error function

$$E(a,b) = \sum_i \left(y_i - \underbrace{(a + bx_i)}_{\hat{y}_i} \right)^2 . \qquad (4.3)$$

The formula is general. For a linear regression model with multiple predictors, using the vector notation described earlier,[6] we simply write

$$E(\boldsymbol{w}) = \sum_i (y_i - \hat{y}_i)^2 \text{ , where } \hat{y}_i = \boldsymbol{w}\boldsymbol{x}_i. \qquad (4.4)$$

---

[5] Mainly, the absolute value function is not differentiable everywhere.

[6] The first element of the parameter vector, $\boldsymbol{w}_0$, is the intercept, and the first element of the input vector $\boldsymbol{x}_i$ ($x_{i,0}$) is the constant 1.

You should have realized that we express the error function as a function of model parameters (and not the predictors) in the above formulations. Furthermore, as it is clear in Equation 4.3 that the error function is a quadratic function (a polynomial of degree 2) of model parameters (a and b). Quadratic functions are convex functions with a single global minimum. Taking the derivative of the error function, setting it to 0 and solving it results in the a and b values that gives us the minimum error on the training data. We will skip the derivation here, but present a version of the solution below. The best a and b values that minimumize the sum of squared errors are

$$b = \frac{\sigma_{xy}}{\sigma_x^2} = r_{xy}\frac{\sigma_y}{\sigma_x} \quad \text{and} \quad a = \bar{y} - b\bar{x}.$$

where $\bar{y}$ and $\bar{x}$ are the means of $\boldsymbol{x}$ and $\boldsymbol{y}$, σ_{xy} is the covariance of x and y, and σ_x^2 is the variance of x, and r_{xy} is the correlation coefficient between x and y. The important thing to note is that, the slope indicates the relation between x and y. In particular, it is proportional to the covariance between the variables, or, as the second formulation of b indicate, it is a scaled version of the correlation between the predictor and the outcome variable.

4.4 Least-squares regression as maximum-likelihood estimation

One of the reasons for using squared errors is the fact that sum of squared errors are mathematically convenient to work with. In a way, there is nothing special about minimizing sum of squared errors. One can also minimize some other measure of error, for example, sum of absolute values of the residuals. In fact, there are cases where such an alternative estimation method is desirable.[7] The error function defined this way, e.g., unlike sum of absolute errors, is differentiable and convex. Hence, it allows us to find an exact analytic solution to the minimization problem. However, there is another fact that makes least-squares regression interesting.

A general method of estimation in statistics and machine learning is the *maximum-likelihood estimation* (MLE). The general idea of the MLE is this: given a family of models, we prefer the one that assigns the maximum likelihood to the observed (training) data. In case of linear regression, to be able to determine the likelihood of a particular data point (x_i, y_i), we need to make an assumption about how the data is distributed around the regression line. If we assume that the residuals are distributed normally with zero mean,[8] then likelihood assigned to a particular data point (x_i, y_i) is $\mathcal{L}(\boldsymbol{w}) = p(y_i \mid x_i; \boldsymbol{w})$. Note that we view likelihood as a function of model parameters. Informally, the model will assign high likelihood if the data point is close to its prediction, and for data points farther from the model prediction, likelihood will be low. Since we assume that each data point is independently sampled, we obtain the likelihood of the training data by multiplying the individual likelihoods of all the data points.

[7] Particularly, least-squares regression is known to be sensitive to large residuals, especially if those are close to the extreme values of the predictor. Estimating a regression model by minimizing absolute values for the residuals is more 'robust' against the outliers, and often used as a robust alternative to least-squares regression.

[8] We know from central limit theorem that this is a reasonable assumption in many problems, since the errors are likely to be due to sum of many random factors (variables).

As a result, we want to maximize

$$\mathcal{L}(\boldsymbol{w}) = \prod_i p(y_i \,|\, x_i, \boldsymbol{w})$$

where $p(\cdot)$ is the probability density function of the normal distribution. In practice, we prefer to work with logarithms, and minimization rather than maximization.[9] Now, if we put all of the above together we want to minimize the objective function $E(\boldsymbol{w})$

$$\begin{aligned}
\hat{\boldsymbol{w}} &= \arg\min_{\boldsymbol{w}} -\log \mathcal{L}(\boldsymbol{w}) \\
&= \arg\min_{\boldsymbol{w}} -\log \prod_i \frac{e^{-\frac{(y_i-\hat{y}_i)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}} \\
&= \arg\min_{\boldsymbol{w}} -\sum_i \log e^{-\frac{(y_i-\hat{y}_i)^2}{2\sigma^2}} - \log \sigma\sqrt{2\pi} \\
&= \arg\min_{w} \sum_i (y_i - \hat{y}_i)^2 \,.
\end{aligned}$$

The derivation above skips over some details, and it is not essential to follow all the steps for our purposes.[10] However, what it tells us it that if we assume that the residuals are distributed normally, the least-squares solution is also the maximum likelihood solution.

4.5 *Measuring success*

For any machine learning system, we need a way to measure its success. Remember, however, our aim is to generalize, and predict new data correctly rather doing well on the training set. We will leave this issue aside for now, and concentrate on the measures we use to assess the success of the model.

The error function we use during estimation provides a clear metric of success. The smaller the error, the better the model. In case of regression, the error we minimize is the sum of squared error (SSE). However, this sum depends on the size of the data it is calculated on. Hence, we want to take the effect of the data size out, so that two systems that are tested on different data sets should be comparable. One can easily achieve this by taking the mean of the SSE, MSE. However, it is often desirable to measure the error in the same units as our data. For example, if our task is to evaluate a regression model predicting grades of student essays, we want to know error in number of grade points on average, rather than its square. Hence, the most error common measure to check and report is the root mean square error (RMSE), which is defined as

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

Another quantity, that measures success rather than error, is *coefficient of determination* (R^2). R^2 is the ratio of conditional variance

[9] Since logarithm of a variable increases and decreases (monotonically) with the value of the variable, maximizing or minimizing the logarithm will maximize or minimize the variable itself.

[10] It is not really difficult to follow though. All you need to remember is that $\log e^x = x$, and constants (the terms that do not include the model parameters) do not affect minimization, and they can be dropped. Also note that the only term in the equation that depends on the parameters is the model's estimation $\hat{y}$.

(the variance around model prediction) divided by the variance of the unconditional mean of the outcome variable y. As expected, the coefficient of determination is also related to RMSE. Put more formally,

$$R^2 = \frac{\sum_i^n (\hat{y}_i - \bar{y})^2}{\sum_i^n (y_i - \bar{y})^2} = 1 - \left(\frac{\text{RMSE}}{\sigma_y}\right)^2 . \tag{4.5}$$

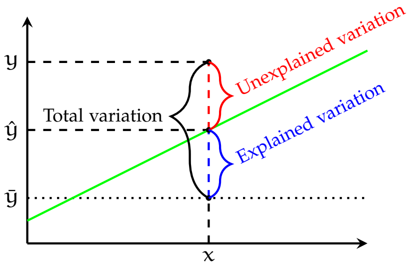


Figure 4.9: A visualization of the explained and unexplained variation in regression.

The R^2 is unitless, and can be interpreted as the variation in the data explained by the model. This is depicted in Figure 4.9. The actual observation for x in the figure is denoted with y, while model's prediction, conditional mean of y given x, is $\hat{y}$. The unconditional mean of the outcome variable is denoted with $\bar{y}$. Total variation (for this data point) refers to the distance of the observation from the mean $\bar{y}$. In a way, if we did not know x, $\bar{y}$ would be our best guess. For this particular data point, knowing x helps the model to make a better prediction. The dashed line segment drawn in blue is the amount the model helps. Yet, we still have some error, the 'unexplained variation' marked with red in the figure. The R^2 is the ratio of the explained variation to the total variation. For a simple regression model, R^2 is the square of the correlation coefficient between x and y. However, as you can also see from Equation 4.5, R^2 can be calculated for a regression model with any number of predictors, and the interpretation stays the same. R^2 measures the dependence of the linear combination of the predictors and and the outcome variable.

4.6 *Linearity in linear regression*

Linear regression, as we discussed so far is suited for problems where the relation between the predictors and outcome variable is linear. Sometimes, however, the relation is known to be non-linear. For example, if we want to predict some cognitive ability through lifetime, it is likely that it will improve first during childhood, and then deteriorate in later life by by aging. Similarly, rate of increase or decrease of the outcome conditioned on the predictor(s) may not be constant. Figure 4.10 shows such a data set, with the linear regression fit as described above.

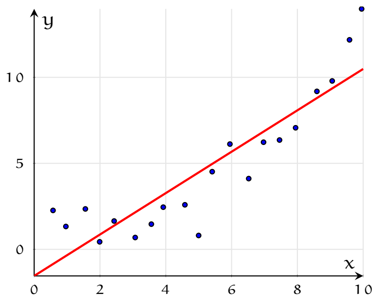


Figure 4.10: A typical data set for regression (dots). And possible linear regression models (blue and red lines).

Although the linear equation found does not seem too bad ($R^2 = 0.36$), it is clear that a curve could be a better fit than a line. In other words, the relation between x and y seems non-linear.

Despite the word 'linear' in the name, the estimation method above works perfectly well for non-linear combinations of the input variable(s) as well. The important restriction about the linearity is about the model parameters, w. As long as the model parameters are linear, we are free to add non-linear (combinations) of the input variables as predictors in a linear regression model. *Polynomial regression* is a particular way to estimate non-linear regression models, where we add higher degree polynomial terms. For example, in our example with a single single variable, formulated as $\hat{y} = w_0 + w_1 x$, we can add the square of the predictor, which would give us the model $\hat{y} = w_0 + w_1 x + w_2 x^2$. For the estimator, the higher order term x^2 is just another predictor, and the parameters can be estimated as usual.

Figure 4.11 presents examples of polynomial regression fit to the data from Figure 4.10. The curves represent polynomials of order 2 and 7 (as well as the line showing linear, polynomial order 1, regression). In general, one can use any nonlinear function of the input variables for fitting a linear regression model. This includes functions involving combinations of input variables when there are more than one. For example, if we had two predictors, x_1 and x_2, a possible nonlinear combination could be $x_1 \times x_2$. Non-linearity is a concept we will revisit in more detail later.

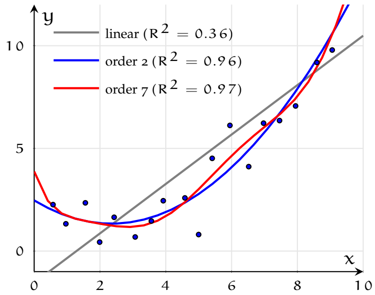


Figure 4.11: Example polynomial regression models fit to the same data in Figure 4.10.

4.7 *Overfitting*

In the polynomial regression examples Figure 4.11, we see that increasing the order of polynomial increases the model's fit to the data (as measured by the R^2 on the training set). However, the highest order polynomial seems to not to generalize, but learn some peculiarities of the training set. Intuitively, the second-order polynomial is a better, more general, fit to this data.

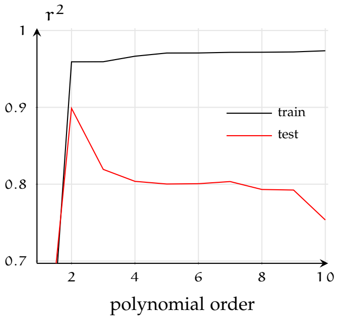


Figure 4.12: Progression of training and test errors in polynomial regression, as the order of the polynomial increased.

To demonstrate the effect Figure 4.12 plots the models fit (R^2) to the training set, as well as a test set obtained from the same distribution. As the degree of the polynomial increases, the fit to the training set increases (although only slightly after degree 2). However, the fit to the test set starts decreasing. Even though we demonstrated this with polynomial regression with different order of polynomials, this is a general phenomenon called *overfitting*. As the model complexity increases (e.g., with inclusion of more predictors, and hence, more parameters) the chances for overfitting increase. The model starts learning the noise in the training set more than the generalizations that are helpful outside the training data.

4.8 *Regularization*

As we have already repeated a few times, our aim is not to get the best results on the training data (for the values of outcome variable we already know). The aim is to find a general solution that works for new data instances for which we do not know the value of the outcome variable. As a result, overfitting is something we have to avoid. Since overfitting is likely when the model is more complex, one option is to select models that are simpler – but not simpler than needed. There are a number of measures that help with model selection which seek a balance between the success of the model on the training data and number of parameters.[11] However, in ML and NLP, we often deal with a very large number of parameters, and the model selection process becomes tedious at best. Here we are going to discuss a more general technique called *regularization* for preventing overfitting.

[11] We will not discuss these metrics of model selection here, you are recommended to revise some of these metric. A well-known criterion used particularly in statistics is Akaike information criterion.

The idea with regularization is to modify the error function we minimize such that as well as parameter values that reduce the training error, we prefer parameter values from a restricted set, which leads to simpler models. This way, we do not (necessarily) simplify our models by reducing the number of parameters, but by setting a preference towards certain parameter values. Instead of minimizing the error term in Equation 4.4, we extend the objective function with a term that prefers smaller weight vectors.

$$J(\boldsymbol{w}) = \sum_i (y_i - \hat{y}_i)^2 + \lambda \|\boldsymbol{w}\| \tag{4.6}$$

where λ is a constant, and $\|\boldsymbol{w}\|$ is the L2 norm of the weight vector (excluding the intercept term).

Equation 4.6 defines the *L2 regularization* where the estimation procedure puts a preference towards coefficient vectors with small L2 norms. Intuitively, to make the L2 norm of the vector smaller, the estimation procedure will push coefficients that are not strongly supported by the data to smaller values. In fact, most effects of overfitting result in very large coefficients, as it requires small changes in the data to have large effects on the output. As a result regularized estimation simplifies the model in a 'soft' manner, rather than simplifying the model by explicitly removing predictors. L2-regularized regression is also called *ridge regression*.

Equation 4.6 can also be expressed in terms of constrained optimization. Minimizing $J(w)$ in Equation 4.6 equivalent finding parameter values that minimize the sum of squared errors with subject to the constraint that the L2 norm of the parameter vector is smaller than a constant s. That is, we minimize

$$\sum_i (y_i - \hat{y}_i)^2 \quad \text{with constraint} \quad \|\boldsymbol{w}\| < s \tag{4.7}$$

This formulation of the optimization problem is equivalent to Equation 4.6, which is generally more convenient to work with. However, the formulation in Figure 4.7 may help understanding the concept better.

Another popular regularization methods is *L1 regularization*, where instead of the L2 norm, we add the L1 norm of the parameter vector to the objective to be minimized. L1-regularized regression is also called *lasso*.

The main difference between the L1 regularization and the L2 regularization is that the L1 regularization tends to set some of the coefficients to 0, while the L2 regularization results in small but non-zero coefficients. A demonstration of this difference is presented in Figure 4.13.

The figures depicts the constraints defined (as in Equation 4.7) by L1 and L2 regularization as blue regions, and the training objective as as a red circle whose smaller values are represented with darker shades in a space of two parameters. Constraint space defined by L1 regularization is 'pointy' (a square in 2D, a (hyper)cube in higher dimensions). Hence, as demonstrated in the lower graph

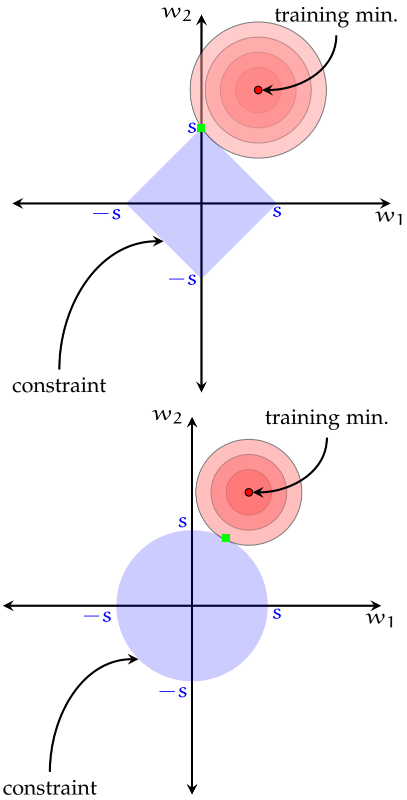


Figure 4.13: Visualization of L1 and L2 regularization.

of, the minimum value of the training objective that also satisfies the constraint is likely to coincide with corners of the hypercube, which will result in the corresponding weight values to be 0. The L2 regularization constraint defines a hypersphere, which will more likely to meet with the training objective in point with non-zero parameter values.

We delayed the discussion of λ in Equation 4.6. λ is a *hyperparameter* that determines the strength of the regularization. Higher λ values will result in stronger regularization. With larger values of λ, the estimation procedure will pay more attention to reducing the norm of the weight vector (or, equivalently the area/volume of the constraint will be smaller). Lower values will result in more attention to reducing training error. The optimum value of λ depends on the problem and the data. As a result it needs to be determined empirically. We will return to tuning hyperparameters in more detail later. For now, it is important to note that to determine the best value of λ we need to leave aside part of the data, often referred to as *development set*. In this scenario, we train our model on the training set multiple times with different λ values, and pick the lambda value that yields best results on the development set.

4.9 *Gradient descent*

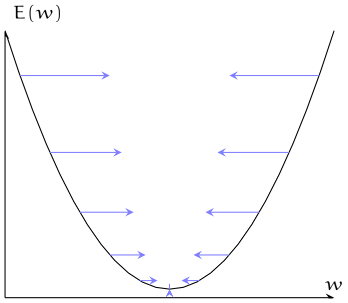


Figure 4.14: The negative of the derivatives on a quadratic curve.

So far, we worked with models for which we can find the best parameter values through an analytic (closed form) solution. That is, we take the derivative of the function with respect to the weights, set it to 0 and solve the resulting equation(s) to find the minimum point of the error function. When we have more than one parameters, we want the gradient vector, the partial derivatives with respect to each parameter to be 0. Otherwise the procedure is the same.

Although there are analytic solutions for the regression model we discussed in this lecture (including with L1 and L2 regularization), there is no known analytic solution for most models we are interested in. In that case, we apply a search based strategy to find the values of the parameters that yield the minimum error. The general procedure that searches for the minimum of the error function, is called the *gradient descent*.

The gradient procedure relies on the fact that the gradient of a function indicate the largest direction of increase on the surface defined by the function. Figure 4.15 presents a quadratic error function of a single variable (parameters) with its negated derivatives. The arrows point to the direction of the minima, as well as indicating the steepness of the curve at that point. Both the direction and the magnitude of the gradient are helpful in estimating a model's parameters using gradient descent. Since the demonstration in Figure 4.15 includes only a single variable, the gradient vectors are one dimensional. With more than one parameter the same idea holds, the gradient vectors and the error surface will be multi dimensional. Figure 4.15 presents a similar error function for two parameters, and

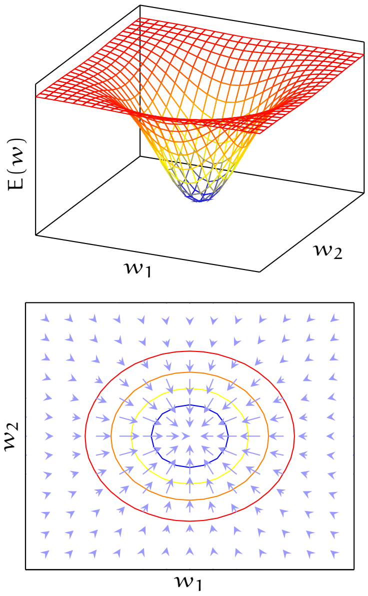


Figure 4.15: A quadratic error curve in 3D (top), and the negated gradient vectors at various points (bottom).

the samples of gradient vectors in the parameter space.

Gradient descent starts with setting the parameter vector to a random value, and updates the parameter vector iteratively in the opposite direction of the gradient, until we rich the minimum point where the gradient vector is 0. Since reaching the exact minimum is highly unlikely, in practice, the search stops when the magnitude of the gradient is smaller than a small constant.

Formally this is an iterative search procedure where we update the parameter vectors at each step i according to

$$\boldsymbol{w}_i \leftarrow \boldsymbol{w}_{i-1} - \eta\nabla E(\boldsymbol{w}_{i-1}). \tag{4.8}$$

To make it more concrete, this would translate to

$$(a_i, b_i) \leftarrow (a_{i-1}, b_{i-1}) - \eta\nabla \sum_j \big(y_j - (a_{i-1} + b_{i-1}x_j)\big)^2$$

for simple linear regression.[12] In words, we update the parameter vector (a, b) in the reverse direction of the gradient of the sum of squared residuals, proportional to the magnitude of the gradient. This means the gradient descend will take larger steps towards the minimum if the surface of the error function is steep, and smaller steps if it is relatively flat, which is likely when we are closer to the minima for a convex function. The multiple η in the above formulas is called the *learning rate*. It is yet another hyperparameter that depends on the problem and the data set. If it is set too low, the procedure will converge slowly, if it is set too high, there is the risk of overshooting, skipping over the minimum point (possibly back-and-forth) and not being able to converge to it.[13]

Gradient descend is an important estimation method used in many modern machine learning methods. We will return to it, and introduce some of the variations during this course.

4.10 *A worked out example*

The discussion of estimation procedure above may feel all too abstract, and keeping up with the all the notation and concepts may be difficult. Now we go through a fully worked-out example, by estimating a simplified regression model for the small data set in Table 4.1, first analytically, then using gradient descent. For simplicity, we will assume that we already know that the intercept is 0. This leaves only a single parameter, the slope to estimate. The regression equation becomes $y = bx$.

4.10.1 *The analytic solution*

To find the best parameter b, we need to find the b value that minimizes the error. Using the least-squares error, the error function is

$$\begin{aligned}\sum_i (y_i - bx_i)^2 &= \underbrace{(-1 + 1.02b)^2}_{i=1} + \underbrace{(0 + 0.15b)^2}_{i=2} + \underbrace{(1 - 1.04b)}_{i=3} \\ &= 2b^2 - 4.12b + 2.15\,.\end{aligned}$$

Table 4.1: A small data set for demonstration of regression. The data is obtained from the true model $y = x$ with some added random noise.

index	x	y
1	−1.00	−1.02
2	0.00	−0.15
3	1.00	1.04

---

[12] Note that for every step of the procedure we need to iterate over all the data set, which may be computationally expensive for large data sets. In more complex systems, as we will introduce later, updates based on smaller parts of the data generally leads to faster (and better) solutions.

[13] For simpler systems reasonable defaults for learning rate generally work fine. For other, more complex systems there are methods that modify the learning rate during training

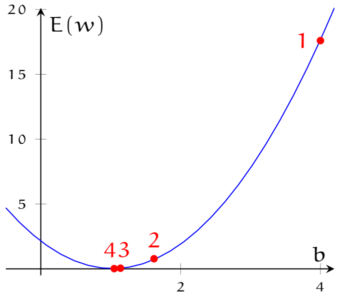
Taking it's derivative with respect to b, we obtain $4b - 4.12$. If we set $4b - 4.12 = 0$ and solve it, we will arrive at the best b value for least-squares regression, which is 1.03 for this data set.

4.10.2 *Gradient descent*

Applying gradient descent for a problem with a closed form solution does not make much sense. However, we will do it here for the sake of demonstration. Now, we assume that we can take the derivative of the error function, as we did above, but we do not know how to solve it analytically. For the gradient descent solution, we need to set our learning rate and a small threshold at which we stop iterating.

Figure 4.16 demonstrates the gradient descent run on our toy problem. For this demonstration we set the learning rate to 0.2, and stopping criterion as the derivative being smaller than 0.1. At first step, we initialize the b value 'randomly' to 4. The derivative of the error function at this point is 11.88. We multiply this number with our learning rate, and subtract the result from the current b value, which gives us the new b value of 1.62. We continue this process, until the derivative is close to 0. The other steps are presented in the lower part of Figure 4.16. Further iterations would estimate the parameter better, getting closer to the analytic solution calculated above (1.03).

Note that for the earlier iterations in Figure 4.16, the absolute value of the derivative (the magnitude of the gradient vector) is larger. Hence, we take larger stops towards the minimum. As we get closer to the minimum, the steps become smaller and smaller, as the magnitude of the gradient decreases alongside the rate of change of the error function.



step	b	gradient	error
1	4	11.88	17.66
2	1.62	2.38	0.73
3	1.15	0.48	0.05
4	1.05	0.10	0.02

Figure 4.16: Demonstration of gradient descent. The red dots in the figure indicate the points on the error curve for each step. The table below the figure lists the step, the b value, its derivative (gradient) and the value of the error function explicitly.

4.11 *A practical issue: categorical predictors*

So far, we assumed that the predictors of the regression model are numeric variables. In many problems, however, we have non-numeric predictors, such as part-of-speech tag of a word or the native language of a speaker. The most common way to include categorical variables as predictors is the *one-hot* or *one-of-k* encoding, where we use binary vectors with all values except the index of the category set to 0. Table 4.2 shows an example where 5 POS tag categories are coded using one-hot encoding. There are other ways of coding categorical predictors, but we will not cover them here as they are rarely used in machine learning and NLP.

Table 4.2: Example one-hot encoding of five POS tag categories.

POS tag	coded
Noun	00 001
Verb	00 010
Adjective	00 100
Adverb	01 000
Pronoun	10 000

Summary

This lecture introduces some of the basic concepts in machine learning alongside an introduction to regression. We will return most of these topic and expand on them during the coming lectures.

Linear regression is one of the most fundamental topics in statistics and machine learning. As a result there are numerous sources that you can read more about it. The familiar sources we use in this class also include introductory chapters or sections on regression, as well as some of the other concepts briefly introduced here. 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. All general/introductory books on statistics and machine learning include an introduction to regression (e.g, MacKay 2003; Bishop 2006; James et al. 2013).

Bibliography

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Springer. ISBN: 978-0387-31073-2.

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. et al. (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.

MacKay, David J. C. (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press. ISBN: 978-05-2164-298-9. URL: `http://www.inference.phy.cam.ac.uk/itprnn/book.html`.