Companion to Machine Learning

Rohan Kumar

Contents

Introduction

0.1 What is Machine Learning

Machine Learning is the field of study that gives computers the ability to learn from data without being explicitly programmed. This is good for problems that require a lot of fine-tuning or for which using a traditional approach yields no good solution. Machine Learning's data dependency allows it to adapt to new data and gain insight for complex problems and large amounts of data.

0.2 Applications of Machine Learning

Machine Learning can be used for a range of tasks and can be seen used in:

- Analyzing images of products on a production line to automatically classify them (Convolutional Neural Net)
- Forecasting company revenue based on performance metrics (Regression or Neural Net)
- Automatically classifying news articles (NLP using Recurrent Neural Networks)
- Summarizing long documents automatically (Natural Language Processing)
- Building intelligent bot for a game (Reinforcement Learning)

0.3 Types of Machine Learning

Supervised Learning Hands-On Machine Learning

In supervised learning, the training set you feed to the algorithm includes the desired solutions, called labels. (e.g determining if an email is spam would be trained a dataset of example emails labelled as spam or not spam.)

Some commonly used supervised learning algorithms are:

- k-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees and Random Forests
- Neural Networks

Unsupervised Learning Hands-On Machine Learning

In unsupervised learning, the training data is unlabeled and the system tries to learn without guidance. The system will try and automatically draw inferences and conclusions about the data and group it as such. (e.g. having a lot of data about blog visitors. Using a clustering algorithm we can group and detect similar visitors).

Some important unsupervised learning algorithms are:

- Clustering
  - K-Means
  - DBSCAN
  - Hierarchical Cluster Analysis
- Anomaly detection and novelty detection
  - One-class SVM
  - Isolation Forest
- Visualization and dimensionality reduction
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - Locally Linear Embedding (LLE)
  - t-Distributed Stochastic Neighbor Embedding (t-SNE)
- Association rule learning
  - Apriori
  - Eclat

Semisupervised Learning — Hands-On Machine Learning

Labelling can be very time-consuming and costly, often there will be plenty of unlabelled and a few labelled instances. Algorithms that deal with data that is partially labeled is called semi-supervised learning. A good example of this is Google Photos. Google clusters and groups your photos based on facial recognition (unsupervised) and then you can label one photo and it will be able to label every picture like that (supervised). Most semi-supervised learning algorithms are combinations of unsupervised and supervised algorithms.

Reinforcement Learning — Hands-On Machine Learning

Reinforcement Learning is a learning algorithm based on a reward system. The learning system, called an agent, can observe the environment, select and perform actions, and get rewards in return (or penalties in the form of negative rewards). It will then learn by itself what the best strategy, called a policy, to get the most reward over time. A policy defines what action the agent should choose when it is in a given situation.

Sources

Throughout this compendium, each piece of information will be formatted as such.

Name / Description of fact Source

Information about fact.

The location which currently contains “Source” could potentially be filled with a variety of sources. Here is how to find the source based off the shortened form.

- **Hands-On Machine Learning** refers to Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition by Aurélien Géron

1 Data Analysis

TODO:

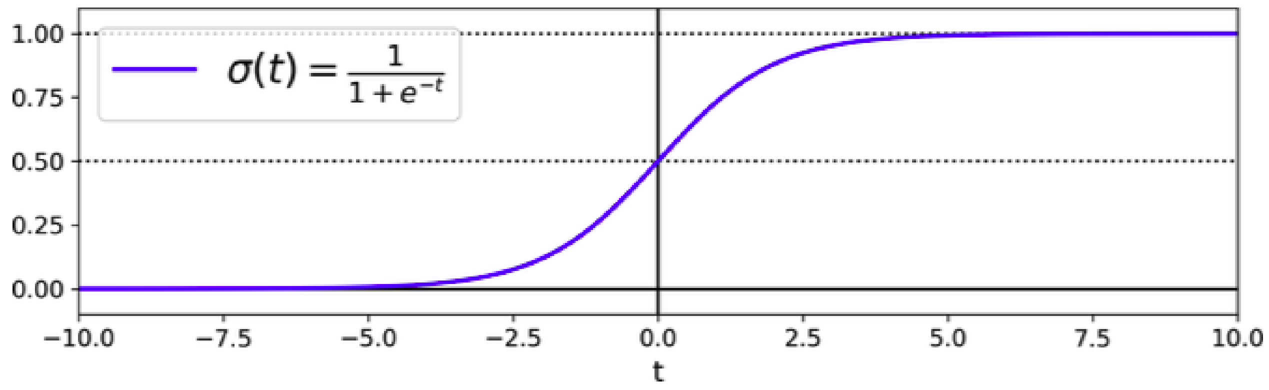
- Notation Sheet
- Data Model
- Overfitting/Underfitting
- Feature Normalization
- Types of Loss Functions
- Show what the gaussian distribution expansion is

2 Activation Functions

2.1 Sigmoid Function

The *sigmoid function* denoted $\sigma(\cdot)$ outputs a number between 0 and 1. It is defined as

$$\sigma(t) = \frac{1}{1 + exp(-t)}$$



The key property of the sigmoid function is that $\sigma(t) < 0.5$ when $t < 0$, and $\sigma(t) \geq 0.5$ when $t \geq 0$, so a sigmoid function is useful for classification since it can predict 1 when $\boldsymbol{w}^T\boldsymbol{x}$ is positive and 0 if it is negative.

3 Convex Optimization

TODO:

- Gradient Descent
- Normal Solution
- Other methods e.g Newtons Method

4 Statistical Learning

TODO:

- Show statistical derivation of Linear Regression
- Mixture of Gaussian
- Show statistical derivation of Logistic Regression

Data is often incomplete, indirect, or noisy. Statistical learning lets us consider forms of uncertainty to help us build better models.

4.1 Bayesian Learning

Bayes' theorem describes the probability of an event H given evidence e.

$$\begin{aligned} P(H|e) &= \frac{P(e|H)P(H)}{P(e)} && (1) \\ &= kP(e|H)P(H) && (2) \end{aligned}$$

where:

- $P(H|e)$: Posterior probability
- $P(e|P)$: Likelihood
- $P(H)$: Prior probability
- $P(e)/k$: Normalizing constant

Bayesian Learning consists of determining the posterior probability using Bayes' theorem.

Suppose we want to make a prediction about an unknown quantity X we can consider the hypothesis space which represents all possible models h_i to predict the scenario.

$$\begin{aligned} P(X|\boldsymbol{e}) &= \sum_i P(X|e, h_i)P(h_i|e) && (3) \\ &= \sum_i P(X|h_i)P(h_i|e) && (4) \end{aligned}$$

This prediction yields the weighted combination of all the hypothesis' in the hypothesis space based on it's likelihood from the evidence. The prior $P(h_i|e)$ is yields the weight for each hypothesis and $P(X|h_i)$ yields the likelihood of the hypothesis for the unknown quantity X.

Bayesian probability is:

- Optimal: give a prior probability, no prediction is correct more often than the Bayesian prediction.

- Overfitting-free: all hypothesis are weighted and considered, eliminating overfitting.

One of the constraints of bayesian learning is that it can be intractable when the hypothesis space grows very large, often as a result of approximating a continuous hypothesis space with many discrete hypothesis. This requires us to approximate Bayesian Learning.

4.2 Approximate Bayesian Learning

4.2.1 Maximum a Posteriori

Maximum a Posteriori (MAP) makes predictions based on only the most probable hypothesis $h_{MAP} = argmax_{h_i} P(h_i|e)$. This differs from Bayesian learning, which makes predictions for all hypothesis weighted by their probability. MAP and Bayesian learning predictions tend to converge as the amount of data increases, and overfitting can be mitigated by giving complex hypothesis a low prior probability. However, finding h_{MAP} may be difficult or intractable.

4.2.2 Maximum Likelihood

Maximum Likelihood (ML) simplifies MAP by assuming uniform prior probabilities and then makes a prediction based on the most probable hypothesis h_{ML}. ML tends to be less accurate than MAP and Bayesian predictions, it is also subject to overfitting due to the prior probabilities being uniform. Finding h_{ML} is easier than finding h_{MAP} since finding h_{ML} for $P(e|h)$ is equivalent to calculating it for $argmax_h \sum_n logP(e_n|h)$.

4.3 Bayesian Linear Regression

Instead of taking the hypothesis $\boldsymbol{w}$ that maximizes the posterior we can compute the posterior and work with that directly as follows:

$$\begin{aligned} P(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{X}) &= \frac{P(\boldsymbol{y}|\boldsymbol{w}, \boldsymbol{X})P(\boldsymbol{w}|\boldsymbol{X})}{P(\boldsymbol{y}|\boldsymbol{x})} \\ &= ke^{-\frac{1}{2}(\boldsymbol{w}-\overline{\boldsymbol{w}})^T \boldsymbol{A}(\boldsymbol{w}-\overline{\boldsymbol{w}})} \\ &= N(\overline{\boldsymbol{w}}, \boldsymbol{A^{-1}}) \end{aligned}$$

where

$$\begin{aligned} \overline{w} &= \sigma^{-2}\boldsymbol{A^{-1}}\overline{\boldsymbol{X}}\boldsymbol{y} \\ A &= \sigma^{-2}\overline{\boldsymbol{X}\boldsymbol{X}}^T + \boldsymbol{\Sigma^{-1}} \end{aligned}$$

4.3.1 Prediction

Let us consider an input $\boldsymbol{x_*}$ for which we want a corresponding prediction y_*.

$$\begin{aligned}
P(y_*|\overline{\boldsymbol{x_*}}, \overline{\boldsymbol{X}}, \boldsymbol{y}) &= \int_{\boldsymbol{w}} P(y_*|\overline{\boldsymbol{x_*}}, \boldsymbol{w}) P(\boldsymbol{w}|\overline{\boldsymbol{X}}, \boldsymbol{y}) d\boldsymbol{w} \\
&= k \int_{\boldsymbol{w}} e^{-\frac{(y_* - \overline{\boldsymbol{x}}^T \boldsymbol{w})^2}{2\sigma^2}} k e^{-\frac{1}{2}(\boldsymbol{w}-\overline{\boldsymbol{w}})^T \boldsymbol{A} (\boldsymbol{w}-\overline{\boldsymbol{w}})} d\boldsymbol{w} \\
&= N(\overline{\boldsymbol{x_*}}^T \boldsymbol{A}^{-1} \overline{\boldsymbol{X}} \boldsymbol{y}, \overline{\boldsymbol{x_*}}^T \boldsymbol{A}^{-1} \overline{\boldsymbol{x_*}})
\end{aligned}$$

This gives us a gaussian distribution of the solution. Generally for the prediction we take the mean of the distribution.

4.4 Mixture of Gaussians

Now we consider the probabilistic generative model for classification. We can compute the posterior $P(C|\boldsymbol{x})$ according to Bayes' theorem to estimate the probability of the class for a given data point. Here we are using Bayes theorem for inference rather than for Bayesian learning (estimating parameters of a model).

$$\begin{aligned}
P(C|\boldsymbol{x}) &= \frac{P(\boldsymbol{x}|C)P(C)}{\sum_C P(\boldsymbol{x}|C)P(C)} \\
&= kP(\boldsymbol{x}|C)P(C)
\end{aligned}$$

where:

- $P(C)$: Prior probability of class C
- $P(\boldsymbol{x}|C)$: class conditional distribution of $\boldsymbol{x}$

with the following assumptions:

- In classification the number of classes is finite, so a natural prior $P(C)$ is the multinomial $P(C = c_k) = \pi_k$
- when $\boldsymbol{x} \in \mathbb{R}$ then it is often ok to assume that $P(\boldsymbol{x}|C)$ is Gaussian.
- Assume the same covariance matrix $\boldsymbol{\Sigma}$ is used for each class.

From our assumptions we get

$$P(\boldsymbol{x}|c_k) \propto e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu}_k)}$$

4.4.1 Binary Classification

Subbing our assumptions into Bayes theorem for binary classification and simplifying, we get the following posterior distribution for classes c_k, c_j.

$$P(c_k|\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{w}^T \boldsymbol{x} + w_0}}$$

where:

$$\boldsymbol{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_j)$$
$$w_0 = \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \frac{1}{2} \boldsymbol{\mu}_j^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_j + ln\frac{\pi_k}{\pi_j}$$

We can observe that this equation is the logistic sigmoid and we can draw the class boundary at $\sigma(\boldsymbol{w}^T\boldsymbol{x} + w_0) = 0.5$.

4.4.2 Multinomial Classification

Now similarly for a multi-class problem with K classes we get.

$$P(c_k|\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_k^T \boldsymbol{x}}}{\sum_j e^{\boldsymbol{w}_j^T \boldsymbol{x}}}$$

where

$$\boldsymbol{w}_k^T = (-\frac{1}{2} \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + ln\pi_k, \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1})$$

This process can be extrapolated for classes that aren't all distributed with a gaussian distribution (e.g exponential, poisson, bernoulli etc ...). We can see that this is a specific case of the softmax distribution which is a generalization of the sigmoid and is discussed in further detail in the next section.

4.4.3 Parameter Estimation

Let $\pi = P(y = C_1)$ and $1 - \pi = P(y = C_2)$ where $P(x|C_1) = N(x|\mu_1, \Sigma)$ and $P(x|C_2) = N(x|\mu_2, \Sigma)$. In order to actually use bayesian inference to get the classification probability of our input data, we need to learn the parameters π, μ_1, μ_2 and Σ. We can estimate the parameters by maximum likelihood, maximum a posteriori or bayesian learning. This example will demonstrate using maximum likelihood to learn these parameters.

We can express the Likelihood of our training set as $L(\boldsymbol{X}, \boldsymbol{y}) = P(\boldsymbol{X}, \boldsymbol{y}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma})$. We want to maximize the likelihood in order to use Bayes inference.

$$L(\boldsymbol{X}, \boldsymbol{y}) = \prod_n [\pi|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}]^{y_n} [(1-\pi)|N(\boldsymbol{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})]^{1-y_n}$$

Taking the log we can turn this into an optimization problem of finding

$$argmax_{\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}} \sum_n y_n [ln\pi - \frac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_1)]$$
$$+ (1 - y_n)[ln\pi - \frac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_2)]$$

Now to estimate π (probability of class) we can take:

$$0 = \frac{\partial lnL(X,y)}{\partial \pi}$$
$$\pi = \frac{\sum_n y_n}{N}$$

Now to estimate μ_1 and μ_2 (mean of classes) we can take:

$$0 = \frac{\partial lnL(X,y)}{\partial \mu_1}$$
$$\mu_1 = \frac{\sum_n y_n x_n}{N_1}$$

and

$$0 = \frac{\partial lnL(X,y)}{\partial \mu_2}$$
$$\mu_2 = \frac{\sum_n (1-y_n) x_n}{N_2}$$

Now to estimate Σ (covariance matrix) we can take:

$$0 = \frac{\partial lnL(X,y)}{\partial \Sigma}$$
$$\Sigma = \frac{N_1}{N} S_1 + \frac{N_2}{N} S_2$$

where S_k are the empirival covariance matrices of the class k

$$S_1 = \frac{1}{N_1} \sum_{n \in C_1} (x_n - \mu_1)(x_n - \mu_1)^T$$
$$S_2 = \frac{1}{N_1} \sum_{n \in C_2} (x_n - \mu_2)(x_n - \mu_2)^T$$

5 Linear Models

5.1 Linear Regression

5.1.1 Formulation

Linear Regression is a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. Our main objective is to generate a line that minimizes the distance from the line to all of data points. This is essentially minimizing the error and maximizing our prediction accuracy.

5.1.2 Simple Regression

A simple two variable linear regression uses the slope-intercept form, where m and b are the variables our algorithm will try to "learn". x represents our input data and y represents the prediction.

$$y = mx + b$$

5.1.3 Multivariable Regression

Often times there are more than one feature in the data and we need a more complex multi-variable linear equation as our hypothesis. We can represent our hypothesis with the follow multi-variable linear equation, where $\boldsymbol{w}$ are the weights and $\boldsymbol{x}$ is the input data.

$$\begin{aligned} h_{\boldsymbol{w}}(\boldsymbol{x}) &= w_0x_0 + w_1x_1w_2x_2 + ... + w_nx_n \\ &= \boldsymbol{w}^T\boldsymbol{x} \end{aligned}$$

5.1.4 Cost Function

To predict based on a dataset we first need to learn the weights that minimize the mean squared error (euclidean loss) of our hypothesis. We can define the following to be our cost function to minimize with m being the number of data points and i being the i^{th} training example. This can also be proven by applying maximum likelihood with gaussian noise. Similarly we can come up with the regularized least square problem by applying maximum a posteriori on noisy linear regression.

$$\begin{aligned} J(\boldsymbol{w}) &= \frac{1}{2m}\sum_{i=1}^{m}(h_{\boldsymbol{w}}(\boldsymbol{x}^i) - y^i)^2 \\ &= \frac{1}{2m}(X\boldsymbol{w} - \boldsymbol{y})^T(X\boldsymbol{w} - \boldsymbol{y}) \end{aligned}$$

5.1.5 Gradient Descent Solution

Now to solve for $\boldsymbol{w}$ we can use Gradient Descent and iteratively update $\boldsymbol{w}$ until it converges. We get the slope of the cost function to be:

$$\frac{\partial J\boldsymbol{w}}{\partial \boldsymbol{w}_j} = \frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{w}^T\boldsymbol{x}^i - y^i)\boldsymbol{x}_j^i$$

now applying a step α we can iteratively change $\boldsymbol{w}$ until it reaches the global minima.

$$\boldsymbol{w}_j := \boldsymbol{w}_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_{\boldsymbol{w}}(\boldsymbol{x}^i) - y^i)$$

5.1.6 Normal Equation Solution

The closed form solution to the linear system in $\boldsymbol{w}$

$$\frac{\partial J\boldsymbol{w}}{\partial \boldsymbol{w}_j} = \frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{w}^T\boldsymbol{x}^i - y^i)\boldsymbol{x}^i_j = 0$$

writing this as a linear system in w we get $A\boldsymbol{w} = b$ where

$$A = \sum_{n=1}^{N}(\boldsymbol{x}_n\boldsymbol{x}_n^T) \text{ and } b = \sum_{n=1}^{N}(\boldsymbol{x}_{\boldsymbol{n}}y_n)$$

so we can solve for $\boldsymbol{w} = \boldsymbol{A}^{-1}\boldsymbol{b}$ and get the following vectorized solution.

$$\boldsymbol{w} = (X^TX)^{-1}X^T\boldsymbol{y}$$

5.2 Logistic Regression

5.2.1 Formulation

Logistic regression is an algorithm used for classification. It is used to estimate the probability that an instance belongs to a particular class. If the estimated probability is greater than 50%, then the model predicts the instance belongs to that class, and otherwise it predicts it does not. Its motivation stems from probabilistic discriminative models except now we are learning $P(c_k|\boldsymbol{x})$ by max likelihood instead of learning $P(c_k)$ and $P(\boldsymbol{x}|c_k)$ by max likelihood and $P(c_k|\boldsymbol{x})$ by Bayesian Inference.

5.2.2 Prediction

Logistic Regression computers the weighted sum of the input features (plus a bias term) and outputs the logistic (Sigmoid Function) of the result. The hypothesis is given by

$$p = h_{\boldsymbol{w}}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x})$$

Once the Logistic Regression model has estimated the probability that an instance *boldsymbolx* belongs to the positive class, it can make its prediction y easily.

$$y = \begin{cases} 0 & p < 0.5 \\ 1 & p \geq 0.5 \end{cases}$$

5.2.3 Cost Function

The objective of training the model is such that the model estimates high probabilities for positive instances ($y = 1$) and low probabilities for negative instances ($y = 0$). This concept is captured through the cost function shown below.

$$J(\boldsymbol{w}) = \begin{cases} -log(p) & y = 1 \\ -log(1-p) & y = 0 \end{cases}$$

This makes intuitive sense because $-log(t)$ grows very large when t approaches 0, so the cost will be large if the model estimates a probability close to 0 for a positive instance. The cost will also be very large if the model estimates a probability close to 1 for a negative instance. On the other hand $-log(t)$ is close to 0 when t is close to 1, so the cost will be close to 0 if the estimated probability is close to 0 for a negative instance or close to 1 for a positive instance.

We can express the cost as a single expression called the *log loss.*

$$J(\boldsymbol{w}) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(h_{\boldsymbol{w}}(\boldsymbol{x}^{(i)})) + (1 - y^{(i)}log(1 - h_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}))]$$

5.2.4 Solution

Unfortunately, there is no known closed-form solution to compute the value of $\boldsymbol{w}$ that minimizes the cost function. The cost function however, is convex, so Gradient Descent or any other optimization algorithm is guaranteed to find the global minimum. The gradient can be expressed as:

$$\frac{\partial J\boldsymbol{w}}{\partial \boldsymbol{w}_j} = \frac{1}{m}\sum_{i=1}^{m}(\sigma(\boldsymbol{w}^T\boldsymbol{x}) - y^{(i)}x_j^{(i)})$$

Some faster more sophisticated methods are

- Conjugate Gradient
- BFGS
- L-BFGS

5.2.5 Softmax Regression

The Logistic Regression model can be generalized to support multiple classes. When given an instance $\boldsymbol{x}$, the Softmax Regression model computes a score $f_k(\boldsymbol{x})$ for each class k, then estimates the probability of each class by applying the *softmax function* to the scores.

$$f_k(\boldsymbol{x}) = \boldsymbol{w}_k^T\boldsymbol{x}$$

Once the score of every class for the instance $\boldsymbol{x}$ is computed, you can estimate the probability π_k that the instance belongs to class k. The function computes the exponential of each score, the normalizes them.

$$\pi_k = P(y^{(i)} = k|\boldsymbol{x}^{(i)}, \boldsymbol{w}) = \frac{e^{f_k(\boldsymbol{x})}}{\sum_{j=1}^{K} e^{f_j(\boldsymbol{x})}}$$

The Softmax Regression classifier predicts the class with the highest estimated probability.

The cost function associated with the Softmax Regression Classifier is the Cross Entropy cost function; it penalizes the model when it estimates a low probability for a target class. Cross entropy is used to measure how well a set of estimates class probabilities matches the target class. The cost function is represented as such

$$J(\boldsymbol{w}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} log(\pi_k^{(i)})$$

with gradient vector

$$\nabla_{w_k} J(\boldsymbol{w}) = \frac{1}{m}\sum_{i=1}^{m}(\pi_k^{(i)} - y_k^{(i)})\boldsymbol{x}^{(i)}$$

that can be paired with an optimization algorithm to solve.

5.3 Generalized Linear Models

Often times our data won't be linear and it could be of a higher degree polynomial or a completely different distribution altogether. We can turn this non-linear problem into a linear regression problem by mapping the data to a different vector space using a basis function.

To demonstrate, let us consider Linear Regression on a nonlinear N x 1 (feature) dataset. Let ϕ denote the polynomial basis function where $\phi_j(\boldsymbol{x}) = x^j$. Then we can express our hypothesis as:

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = w_0\phi(x) + w_1\phi_1(x) + w_2\phi_2(x) + ... + w_3\phi_3(x)$$

A dataset with 3 features with a polynomial basis would have a hypothesis as such

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 + w_5x_1^2x_2 + w_6x_1x_2^2 + w_7x_1^2x_2^2 + w_8x_1^3 + w_9x_2^3$$

This can then be extrapolated to logistic regression and n-features. Some commonly used basis functions are:

- Polynomial: $\phi_j(\boldsymbol{x}) = x^j$
- Gaussian: $\phi_j(\boldsymbol{x}) = e^{(\frac{x-\mu_j}{2s^2})}$
- Sigmoid: $\phi_j(\boldsymbol{x}) = \sigma(\frac{x-\mu_j}{s})$
- Fourier Basis, Wavelets, etc ...

5.4 Regularization

Small outliers can drastically change our values of $\boldsymbol{w}$ so rely on regularization to reduce overfitting. Polynomial models can be easily regularized by reducing the number of polynomial degrees. For a linear model, regularization is typically achieved by constraining the weights of the model. The regularization term should only be added to the cost function during training. Once the model is trained, the non-regularized cost should be used to measure the model's performance. The bias term w_0 is not regularized.

5.4.1 Ridge Regression

Ridge Regression (Tikhonov Regularization) is a regularized version of Linear regression with a regularization term of $\frac{\lambda}{2}\|w\|_2^2$ (l_2-norm) added to the cost function. This forces the learning algorithm to fit the data but also keep the model weights as small as possible. The hyperparameter λ controls how much you want to regularize the model.

$$J(\boldsymbol{w}) = ERROR(\boldsymbol{w}) + \frac{\lambda}{2}\|w\|_2^2$$

5.4.2 Lasso Regression

Least Absolute Shrinkage and Selection Operator Regression is another regularized version of Linear Regression, it adds a regularization term to the cost function but uses the l_1 norm of the weight vector instead of half the square of the l_2 norm.

$$J(\boldsymbol{w}) = ERROR(\boldsymbol{w}) + \lambda \sum_{i=1}^{n} |w_i|$$

An important characteristic of Lasso Regression is that it tends to eliminate the weights of the least important features (i.e, set them to zero). Lasso Regression automatically performs feature selection and outputs a *sparse model*.

5.4.3 Elastic Net

Elastic Net is a middle ground between Ridge Regression and Lasso Regression. The regularization term is a simple mix of both Ridge and Lasso's regularization terms and you can control the mix ratio r. When $r = 0$, Elastic Net is equivalent to Ridge Regression, and when $r = 1$, it is equivalent to Lasso Regression.

$$J(\boldsymbol{w}) = ERROR(\boldsymbol{w}) + \frac{(1-r)\lambda}{2}\|w\|_2^2 + r\lambda \sum_{i=1}^{n} |w_i|$$

5.4.4 Early Stopping

Early Stopping is a different way to regularize iterative learning algorithms such as Gradient Descent. This method aims to stop training as soon as the validation error reaches a minimum. For all convex optimization problems there will be a global minima, once that global minima is reached the curve will start going up. This proposes to stop as soong as we reach the minimum.

Index