Machine Learning Basics

Carlos Valle

Departamento de Informática
Universidad Técnica Federico Santa María

cvalle@inf.utfsm.cl

September 14, 2018

Overview

What is Machine Learning?

Arthur Samuel

"Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed"

Example: Dog recognizer

- Dog or no dog?

Example: Dog recognizer (2)

- Dog or no dog?

Example: Dog recognizer (3)

- Dog or no dog?

Training versus programming

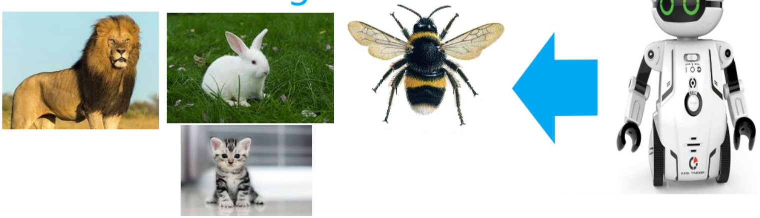
Algorithm 1 Dog or no dog pseudocode

if $four\_legs \,\&\, whiskers$ **then**
    return dog
else
    return no dog
end if

Training versus programming

Training versus programming

No dogs



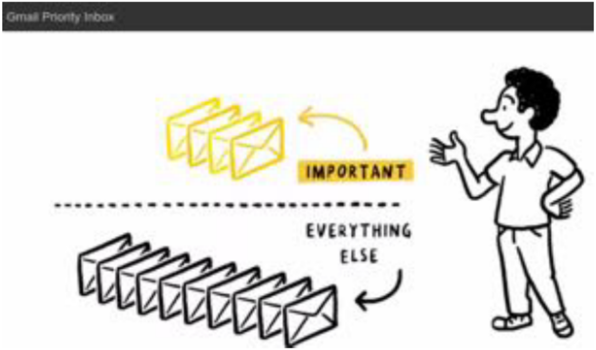
What is Machine Learning?

Tom Mitchell

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"

Example

- Spam Detection



Example (2)

- Spam Detection

Example (3)

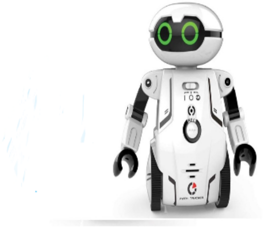
- Spam Detection

NO SPAM

Example (4)

- Spam Detection

?



- Suppose we have a dataset giving weight, gender and calorie consumption a day from 40 people.

Weight (Kg)	Gender (M/F)	calorie cons. (cal)
85	M	2075
58	F	1757
52	M	2783
55	F	3500
⋮	⋮	⋮

- We would like to predict the calorie consumption per day of other people.
- In this case, weight and gender are called input features. Each vector input $\mathbf{x}_m$ has these two features.
- A calorie consumption per day y_m is called target.
- A pair $(\mathbf{x}_m, y_m)$ is called a train example. A train set is a group of M training examples $S_M = \{(\mathbf{x}_m, y_m)\}, m = 1 \ldots M$.

Preliminary definitions

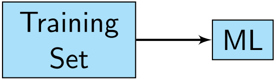
- Features can be either numerical or categorical.
- Let $\mathcal{X}$ be the feature space, and $\mathcal{Y}$ the output space.
- Our goal is to obtain a mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$, commonly called the *hypothesis* or learner), $\mathcal{X} \subseteq \mathbb{R}^n$, $y \in Y \subseteq \mathbb{R}$.
- Let $\mathcal{S}$ the space that spans the possible samples, drawn from an unknown distribution $P(\mathbf{x}, y)$.
- A *learning algorithm* is a map from the space of train sets to the hypothesis space $\mathcal{H}$ of possible functional solutions

$$\begin{aligned} \mathcal{A} \quad : \quad & \mathcal{S} \rightarrow \mathcal{H} \\ & S_M \rightarrow \mathcal{A}(S_M) = f. \end{aligned} \tag{1}$$

Learning process

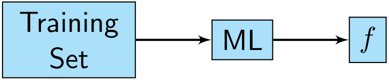
- If the target is continuous, we have a regression problem
- If the target can take a finite number k of discrete values, we have a classification problem. In particular $k = 2$ the problem is called binary classification.

Learning process



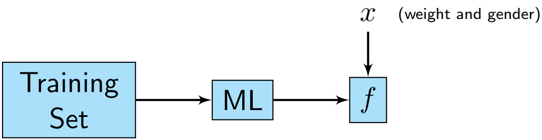
- If the target is continuous, we have a regression problem
- If the target can take a finite number k of discrete values, we have a classification problem. In particular $k = 2$ the problem is called binary classification.

Learning process



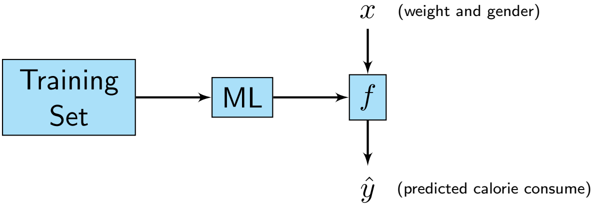
- If the target is continuous, we have a regression problem
- If the target can take a finite number k of discrete values, we have a classification problem. In particular $k = 2$ the problem is called binary classification.

Learning process



- If the target is continuous, we have a regression problem
- If the target can take a finite number k of discrete values, we have a classification problem. In particular $k = 2$ the problem is called binary classification.

Learning process



- If the target is continuous, we have a regression problem
- If the target can take a finite number k of discrete values, we have a classification problem. In particular $k = 2$ the problem is called binary classification.

Learning process (2)

- A main challenge is to construct an automatic method or algorithm able to estimate future examples based on the observed phenomenon in the train set.
- This key property of an algorithm is known as the generalization ability.
- The algorithms that memorize the train samples but have poor predictive performance with unknown examples, this undesirable problem is well-known as overfitting.

Loss function

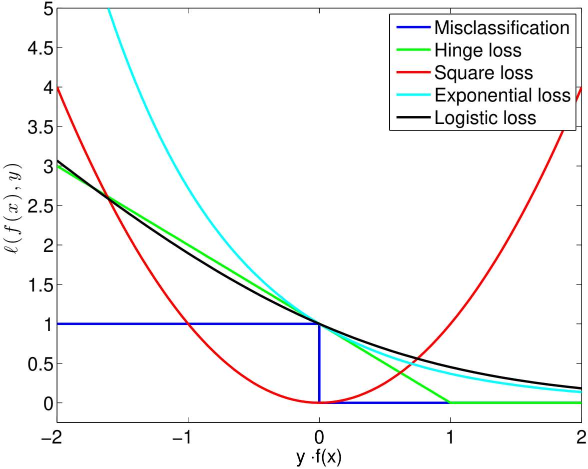
- The quality of the algorithm $\mathcal{A}$ is measured by the loss function given by $\ell : \mathbb{R} \times \mathcal{Y} \to [0, \infty)$, which quantifies the accuracy of the observed response $f(\mathbf{x})$ with respect to the true or desired response y.
- This function does not penalize the exact predictions, i.e., $\ell(y, f(\mathbf{x})) = 0$ if and only if $y = f(\mathbf{x})$.
- ℓ is a non-negative function, hence, the hypothesis will never profit from additional good predictions.
- In regression settings we use the quadratic loss function $\ell(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$
- While in classification we have the misclassification loss function

$$\ell(f(\mathbf{x}), y) = \begin{cases} 0 & \text{if } f(\mathbf{x}) = y \\ 1 & \text{if } f(\mathbf{x}) \neq y \end{cases}.$$

Loss function (2)

- However, In binary classification problem (where $y \in \{-1, 1\}$), the margin $yf(\mathbf{x})$ is introduced as a quality measure.
- This quality amount leads to several loss functions such as the *hinge loss*
$$\ell(f(\mathbf{x}), y) = \max(1 - yf(\mathbf{x}), 0) = |1 - yf(\mathbf{x})|_+.$$
- The *logistic loss* $\ell(f(\mathbf{x}), y) = \log_2 \left(1 + e^{-yf(\mathbf{x})}\right)$
- The *exponential loss* $\ell(f(\mathbf{x}), y) = e^{-yf(\mathbf{x})}$.
- Note that the square loss can be arranged as $\ell(f(\mathbf{x}), y) = (y - f(\mathbf{x}))^2 = (1 - yf(\mathbf{x}))^2$ taking into account that $y^2 = 1$.
- And the misclassification loss can be written as $\ell(f(\mathbf{x}), y) = I(yf(\mathbf{x}) < 0)$, where $I(\cdot)$ is the indicator function.

Loss function (3)



- Logistic Loss and exponential loss can be viewed as a continuous approximation of the misclassification function

Motivation: Linear Regression

- We want to predict the total daily travel time of a trucker considering both the distance traveled and the number of deliveries made as input features:

Distance travel time (km)	Number of deliveries	Travel time (hr)
50	4	8.4
75	10	12.3
34	3	6.5
62	5	10.0
$\vdots$	$\vdots$	$\vdots$

- Here the inputs $\mathbf{x}$ are bi-dimensional vectors. We let $\mathbf{x}_m^{(i)}$ denote the feature i of the m-th example.

Linear model

- We could approximate the total daily travel time y as a linear function of the distance traveled and the number of deliveries made:

$$f(x) = \beta_0 + \beta_1 x^{(1)} + \beta_2 x^{(2)},$$

- where $\beta_i, i = 0, 1, 2$ are the parameters of the linear model.
- Thus, the space of linear functions mapping from $\mathcal{X}$ to $\mathcal{Y}$ is parametric.
- We define $x^{(0)} = 1$ to define the linear model in matrix form:

$$f(x) = \sum_{i=0}^{I} \beta_i x^{(i)} = \beta^T \mathbf{x}, \tag{2}$$

- where I is the number of features.

How do we select the β 's?

- We can get the parameters of the model by minimizing the quadratic loss funcion over the train set:

$$J(\beta) = \frac{1}{2} \sum_{m=1}^{M} \left(f\left(\mathbf{x}_m\right) - y_m\right)^2 \tag{3}$$

- where $\beta = (\beta_0, \beta_1, \ldots, \beta_I)^T$.
- We need to choose β which minimizes $J(\beta)$.

- This problem can be expressed in matrix form:

$$J(\beta) = \frac{1}{2}(Y - X\beta)^T (Y - X\beta),$$

- where X is an $N \times (I+1)$ matrix.

From matrix calculus

- If x is a column vector:
- $\frac{\partial u^T v}{\partial x} = \frac{\partial u}{\partial x} \cdot v + \frac{\partial v}{\partial x} \cdot u$
- $\frac{\partial Ax}{\partial x} = A^T$

LMS Algorithm (2)

- Then,

$$
\begin{aligned}
\frac{\partial J(\beta)}{\partial \beta} &= -X^T(Y - X\beta) \\
\frac{\partial^2 J(\beta)}{\partial \beta \partial \beta^T} &= X^T X
\end{aligned}
$$

- Equalizing the first derivative to zero we get the normal equations:

$$X^T(Y - X\beta) = 0 \Longrightarrow \widehat{\beta} = (X^T X)^{-1} X^T Y$$

Bias of the LMS Algorithm

- Assuming that the linear model is correct, i.e., $Y = X\beta + \epsilon$ for some unknown β. Furthermore, we assume that $E[\epsilon] = 0$ and $\text{Cov}(\epsilon) = E[\epsilon\epsilon^T] = \sigma^2 I$ (uncorrelated noise). From the least squares solution $\beta_{ls} = \left(X^T X\right)^{-1} X^T Y$. Thus

$$E\left[\beta_{ls}\right] = E\left[\left(X^T X\right)^{-1} X^T Y\right] \tag{4}$$

$$= \left(X^T X\right)^{-1} X^T E\left[Y\right] = \left(X^T X\right)^{-1} X^T X\beta = \beta\,, \tag{5}$$

- That is, β_{ls} is an unbiased estimator of β.
- Furthermore

$$\begin{aligned}
\beta_{ls} - E\left[\beta_{ls}\right] &= \left(X^T X\right)^{-1} X^T Y - \beta \\
&= \left(X^T X\right)^{-1} X^T Y - \left(X^T X\right)^{-1} X^T X\beta \\
&= \left(X^T X\right)^{-1} X^T \left(Y - X\beta\right) \\
&= \left(X^T X\right)^{-1} X^T \epsilon
\end{aligned} \tag{6}$$

Variance of the LMS Algorithm

- Thus, from the assumption $E[\epsilon\epsilon^T] = \sigma^2 I$ we obtain

$$\begin{aligned}\text{Cov}(\beta_{ls}) &= E\left[\left(\beta_{ls} - E\left[\beta_{ls}\right]\right)\left(\hat{\beta}^{\text{ls}} - E\left[\beta_{ls}\right]\right)^T\right] \\ &= E\left[\left(X^T X\right)^{-1} X^T \epsilon\epsilon^T X \left(X^T X\right)^{-1}\right] \\ &= \left(X^T X\right)^{-1} X^T E\left[\epsilon\epsilon^T\right] X \left(X^T X\right)^{-1} \\ &= \sigma^2 \left(X^T X\right)^{-1} X^T X \left(X^T X\right)^{-1} \\ &= \sigma^2 \left(X^T X\right)^{-1}\end{aligned} \tag{7}$$

- Typically the variance σ^2 is estimated as

$$\hat{\sigma}^2 = \frac{1}{N - I - 1} \sum_{m=1}^{M} \left(y_m - \hat{y}_m\right)^2 .$$

Gradient descent algorithm

- Let consider the gradient descent algorithm which start with some initial β and repeatedly performs:

$$\beta_i^{p+1} = \beta_i^p - \alpha \frac{\partial}{\partial \beta_i} J(\beta)$$

- Let's calculate the derivative of the loss function for a single train example $(\mathbf{x}_m, y_m)$:

$$\begin{aligned}
\frac{\partial}{\partial \beta_i} J(\beta) &= \frac{\partial}{\partial \beta_i} \frac{1}{2} \left(f\left(\mathbf{x}_m\right) - y_m \right)^2 \\
&= \left(f\left(\mathbf{x}_m\right) - y_m \right) \cdot \frac{\partial}{\partial \beta_i} \left(\sum_{i=0}^{I} \beta_i x^{(i)} \right) \\
&= \left(f\left(\mathbf{x}_m\right) - y_m \right) \mathbf{x}_m^{(i)}
\end{aligned}$$

Gradient descent algorithm (2)

- Then, for a single example:

$$\beta_i^{p+1} = \beta_i^p - \alpha(f(\mathbf{x}_m) - y_m)\,\mathbf{x}_m^{(i)}$$

- This rule is called Widrow-Hoff learning rule.
- Note that the amount of the update is proportional to the error: $(f(\mathbf{x}_m) - y_m)$

Batch gradient descent algorithm

- We can apply the learning rule above for the train set:

Algorithm 2 Batch gradient descent algorithm

1: **repeat**
2: $\beta_i^{p+1} = \beta_i^p - \alpha \sum_{m=1}^{M} (f(\mathbf{x}_m) - y_m)\, \mathbf{x}_m^{(i)}$ (for every i)
3: **until** Convergence

- In general, gradient descent can reach a local minimum.
- However, J is a convex function. Thus, the optimization problem has only one global optimum.

Stochastic gradient descent algorithm

- We can modify the learning rule above for each example:

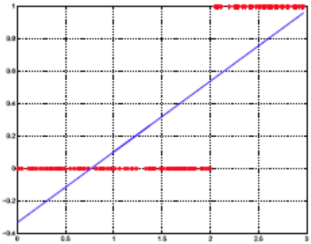
Algorithm 3 Stochastic gradient descent algorithm

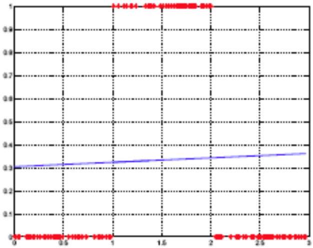
```
1: repeat
2:     for m := 1 to M do
3:         β_i^{p+1} := β_i^p − α(f(x_m) − y_m) x_m^{(i)}   (for every i)
4:         β_i^p := β_i^{p+1}
5:     end for
6: until Convergence
```

- This method is called stochastic or online gradient descent.
- Usually, this technique converge faster than batch gradient descent.
- However, using a fixed value for α it may never converge to the minimum of $J(\beta)$, oscillating around it.
- To avoid this behavior, it is recommended to slowly decrease α to zero along the iterations.

Why don't use a linear regression algorithm?

- Linear regression models might mask some classes.

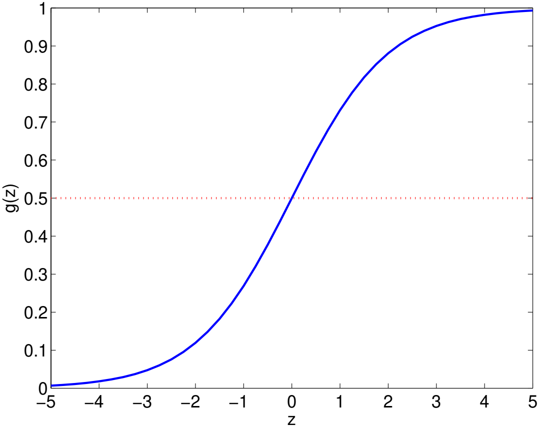




Logistic regression

- It measures the relationship between the class and the input vector by estimating probabilities using a logistic function:
- $$f_\beta(\mathbf{x}) = g(\beta^T\mathbf{x}) = \frac{1}{1+e^{-\beta^T\mathbf{x}}},$$
- where
$$g(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z}$$
is the logistic or sigmoid function
- As before, $\beta^T\mathbf{x} = \beta_0 + \sum_{i=1}^{I}\beta_i x(i)$
- However, it is worth mentioning that $f_\beta(\mathbf{x})$ is not linear.

Logistic function



- When $z = 0 = \beta^T x$, $g(z)$ is on the decision threshold 0.5.
- When $z \to \infty$, $g(z) \to 1$.
- When $z \to -\infty$, $g(z) \to 0$.

Logistic function (2)

- The derivative of $g(z)$ is
- $$\begin{aligned} g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\ &= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\ &= \frac{1}{(1+e^{-z})}\cdot\left(1-\frac{1}{(1+e^{-z})}\right) \\ &= g(z)(1-g(z)) \end{aligned}$$

How to obtain β?

- Let us assume that

$$\begin{aligned} P(y=1|\mathbf{x};\beta) &= f_\beta(\mathbf{x}) \\ P(y=0|\mathbf{x};\beta) &= 1-f_\beta(\mathbf{x}) \end{aligned}$$

- Thus $p(y|\mathbf{x};\beta) = (f_\beta(\mathbf{x}))^y(1-f_\beta(\mathbf{x}))^{1-y}$

How to obtain β? (2)

- Assuming that the M training example are independent, we can express the likelihood function as:

$$\begin{aligned} L(\beta) &= \prod_{m=1}^{M} p(y_m|\mathbf{x}_m;\beta) \\ &= \prod_{m=1}^{M} (f_\beta(\mathbf{x}_m))^{y_m}(1-f_\beta(\mathbf{x}_m))^{1-y_m} \end{aligned}$$

- As we did before, we maximize the log likelihood function:

$$\begin{aligned} \ell(\beta) &= \sum_{m=1}^{M} y_m \log f_\beta(\mathbf{x}_m) + (1-y_m)\log(1-f_\beta(\mathbf{x}_m)) \\ &= \sum_{m=1}^{M} y_m \beta^T \mathbf{x}_m - \log(1+e^{\beta^T \mathbf{x}_m}) \end{aligned} \quad (8)$$

Stochastic gradient ascent

- Now we can use gradient descent algorithm:

$$\beta^{p+1} = \beta^p + \alpha \nabla_\beta \ell(\beta)$$

- Let's obtain the derivative:

$$\begin{aligned}
\frac{\partial}{\partial \beta_i}\ell(\beta) &= \left(y\frac{1}{f_\beta(\mathbf{x})} - (1-y)\frac{1}{1-f_\beta(\mathbf{x})} \right) \frac{\partial}{\partial \beta_i} f_\beta(\mathbf{x}) \\
&= \left(y\frac{1}{f_\beta(\mathbf{x})} - (1-y)\frac{1}{1-f_\beta(\mathbf{x})} \right) f_\beta(\mathbf{x})(1-f_\beta(\mathbf{x})) \frac{\partial}{\partial \beta_i} \beta^T \mathbf{x} \\
&= \left(y(1-f_\beta(\mathbf{x})) - (1-y) f_\beta(\mathbf{x}) \right) x^{(i)} \\
&= \left(y - f_\beta(\mathbf{x}) \right) x^{(i)}
\end{aligned}$$

- Thus, the stochastic gradient ascent rule is

$$\beta_i^{p+1} = \beta_i^p + \alpha \left(y - f_\beta(\mathbf{x}) \right) x^{(i)}$$

How to select a model in practice?

- How to automatically choose the parameters of the model?
- We will assume we have a finite set of models $\mathcal{M} = \{M_1, \ldots, M_d\}$
- In case that the parameter(s) is (are) continuous we can discretize it (them).
- As we stated before, choosing the parameters which minimizes the train error does not guarantee generalization.

Hold-out cross validation

- One option is to do the following:

Algorithm 4 Hold-out cross validation

1: Randomly split S into S_{train} (For instance 75% of the data) and S_{cv} (the remaining 25%). Where S_{cv} is the hold-out cross validation set.
2: Train each model M_i on S_{train} only to get the hypothesis f_i.
3: Select and output the hypothesis f_i with the smallest error $\hat{\varepsilon}_{S_{CV}}(f_i)$ on the hold out cross validation set.

- Thus, for each f_i we obtain a better estimation of the generalization error.
- And we can select the hypothesis with the minimum estimated generalization error.
- Usually the size of the hold out cross validation set is set between $1/4$ and $1/3$.
- We are losing data that we are not using for training. And we estimate the generalization error only from one cross validation set.

K-fold cross validation

Algorithm 5 K-fold cross validation

Randomly split S into K disjoint subsets of M/K training examples each. Let's call these subsets $S_1, \ldots, S_K$.
for each model M_i **do**
    for $j = 1, \ldots, K$ **do**
        $f_{ij} \leftarrow \mathcal{A}_i(S \setminus S_j)$ (train on all data except S_j)
        $\hat{\varepsilon}_{S_j}(f_{ij}) = \frac{1}{|S_j|} \sum_{(\mathbf{x}_m, y_m) \in S_j} \ell(y_m, f_{ij}(\mathbf{x}_m))$ (Compute the error over the validation test)
    end for
    $\hat{\varepsilon}_{M_i} = \frac{1}{k} \sum_{k=1}^{K} \hat{\varepsilon}_{S_k}(f_{ik})$ (Calculate the average of the validation error over the k folds)
end for
Pick the model M_i with the lowest $\hat{\varepsilon}_{M_i}$
$f \leftarrow \mathcal{A}_i(S)$ (train a hypothesis with all training data according to the model M_i).
Output: f

- $\mathcal{A}_i$ is the learning algorithm according to the model M_i

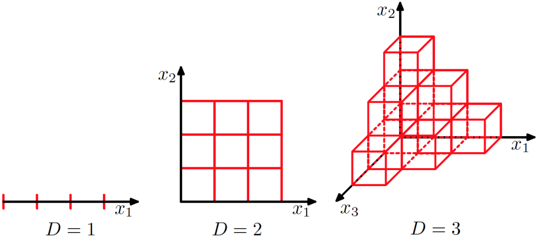
K-fold cross validation (2)

- Usually $K = 5$ or $K = 10$.
- When $K = M$ this method is called leave-one-out cross validation.
- In classification problems, sometimes the proportion of examples of each class are unequal. In this case, we can adapt cross validation in such a way that each fold has the same proportions. Thus, all folds will be equally unbalanced.
- This method is called stratified K-fold cross validation.

Curse of dimensionality

- In real applications of machine learning, we have to deal with input spaces of high dimensionality comprising many inputs variables.
- Consider the problem of labeling a point according to the labels of its neighbors.
- To tackle this problem, we will divide the input space in cells, and we will label the point as the class having the largest number of training points in the same cell.
- This approach has a serious problem when x is a high-dimensional vector.
- If we divide a region of a space into regular cells, then the number of such cells grows exponentially with the dimensionality of the space.
- Therefore, with an exponentially large number of cells we would need an exponentially large amount of training examples in order to ensure that the cells are not empty.

Curse of dimensionality (2)



- In regression problems we would like to model the problem by considering the dependencies among the input features:

$$f(\mathbf{x}, \beta) = \beta_0 + \sum_{i=1}^{I} \beta_i \mathbf{x}^{(i)} + \sum_{i=1}^{I}\sum_{j=1}^{I} \beta_{ij} \mathbf{x}^{(i)} \mathbf{x}^{(j)} + \sum_{i=1}^{I}\sum_{j=1}^{I}\sum_{k=1}^{I} \beta_{ijk} \mathbf{x}^{(i)} \mathbf{x}^{(j)} \mathbf{x}^{(k)}$$

Curse of dimensionality (3)

- When I increases, the number of coefficients of β grows proportionally to I^3.
- Note that we only considered up to 3 attributes. Thus, this method is not scalable.
- Now, consider a sphere of radius $r = 1$ in a space of I dimensions.
- The fraction of the volume of the sphere that lies between $r = 1 - \varepsilon$ and $r = 1$ is

$$\frac{V_I(1) - V_I(1-\varepsilon)}{V_I(1)} = 1 - (1-\varepsilon)^I,$$

- Here, $V_I(r) = K_I r^I$, where K_I is a constant that only depends on I.
- Thus, for high-dimensional spaces, the volume of the sphere is concentrated near to the surface.

Curse of dimensionality (4)

- Fortunately we can develop effective algorithm for high-dimensional data.
- Real Data is usually confined in a subset of the whole input space. So, we have lower effective dimensionality.
- Real data exhibits local smoothness properties. Hence, small changes in the input variables produces small changes in the target variables.
- We can exploit local interpolation-like techniques to predict the target of the new instances.

Bias-variance tradeoff

- A main challenge is to construct a learning algorithm able to extrapolate.
- That is, to estimate future examples based on the observed phenomenon in the train set. (generalization error)
- This key property of an algorithm is known as the generalization ability.
- On the other hand, we find the algorithms that memorize the training samples but have poor predictive performance with unknown examples, this undesirable problem is well-known as overfitting.
- If we compute a different model for each training sample that we have collected; the bias is the set of points that cannot be well predicted by the model, and the variance measures how different the predictions along the training samples are.

Bias-variance tradeoff (2)

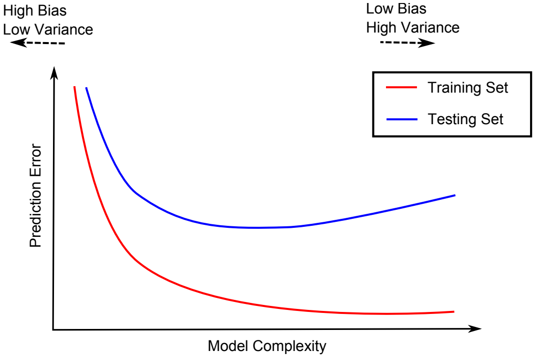
- Mathematically,

$$
\begin{aligned}
E[(y - f(\mathbf{x}))^2] &= E[((y - E[f(\mathbf{x})]) + (E[f(\mathbf{x})] - f(\mathbf{x})))^2] \\
&= E[(y - E[f(\mathbf{x})])]^2 + E[2(y - E[f(\mathbf{x})])(E[f(\mathbf{x})] - f(\mathbf{x}))] \\
&\quad + E[(E[f(\mathbf{x})] - f(\mathbf{x}))^2] \\
&= E[(y - E[f(\mathbf{x})])]^2 + E[(E[f(\mathbf{x})] - f(\mathbf{x}))^2] \\
\text{MSE} &= \text{bias}^2(f) + \text{var}(f)
\end{aligned}
$$

- From the machine learning point of view, this trade-off is strongly related with the complexity of the learner f.
- A learner with low complexity has high bias covering the training points, which can lead to underfitting.

Bias-variance tradeoff (3)

- While, if the complexity of the learner is too high, the prediction tends to be closer (lower bias) to the training data and consequently generates overfitting.



Ridge regression

- It shrinks the coefficient values by introducing a regularization term,

$$\widehat{\beta}^{ridge} = \text{argmin}_{\beta} \left\{ \sum_{m=1}^{M} (y_m - \beta_0 - \sum_{i=1}^{I} x_{mi}\beta_i)^2 + \lambda \sum_{i=1}^{I} \beta_i^2 \right\} \quad (9)$$

- $\lambda \geq 0$ controls the regularization level
- Equivalently,

$$\begin{aligned} \widehat{\beta}^{ridge} &= \text{argmin}_{\beta} \sum_{m=1}^{M} \left(y_m - \beta_0 - \sum_{i=1}^{I} x_{mi}\beta_i \right)^2 \\ \text{s.a.} & \quad \sum_{i=1}^{I} \beta_i^2 \leq s \end{aligned}$$

- It is dependent of the scale of data. Thus, normalizing data is strictly necessary.
- β_0 is not bounded.

Reparametrizing

- We can transform data by using $x_{mi} \leftarrow (x_{mi} - \overline{x}_i)$
- Thus, we estimate $\beta_0 = \overline{y} = \sum_{m=1}^{M} y_m / M$
- Now X has I columns. Then, equation (9) can be expressed as

$$RSS(\lambda) = \frac{1}{2}((Y - X\beta)^T(Y - X\beta) + \lambda\beta^T\beta)$$

- The solution of this equation is

$$\widehat{\beta}^{ridge} = (X^T X + \lambda I)^{-1} X^T Y$$

- Now the problem is non-singular.
- For orthonormal inputs we have, $\widehat{\beta}_{ridge} = \gamma\widehat{\beta}_{LS}, 0 \leq \gamma \leq 1$

Any questions?