

# CIS 526: Machine Learning

## Lecture 2 (Sep 09, 2003)

prepared by Vladimir Vacic

### Standard Accuracy Measures

In order to assess accuracy of a predictor, we need to define a **loss function**  $L(y, f(\mathbf{x}))$ , where  $y$  is true output, and  $f(\mathbf{x})$  is the prediction. Loss function is sometimes denoted with  $R$  (risk function).

Given a **predictor**  $f(\mathbf{x})$  its **accuracy** is defined as  $E_D[L(y, f(\mathbf{x}))]$ , where  $E$  is expectation and  $D$  is the distribution. In words,  $E_D[L(y, f(\mathbf{x}))]$  is an expected loss over an example randomly taken from an underlying distribution.

Having the notion of an accuracy measure, we can refine the goal of supervised learning – the goal is to **minimize prediction loss**. Accuracy measures for regression and classification are different.

#### Regression

Most popular loss function for regression is  $L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$ , known as **squared loss**. Accuracy of a predictor in that case is called **Mean Squared Error** (MSE),

$$MSE = E_D[(y - f(\mathbf{x}))^2] = \int_D (y - f(\mathbf{x}))^2 \cdot p(\mathbf{x}, y) \cdot d\mathbf{x} \cdot dy$$

Math note: For a scalar  $x$  and any function  $f$ ,  $E[f(x)] = \int_{x \in R} (x) \cdot p(x) \cdot dx$ , where  $p(x)$  is the probability density function (pdf).

In general, a loss function can be arbitrarily defined, although typically loss is zero for correct prediction, and larger than zero if the prediction was not correct.

In practice, the available data set is finite. Assuming there are  $N$  examples MSE can be estimated as

$$MSE \approx \frac{1}{N} \sum_{i=1}^N (y - f(\mathbf{x}))^2, \text{ note that } MSE = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{i=1}^N (y - f(\mathbf{x}))^2$$

## Classification

In general,  $L(y, f(\mathbf{x})) = l_{i,j}$ ,  $y \in C_i$ ,  $f(\mathbf{x}) \in C_j$ , assuming that we have several distinct types of objects, i.e.,  $K$  possible classes:  $y \in \{C_1, C_2, C_3, \dots, C_K\}$

In practice, **0-1 (zero-one) loss** is typically used:  $L(y, f(\mathbf{x})) = \begin{cases} 1, & y \neq f(\mathbf{x}) \\ 0, & y = f(\mathbf{x}) \end{cases}$

Examples: Suppose we are testing a blood sample for a certain disease. Thus, there are two classes,  $y \in \{0, 1\}$ , where  $y = 1$  means that a person has the disease, and  $f(\mathbf{x}) = 1$  means that we predicted that the person has the disease. The 0-1 loss function is defined as follows:

		$f(\mathbf{x})$	
		0	1
$y$	0	0	1
	1	1	0

What is the price of a mistake? How much would be the cost of a specific kind of a mistake? We may wish to penalize specific kind of mistakes differently:

		$f(\mathbf{x})$	
		0	1
$y$	0	0	100
	1	$10^6$	0

If the cost of a mistake is high, we might want to design a very conservative predictor which predicts disease if there is a slightest chance for it.

## Optimal predictors

### Regression

The optimal prediction function  $f(\mathbf{x})$  minimizes MSE:

$$MSE = \int_D (y - f(\mathbf{x}))^2 \cdot p(\mathbf{x}, y) \cdot d\mathbf{x} \cdot dy = \int_X \left( \int_{Y|X} (y - f(\mathbf{x}))^2 \cdot p(y | \mathbf{x}) \cdot dy \right) \cdot p(\mathbf{x}) \cdot dx$$

Therefore, for any given input  $\mathbf{x}$  we want to minimize

$$g(A) = \int (y - A)^2 \cdot p(y | \mathbf{x}) \cdot dy,$$

where with  $A = f(\mathbf{x})$  (remember that in the inner integral  $\mathbf{x}$  is fixed, so it can be considered as a constant).

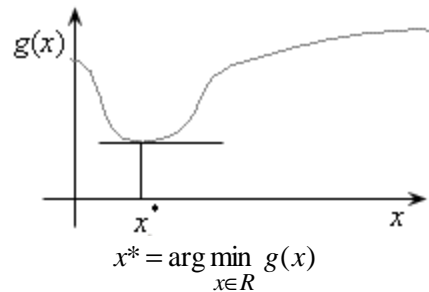
Regression problem is now reduced to:

*Find an optimal value of  $A$  that minimizes the integral  $g(A)$ .*

## Math Background:

### Unconstrained optimization

Problem: Given a function  $g(\mathbf{x})$  find its minimum (i.e. minimize  $g(\mathbf{x})$ )



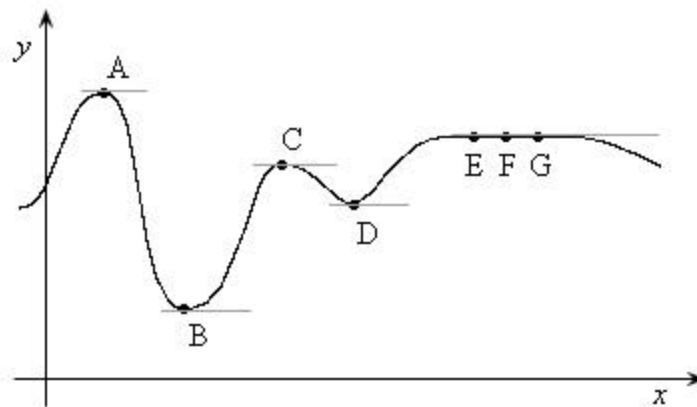
**Necessary** condition for a data point  $\mathbf{x}^*$  to be a minimum:

$$\frac{\partial g(\mathbf{x}^*)}{\partial x} = 0, \text{ for scalars, } \nabla g(\mathbf{x}^*) = 0, \text{ for vectors (Jacobian, J)}$$

Note: if  $\mathbf{x} = [x_1, x_2, \dots, x_n]$ , then  $\nabla g(\mathbf{x}) = \begin{bmatrix} \frac{\partial g(\mathbf{x})}{\partial x_1} \\ \vdots \\ \frac{\partial g(\mathbf{x})}{\partial x_n} \end{bmatrix}$

What is the **sufficient** condition for a data point  $\mathbf{x}^*$  to be a minimum?

Example:



If we observe the figure we notice that several points satisfy the necessary condition  $\nabla g(\mathbf{x}^*) = 0$ : point A is the global maximum, point B is the global minimum, point C is a local maximum, point D is a local minimum, points E, F, G are saddle points.

Sufficient condition for a data point  $\mathbf{x}^*$  to be a minimum:

$$\frac{\partial^2 g(x)}{\partial x^2} > 0, \text{ for scalars, } \nabla^2 g(\mathbf{x}) \text{ is a positive definite, for vectors (Hessian, H)}$$

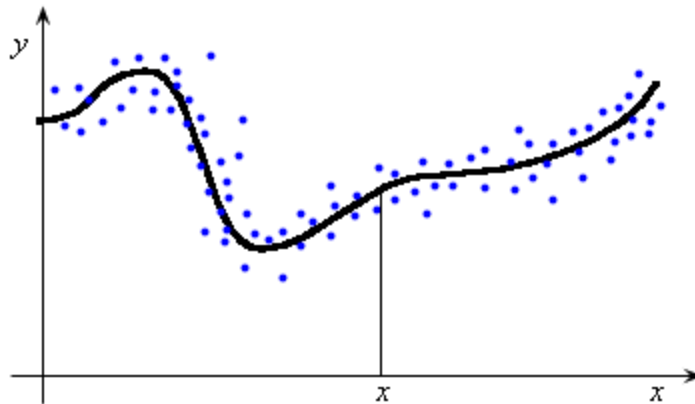
$$\text{Note: Hessian is defined as } H = \{h_{ij}\}, h_{ij} = \frac{\partial^2 g(\mathbf{x})}{\partial x_i \cdot \partial x_j}$$

## Back to Regression

To find minimum of  $g(A)$ :

$$\frac{\partial g(A)}{\partial A} = 0 \Rightarrow \int y \cdot p(y|\mathbf{x}) \cdot dy = \int A \cdot p(y|\mathbf{x}) \cdot dy \Rightarrow E[y|\mathbf{x}] = A$$

Therefore, the optimal regression function that minimizes MSE is:  $f^*(\mathbf{x}) = E[y|\mathbf{x}]$



$E[y|\mathbf{x}]$ , the optimal function that minimizes the error, is called the **regression function** (hence the term regression).

**Note:** The optimal solution is a theoretical result. We still have to learn how to estimate  $E[y|\mathbf{x}]$  as good as possible given a finite data set  $D$ .

## Classification

The optimal prediction function  $f(\mathbf{x})$  minimizes the classification loss:

$$\text{Loss} = \int L(y, f(\mathbf{x})) \cdot p(y|\mathbf{x}) \cdot p(\mathbf{x}) \cdot d\mathbf{x} \cdot dy = \int \left( \int_{Y|X} L(y, f(\mathbf{x})) \cdot p(y|\mathbf{x}) \cdot dy \right) \cdot p(\mathbf{x}) \cdot d\mathbf{x}$$

Therefore, to minimize Loss we need to minimize the inner integral for any given  $\mathbf{x}$ .

Note that for classification  $y$  is a discrete variable. Therefore, the inner integral is actually a sum

$$\sum_{i=1}^K L(y \in C_i, f(x)) \cdot P(y \in C_i | \mathbf{x}).$$

**Notation:**  $P(y \in C_i | \mathbf{x})$  is called **posterior class probability**, or **class conditional probability**,  $P(y \in C_i)$  is called **prior class probability**

Therefore, given an input  $\mathbf{x}$ , and by denoting  $A = f(\mathbf{x})$ , the optimal classification is class  $A$  that

minimizes  $g(A) = \sum_{i=1}^K L(y \in C_i, A) \cdot P(y \in C_i | \mathbf{x})$ , i.e.,

$$A = \arg \min_A \sum_{i=1}^K L(y \in C_i, A) \cdot P(y \in C_i | \mathbf{x}) \quad (\text{Eq.1})$$

**Note:** In a special case with 0-1 loss,  $g(A) = 1 - P(y \in A | \mathbf{x})$ . Therefore, the optimal decision is the class with the highest posterior!!

**Conclusion:** If posteriors  $P(y \in C_j | \mathbf{x})$ ,  $j = 1, 2, 3, \dots, K$ , are known then the optimal classification is easily calculated

**Terminology:** The optimal classification function is called the **Bayes classifier**, and the (Eq.1) is called the **Bayes classification (decision) rule**.

## Relationship between Conditional Expectation $E[y|\mathbf{x}]$ and Posteriors $P(y \in C | \mathbf{x})$

- Target variable  $y \in \{C_1, C_2, C_3, \dots, C_K\}$  in classification is a **descriptive variable**;
- To treat **classification as a regression problem** we should transform the target  $y$  into numerical values;
- The **choice** of numerical class representation is quite arbitrary;
- Careful numerical class representation is a **critical step**.

### Binary Classification

Let us represent the classes  $C_0$  and  $C_1$  with numerical values 0 and 1, i.e., if  $y \in C_0$  then  $y = 0$ , and if  $y \in C_1$  then  $y = 1$

Since we have assigned numeric values to classes, binary classification can be considered to be a regression problem. The optimal predictor for regression is

$$f^*(\mathbf{x}) = E[y | \mathbf{x}] = 0 \cdot P(y \in C_0 | \mathbf{x}) + 1 \cdot P(y \in C_1 | \mathbf{x}) = P(y \in C_1 | \mathbf{x})$$

Therefore, the optimal predictor outputs the posterior probability of class  $C_1$ . By applying the Bayes classification rule we can obtain the Bayes classifier!!

#### **Important Conclusion:**

With appropriate class representation, the optimal classification is equivalent to optimal regression.

### Multi-class Classification ( $K$ classes)

Could the previous result be generalized to multi-class classification?

**Example.**

3 classes: “white”, “black”, and “blue”.

Let us examine the representation:  $y = -1$  (class is “white”),  $y = 0$  (class is “black”), and  $y = 1$  (class is “blue”)

Discussion: The representation is inappropriate since it enforces order; implies that “white” and “blue” are further away than let’s say “black” and “blue.” What if it is evident that an example could be either “white” or “blue,” but definitely not “black”? The proposed representation would probably lead to a completely misleading answer “black”!!

### **Solution:**

Decompose the multi-class problem to  $K$  binary classification problems:

**Problem  $i$**  ( $1 \leq i \leq K$ ):

if  $y \in C_i$  then  $y = 0$

if  $y \notin C_i$  then  $y = 1$

The regression function  $E[y | \mathbf{x}]$  on **Problem  $i$**  will equal the posterior  $P(y \in C_i | \mathbf{x})$ . By repeating for all  $K$  classes, all posteriors will be available, which is sufficient to construct the Bayes classifier!!

## **Approaches to Minimizing MSE from a Finite Dataset**

**Goal:** Given a dataset  $D$  find a mapping  $f$  that minimizes MSE.

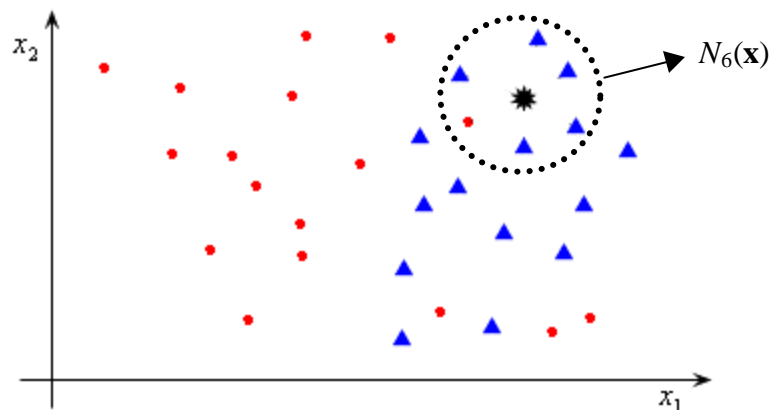
Two extreme approaches:

1. Nearest neighbor algorithm (non-parametric approach)
2. Linear regression (parametric approach)

Parametric approach assumes a functional form: e.g., the output is a linear function of the inputs.

Non-parametric approach assumes that data points close in the attribute space are similar to each other. It does not assume any functional form.

### **Nearest Neighbor Algorithms**



k-nearest neighbor (k-NN):  $f(\mathbf{x}) = \frac{1}{k} \sum_{\mathbf{x}_i \in N_k(\mathbf{x})} y_i$ , where  $N_k(\mathbf{x})$  are the  $k$  nearest neighbors of  $\mathbf{x}$

**Regression:** The prediction is the **average**  $y$  among the  $k$  nearest neighbors  
**Classification:** The prediction is the **majority class** among the  $k$  nearest neighbors

k-NN is also known as a “lazy learning algorithm” because it does not perform any calculations prior to seeing a data point; it has to analyze the whole dataset for the nearest neighbors every time a new data point appears.

**Note:** Parametric learning algorithms learn a function from the dataset; they are much faster in giving predictions but need to spend some time beforehand.

Theorem: If  $N \rightarrow \infty, k \rightarrow \infty, \frac{k}{N} \rightarrow 0$  (large number of neighbors in a very tight neighborhood) k-NN is an optimal predictor.

Practically, if we have 2 dimensions (2 attributes), k-NN is a good try; if we have more attributes, we may run out of data points (in practice, data size is always limited).

**Example:** Generate an M-dimensional dataset such that all attributes  $X_i, i = 1, 2, \dots, M$ , are uniformly distributed in interval  $[0,1]$ . What is the length  $r$  of the size of a hypercube that contains 10% of all data points?

M	$r$
1	0.1
2	0.31
5	0.63
10	0.80
100	0.98

In high dimensions, all neighboring points are far away, and could not be used to accurately estimate values with k-NN!!

## Linear Regression

Assumes a **functional form**

$$f(\mathbf{x}, ?) = \theta_0 \cdot x_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \dots + \theta_M \cdot x_M \quad (\text{Eq.2})$$

where  $\mathbf{x} = (x_0, x_1, \dots, x_M)$  are the attributes and  $\mathbf{q} = (\theta_0, \theta_1, \dots, \theta_M)$  are the function parameters.

**Example:**

$f(\mathbf{x}, ?) = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2^2 + \theta_3 \cdot x_1^4$ , where  $\mathbf{x} = (x_1, x_2)$  are the attributes and  $\mathbf{q} = (\theta_0, \theta_1, \theta_2, \theta_3)$  are the function parameters. Note that function  $f(\mathbf{x}, \mathbf{q})$  from the example is linear in the parameters. We can easily transform it into a function from from (Eq.2) by introducing new attributes  $x_0' = 1, x_1' = x_1$  and  $x_2' = x_2^2$ , and  $x_3' = x_1^4$ .

Linear regression is suitable for problems where functional form  $f(\mathbf{x}, \mathbf{q})$  is known with sufficient certainty.

**Learning goal:** Find  $\mathbf{q}$  that minimizes MSE

MSE is a function of parameters  $\mathbf{q}$ , so the problem of minimizing MSE can be solved by standard methods of the unconstrained optimization.

**Illustration of the linear regression:**

