**CSE 475: Statistical Methods in AI** **Monsoon 2019**

SMAI-M-2019 2: Mathematical Foundations of ML - II

*Lecturer: C. V. Jawahar* *Date: 1 Aug 2019*

## 2.8 Problem Space -II

In the last lecture, we introduced the problem of separating an email as spam or non-spam. We also discussed a number of similar problems (like separating apples and oranges or a patient has certain disease or not) can all be abstracted into a classification problem. In general, our input is a vector in $d$ dimension – $\mathbf{x}$. And our objective is to predict the $y$, an integer (classification) or a real number/vector (regression). Our objective is to learn a parameterized function $f(\mathbf{w}, \mathbf{x})$ that can predict $y$. A simple example is $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$.

How do we learn this function? That is the main story.

We are given many labeled examples $\mathcal{D} = \{(\mathbf{x_i}, \mathbf{y_i})\}$ where $i = 1, \ldots, N$ to learn the function (i.e., design the spam filter).

- **Training:** Training is the process of finding the function, (primarily finding the parameters $\mathbf{w}$), given sufficiently large training examples. This stage is mostly compute intensive. Outcome of the training is often the parameters/weights $\mathbf{w}$.
- **Testing:** Testing is the process of evaluating the function on a new sample (that is often unseen at the training time, like a new email that we receive) and predicting $y$. This is computationally light.
- **Performance Evaluation:** Given that we have now a spam filter, we would like to see how good is it objectively. A simple scheme could be how many emails are correctly classified. Percentage of the correctly classifier emails (a number in the range of 0-100) is a simple naive performance measure.

Q: Can you tell me why the percentage accuracy is a bad choice for a spam filter? What do we care in a spam filter?

There are a number of performance metrics useful for us. Here are some of them (i) Percentage Accuracy (ii) Hit Ratio (iii) Miss Ratio (iv) False Positive Rates (v) Precision (vi) Recall (vii) Average Precision (AP) (viii) AUC (Area under the curve) etc.

Q: Read and understand about these metrics on your own.

How do we learn the function in practice? What we have is only $N$ labeled examples. This is the popular strategy in our labs/experiments. (In practice, there is a real world deployment and seeing many many more real unlabeled emails.). The standard protocol followed is like this. We split the samples into two sets "Train" and "Test". Or we split $\mathcal{D}$ into two mutually exclusive subsets $\mathcal{D}_1$ and $\mathcal{D}_2$.

In some cases, a well defined partitions exists, in some other cases, a random partition is enforced. (If no detail is available let us split as 80:20 for Train:Test).

To summarize, this is what we will do in the exercises:

- **Input:** We are given $N$ labeled examples. we artificially split the data into two mutually exclusive sets train and test.
- **Training:** We make an assumption of the form of the function (eg. $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$) and we choose our favorite algorithm to learn the parameters of the function $\mathbf{w}$.
- **Testing:** We evaluate the solution on the test-set (say $\mathcal{D}_2$ and predict the output for each one independently.
- **Performance Evaluation:** We evaluate the performance of our solution by comparing the predicted output with actual (ground truth or what is provided to us) output.
  - Percentage accuracy is a simple popular measure for classification.
  - Mean square error (MSE) is a popular error measure for regression like problem.

**Point of Concern** The seperation of Train and Test set is very imnportant. It is quite likely that we may use test set for training, inadvertently. This has to be strictly avoided for practicing ML.

- For example, your code will not use the data but you (as a person) will look at the test data many time, you are indirectly using the test data for training. This happens in practice. This should be avoided.

– Sometime, we do not see the test-data as individuals, but as a community, we see it many manmy times (like popular data sets) leading/encouraging development of algorithms that use this information since it is rewarding to do well on such a specific public benchmark. This also leads to algorithms that do well on the train test but not well on *real novel situations*.

**Problem:** In the previous protocol, we assumed that each sample is tested independently. What about we test all the $N_2$ samples together and output a ranked list of samples based on the confidence. This is a popular requirement in many ranking style of problems. (eg. information retrieval). A popular measure in such cases, is "Average Precision". With one or two numerical examples, explain average precision.

## 2.9 Matrices and Vectors - II

### 2.9.1 Augmented Vectors and Feature Maps

We started with the problem of $\mathbf{w}^T\mathbf{x} >$ or $< 0$. Note that this assumes that our line/plane/hyperplane pass through origin and it does not give us a complete family of solutions. We should also add an additional $w_0$ to address this. Note that this is same as $\mathbf{w}^T\mathbf{x} >$ or $< w_0$

The notation could be simpler if we do not have $w_0$. We do that by augmenting the vector $\mathbf{x}$ with an additional quantities. i.e., the new $\mathbf{x}$ is the old $\mathbf{x}$ with an additional 1 concatenated at the end. Similarly the $\mathbf{w}$ is augmented with $w_0$ and the decision function gets simplified as

$$y = f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} \tag{2.1}$$

Note that we did not introduce a different notation with and without augmentation. This is to make the notations simpler. (Text books (such as Duda and Hart) may be using different notations for these). Hope you appreciate the convenience of augmenting with 1.

What more we can do with augmentation (or explicit feature maps)? We can infact create a new vector from the old ones. This also generalizes the trick of augmentation.

Consider that our original vector is $[x_1, x_2]^T$. We now know how to create a new vector $[x_1, x_2, 1]$. Why not $[x_1^2, x_2^2, x_1x_2, x_1, x_2, 1]$ ? Such a modification will allow us to learn a new model $\mathbf{w}^T\mathbf{x}$ as

$$w_5x_1^2 + w_4x_2^2 + w_3x_1x_2 + w_2x_1 + w_1x_2 + w_0 \tag{2.2}$$

which is really a quadratic function. Our linear algorithm (that we see soon ) is able to learn nonlinear models too. The objective of introducing this at this stage is to convince that the algorithms that we will discuss are very powerful and useful. Linearity does not constrain us too much.

#### 2.9.1.1 Properties/Terms

1. **Norm** A norm of a vector is informally the length of the vector, represented as $||\mathbf{x}||$. A popular way to visualize is as the Euclidean L2 norm as

$$\sqrt{\sum_{i=1}^{n} x_i^2}$$

2. $L^p$ **Norms** A popular class of norms of our interest is Lp norm defined as:

$$||\mathbf{x}||_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$$

When we $p = 2$, it is L2 norm, which we already saw. Another popular norm is $L1$ norm. $||\mathbf{x}||_1 = \sum_{i=1}^{n} |x_i|$

There are two special (pseudo) norms of interest

$$L_\infty = \max_i |x_i| \text{ and } L_0 = \text{ number of no-zero } x_i$$

There are also matrix norms. The most popular one is Frobenius norm defines as

$$||\mathbf{A}||_F = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N} A_{ij}^2} = \sqrt{Tr(A^TA)}$$

3. **Distance Or Similarity functions** that can compare two vectors.
   (a) Euclidean Distance
   (b) Weighted Euclidean (L2) distance
   (c) Cosine Similarity
   (d) ?? (your own cricket distance. But is it metric?)

### 2.9.2 Matrices

It is assumed that you are familiar with the notion of matrices. Here are some terms that you should revise, practice on how to numerically compute, if appropriate.

1. **Symmetric, Identity, Triangular** matrices
2. **Rows, Columns, Row-space, column-space**

3. **Determinant**
4. **Rank**
5. **Trace**
6. **Notion of Linear Independence**
7. **PD, PSD**
8. **Graph-Schmidt Orthogonalization**

### Many known results

- What is $(ABC)^T$?
- What is $(ABC)^{-1}$
- Is $AB$ same as $BA$?

## 2.10 Interpretations

### 2.10.1 Vectors

- Vectors and Directions
- Geometric Interpretation of $\mathbf{W}$ in $\mathbf{w}^T\mathbf{x}$
- Basic Vectors, Span

### 2.10.2 Matrices

- Matrix Multiplication as Linear Transformation
- Solving $\mathbf{Ax} = \mathbf{b}$
- Solving $\mathbf{Ax} = \mathbf{0}$
- Disadvantages of closed form solution for solving equations
- Solving $\mathbf{Ax} = \mathbf{b}$ when $\mathbf{A}$ is non square or rank deficient. Some special cases.
- Translation, Rotation and Scaling of Data
- Normalization of data

## 2.11 Home Works (Submit by Aug 9 5pm

1. We know that the cosine similarity is a popular way to compare two vectors. Write the expression. This is in the range of [-1,+1]. When can it be -1, 0 and +1? Demonstrate with an example in 4 dimensional space.

2. Consider a set of samples $\mathcal{D} = \{\mathbf{x_i}, i \in 0, 1, 2, ..., n\}$. We also have a linear classifier defined by $\mathbf{w}$ that classifies samples based on whether $\mathbf{w}^T\mathbf{x_i}$ is greater than zero or less than / equals to zero into class A or B respectively.

   - If all samples are multiplied by a scalar $\alpha \in R^+$ as $\alpha\mathbf{x}_i$, will the accuracy of the classifier change on this set? Why?
   - If all samples are multiplied by independent random scalars $\alpha_i$ as $\alpha_i\mathbf{x}_i$, will the accuracy of the classifier remain unchanged? Why?
   - If the samples were linearly transformed by a matrix $\mathbf{A}$ ($\mathbf{Ax}$ that is orthonormal (i.e., $\mathbf{A^TA} = \mathbf{AA^T} = \mathbf{I}$), will the accuracy change?
   - If the samples were linearly transformed by a matrix $\mathbf{A}$ that is rank deficient (with determinant zero), will the accuracy remain unchanged?
   - From the above specific examples, discuss when and how the accuracy will remain unchanged, in general. Make your argument scientific and analytical as much as possible.

3. Consider a problem of classifying (testing) samples in 2D (i.e., $[x_1, x_2]^T$). Each sample is augmented with 1 to get $\mathbf{x} = [x_1, x_2, 1]^T$. Classification is done as "decide as class A if $\mathbf{w}^T\mathbf{x} > 0$ else decide as class B". Where $\mathbf{w} = [w_1, w_2, w_3]^T$.

   Write a program that generates 50 random samples from class A and another 50 random samples from class B leading to a total of 100 samples. For all the samples $x_1$ and $x_2$ are in the range [-1,+1]. (Indeed there is no structure to these classes, when we generate samples randomly. It is OK for today).

   (i) For the following $\mathbf{w}$ vectors calculate the accuracy (as percentage). (a) $[1, 1, 0]^T$ (b) $[-1, -1, 0]^T$ (c) $[0, 0.5, 0]^T$ (d) $[1, -1, 5]^T$ (e) $[1.0.1.0, 0.3]^T$. Report the accuracies. (ii) Provide the plots (data + decision boundary) for (a), (d) and (e). (iii) Explain why do you get these accuracies given that data is random with equal probabilities. Where you able to guess before the experiment/code?