

CS 231A Computer Vision (Autumn 2012)

Problem Set 0

Due: Oct. 2, 2012 (2:15 pm)

1 Monty Hall (20 points)

In Computer Vision, probability is a widely used mathematical tool to solve problems ranging from low-level to high-level vision. This exercise will serve as a reminder of some basic probability concepts you find useful later in the course.

Consider the following problem: Gold is placed behind one of three curtains. A contestant chooses one of the curtains. Monty Hall, the game host, opens an unselected empty curtain. The contestant can choose either to switch his selection to the third curtain or not. In this problem you will use basic probability to maximize the contestant's probability of winning the gold.

- (a) List the set of all possible outcomes i.e. the sample space for the Monty Hall problem. Let the three curtains be labeled A, B, C . An outcome consists of the curtain with gold, x , the curtain chosen by the contestant, y , and the curtain chosen by Monty, z , denoted as (x, y, z) . For example, the outcome (A, B, C) means that the curtain with the gold is A , the curtain chosen by the contestant is B , and the curtain chosen by Monty is C .
- (b) Assume that placement of the gold behind the three curtains is random, that the contestant's choice of curtains is random and independent of the gold placement, and that Monty Hall's choice of an empty curtain is equally likely between the two alternatives (if the two unselected curtains are empty). Specify the probabilities of each outcome for this random experiment and use it to compute the probability of winning the gold if the contestant decides to switch. For instance, consider computing $P(A, B, C)$. This can be rewritten as $P(A, B, C) = P(C \mid A, B)P(A, B) = P(C \mid A, B)P(A)P(B)$. The first equality holds from the chain rule of probability, and the second equality from the fact that the choice of position of the gold is independent of the choice of the contestant.
- (c) Given your answer to part b, should the contestant switch their choice? That is, compute $P(W \mid S = 1)$ (the probability of winning if the contestant switches) and $P(W \mid S = 0)$ (the probability of winning if the contestant doesn't switch), which tell you whether the contestant should switch curtains or not.

2 Support Vector Machines (20 points)

Machine learning is an important tool to accomplish many tasks in Computer Vision, especially classification problems. One staple algorithm in binary classification is the Support Vector Machine (SVM). In this problem you will implement a SVM package and use it to separate a sample dataset.

First, let's get some insight into why we might want to use a SVM in a binary classification problem. SVMs are an example of supervised learning; that is, after training a model with known input/output pairs, the model can classify new datapoints. In the binary classification problem you have a set of n data points, $\{x^{(1)}, \dots, x^{(n)}\}$, with labels $\{y^{(1)}, \dots, y^{(n)}\}$ where $y^{(i)} \in \{-1, 1\}$. We are looking for a function $h(x)$ to separate the data points with different labels as shown in Fig 1. The data points which lie along the dotted line in Fig 1 are called the support vectors. Specifically the SVM is the solution to the optimization problem

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \\ \text{subject to} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, \dots, n \\ & \xi_i \geq 0, \quad i = 1, \dots, n \end{aligned}$$

where w and b are parameters that define our classifier $h(x) = w^T x + b$. The parameter C controls the relative weighting between the twin goals of making the $\|w\|^2$ small and of ensuring that most examples are correctly classified. If you want more specifics on SVMs, please read Prof. Andrew Ng's Notes at the CS 229 course webpage <http://www.stanford.edu/class/cs229/materials.html>.

In this problem you will install the SVM package `liblinear`. You will use this package through its Matlab interface. The `liblinear` package is written in C, so you will need to have a compiler linked to your Matlab install.

- (a) Use the `liblinear` package to test and train a SVM on an idealized dataset (similar to that shown in Fig 1). This will also act as a tutorial through installation and use of `liblinear`.

Download the current version of the `liblinear` package from <http://www.csie.ntu.edu.tw/~cjlin/liblinear/#download>. First read the README, then if you are using windows set your current Matlab directory to `.\liblinear-version\matlab\` and run the `make.m` script. If you are using Mac OS X or a Linux distribution you should edit Makefile such that the variable `MATLABDIR` is the installation directory of Matlab. To find the Matlab installation directory, open Matlab and use the `matlabroot` command to return the installation path. Then set `MATLABDIR` to this directory in your Makefile. Then type `make` in the terminal to build. In case you have issues making `liblinear`, we have included a compiled version of its `train` function in `PS0_data.zip` which has been tested to work on both `corn` and `myth`.

Download `PS-0_data.zip` from the course webpage and run the `SVM_data_parta` script which will load the data for this problem into your Matlab workspace. This data will act as a test for your installation of `liblinear`.

Once you have installed `liblinear`, write a Matlab script that trains a classifier using `data` and plots your separating $h(x)$ and the given data. The script `plotSVM` is written for you that plots the separating $h(x)$ and the given data. You will know if your answer is

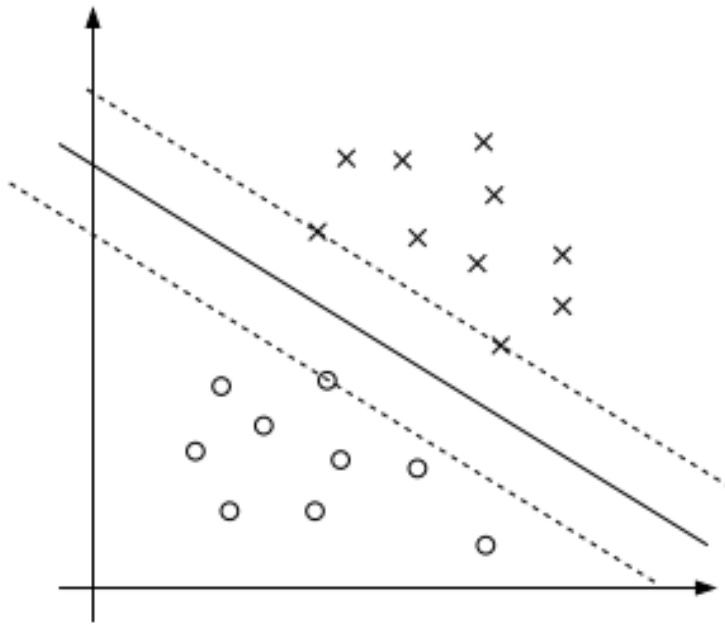


Figure 1: An example of the output of a separating $h(x)$ for a linearly separable data set.

correct because it should be very similar to the data above. There should be no incorrect classifications because the data for this problem is linearly separable. *Hint: the input matrices need to be sparse, so Matlab's `sparse` command will be useful*

Turn in a plot of the classifier and the data, as well as your code.

- (b) In practice, data is rarely linearly separable and so regularization is needed to make the model more robust to outliers. In this part you will experiment with the regularization parameter C to find an optimal fit. Run the `SVM_data_partb` script to get the new data set for this problem. Using `liblinear` and varying the parameter C by using `C = logspace(-4,4,5)`, find a classifier that is visually similar to the classifier you found in part a.

Turn in a plot of the classifier and the data, your value of C , as well as your code.

3 Lagrangian derivation of least norm (20 points)

Many times in computer vision it is necessary to formulate an optimization problem to solve the vision task at hand. In this problem you will use the Lagrangian to solve an optimization problem. Consider finding an $x \in \mathbb{R}^n$ such that $Ax = b$, where $A \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$. In this problem we will assume that a solution x exists. There is the possibility that there is more than one solution. We explore how to pick a single solution when many exist. There is also the possibility that no solution exists for $Ax = b$, and we wish to find an x which nearly satisfies $Ax = b$. We will not address this situation in this problem.

In the scenario where there exists multiple solutions to $Ax = b$ we can formulate an optimization problem to choose one of these solutions. Consider the following optimization problem

$$\begin{array}{ll} \text{minimize} & \|x\|_2^2 \\ \text{subject to} & Ax = b \end{array}$$

The solution to this optimization problem is the least norm solution to $Ax = b$ when there are multiple values of x . This may occur when e.g. $n \geq m$, which we shall assume is true for the remainder of the problem.

- (a) Formulate the Lagrangian $\mathcal{L}(x, \lambda)$ of the optimization problem.
- (b) Find the partial derivatives of $\mathcal{L}(x, \lambda)$ with respect to x and λ , then set the derivatives equal to zero. The x that satisfies these conditions is the optimal solution x_{opt} . Give the closed form solution for x_{opt} in terms of A and b . Assume that A is full rank (has rank m).

4 Interacting with the SVD (20 points)

Linear algebra is essential in Computer Vision, especially in camera models and epipolar/multiview-geometry. Eigenvectors, eigenvalues and the fundamental spaces (nullspace, row space, etc.) are important concepts with which one has to be comfortable. One popular tool is the Singular Value Decomposition (similar to Principle Component Analysis), which is a powerful technique to understand how a matrix transforms a space. From this information, many techniques (such as compression and sensitivity analysis) can be reasonably implemented in very large dimensions. An excellent resource for this topic can be found in Prof. Andrew Ng's lecture notes at www.stanford.edu/class/cs229/notes/cs229-notes10.pdf.

The purpose of this problem is to familiarize yourself with properties of the Singular Value Decomposition through experiments in a low-dimensional setting. We will use the following matrix throughout this problem:

$$A = \begin{bmatrix} -2 & 11 \\ -10 & 5 \end{bmatrix}$$

- (a) Find all solutions to the equation $Ax = b$ where $b = [1 \ 1]^T$
- (b) Calculate the U , Σ , V matrices for the SVD of A . Is this unique? If not, give an alternative solution. *hint: you may find the Matlab command `svd` useful*
- (c) Sketch the ellipsoid corresponding to the SVD of A . The ellipsoid will have semi-axis directions of the left singular vectors u_i , and with corresponding semi-axis half lengths of σ_i .
- (d) Find the rank-1 approximation of A , denoted \tilde{A} . That is, set the smallest magnitude singular value to zero and use this new SVD to calculate \tilde{A} . Sketch the ellipsoid for the SVD of this matrix, \tilde{A} .
- (e) Repeat part a but replace A with its rank-1 approximation, \tilde{A} . *hint: make sure to include the solution for $\tilde{A}x = 0$*

5 Maximum Likelihood Estimation (20 points)

There are many problems in Computer Vision that are most suitable for solving through probabilistic techniques. In fact, the foundation for many state-of-the-art high level vision algorithms (scene understanding, human pose estimation, etc.) is in probabilistic approaches. Maximum Likelihood Estimation, a popular yet straightforward method, is the process of maximizing the probability that a parameterized model would behave in a specified manner. Prof. Andrew Ng gives a good overview of MLE in the context of its similarity to Linear Regression in pages 11-13 of his notes at www.stanford.edu/class/cs229/notes/cs229-notes1.pdf.

In this problem, Consider that N independent samples x_i , $1 \leq i \leq N$ are drawn from a Gaussian distribution with unknown mean and unknown variance σ^2 . Recall that the formula for a Gaussian density is

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

Your task is to estimate these unknown parameters given the N samples by maximum likelihood.

- (a) Write an expression for the probability of the sample x_k given the parameters μ and σ .
- (b) Now write an expression for the likelihood (the joint probability), L , of all N samples given that all samples are identically and independently distributed.
- (c) Find the values of μ and σ that will maximize the log-likelihood, $l = \ln L$. This is equivalent to maximizing the likelihood, but applying the logarithm to part (b) turns the product into a sum, which is much easier to manipulate.