ICS 635 HW1

January 2019

Probability

Problem 1: (Linearity of Expectation)

The expected value of a continuous random variable X, taking values x, is defined as $\mu_x = E[X] = \int p(x)\ x\ dx$ where $p(x)$ is the probability density function for X. The variance is defined as $var(X) = E[(X - \mu_x)^2] = \int p(x)(x - \mu_x)^2 dx$ (often also denoted as σ_x^2).

1. Prove that expectation is linear, i.e., that $E[aX + b] = aE[X] + b$ where a and b are constants.
2. Prove that $var(cX) = c^2 var(X)$ where c is a constant.
3. Prove that $var(X) = E[X^2] - (E[X])^2$.

Problem 2: (Uniform Density)

Let X be a continuous random variable with uniform density $U(a, b)$, with $a < b$, i.e.,

$$p(x)\ =\ p(X = x)\ = \frac{1}{b - a}$$

if $a \leq x \leq b$ and $P(x) = 0$ otherwise.

1. Derive an expression for $E[X]$.
2. Derive an expression for $var(X)$.

Problem 3: (Non-Uniform Density)

Let X_1 and X_2 be independent, continuous random variables uniformly distributed on $[0, 1]$. Let $X = \min(X_1, X_2)$. Compute

1. $E[X]$.
2. $var(X)$.
3. $cov(X, X_1)$.

Problem 4: (Bayes Rule)

1. The probability that a random individual has disease D is $p(D) = .001$. A test for the disease has sensitivity S (i.e, $p(T^+|D) = S$ where T^+ is the event that test is positive) and specificity Q (i.e., the probability of testing negative given that one doesn't have the disease $p(T^-|\text{not } D) = Q$). If a patient tests positive, what is the Bayesian posterior for having the disease, $P(D|T^+)$?

2. Suppose $S = Q = .99$. What is the posterior $p(D|T^+)$?

3. Suppose $S = .99, Q = .90$. What is the posterior $p(D|T^+)$?

Linear Algebra

Problem 5: (The Gradient and the Hessian)

A matrix $A \in \mathbb{R}^{n \times n}$ is *symmetric* if $A^T = A$, that is, $A_{ij} = A_{ji}$ for all i, j. The gradient $\nabla f(\mathbf{x})$ of a function $f : \mathbb{R}^n \to \mathbb{R}$ is the n-vector of partial derivatives. The hessian $\nabla^2 f(\mathbf{x})$ of a function $f : \mathbb{R}^n \to \mathbb{R}$ is the $n \times n$ symmetric matrix of twice partial derivatives,

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix} \quad \text{where} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f(x)}{\partial x_n \partial x_n} \end{bmatrix}$$

1. Let $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} + \mathbf{b}^T\mathbf{x}$ where A is a symmetric matrix and $\mathbf{b}$ is a vector. What is $\nabla f(\mathbf{x})$? (Hint: This can be written as a simple matrix formula.)

2. Let $f(\mathbf{x}) = g(h(\mathbf{x}))$, where $g : \mathbb{R} \to \mathbb{R}$ and $h : \mathbb{R}^n \to \mathbb{R}$ are both differentiable. What is $\nabla f(\mathbf{x})$?

3. Let $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} + \mathbf{b}^T\mathbf{x}$ where A is a symmetric matrix and $\mathbf{b}$ is a vector. What is $\nabla^2 f(\mathbf{x})$?

4. Let $f(\mathbf{x}) = g(\mathbf{a}^T\mathbf{x})$, where $g : \mathbb{R} \to \mathbb{R}$ is continuously differentiable and $\mathbf{a} \in \mathbb{R}^n$ is a vector. What are $\nabla f(\mathbf{x})$ and $\nabla^2 f(\mathbf{x})$? (Hint: $\nabla^2 f(\mathbf{x})$ can be written with ~ 10 symbols.)

Supervised Learning

Problem 6: (Linear Regression)

In this problem we will consider linear regression with L_1 regularization (also known as the LASSO). This model introduces a hyper-parameter $\lambda \in \mathbb{R}_{\geq 0}$ that controls the complexity of the model: $\lambda = 0$ results in no regularization, while large λ results in heavy regularization. Typically, we will need to choose λ based on the validation performance of our model. In this problem, we look at what happens when we choose λ poorly. Recall that the cost function $\mathcal{L}(\mathcal{D}; \boldsymbol{\theta}, \lambda)$ of the L_1-regularized regression model parameterized by $\boldsymbol{\theta} = \mathbf{w} = \langle w_1, w_2, \ldots, w_D \rangle$ on training data set $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_N$ is:

$$\mathcal{L}(\mathcal{D}; \boldsymbol{\theta}, \lambda) = \sum_{n=1}^{N} (y_n - (\mathbf{w}^T\mathbf{x}_n))^2 + \lambda \|\mathbf{w}\|_1$$

where

$$\lambda \|\mathbf{w}\|_1 = \lambda \sum_{d=1}^{D} |w_d| \tag{1}$$

and λ is our regularization constant. (Note: We have "hidden" the bias term using the trick described in class.)

1. Suppose our λ is much too small; that is,

$$\sum_{n=1}^{N} (y_n - (\mathbf{w}^T\mathbf{x}_n))^2 + \lambda \|\mathbf{w}\|_1 \approx \sum_{n=1}^{N} (y_n - (\mathbf{w}^T\mathbf{x}_n))^2$$

   How will this affect the magnitude of:

   (a) The error on the training set?

   (b) The error on the validation set?

   (c) $\mathbf{w}$?

   (d) The number of nonzero elements of $\mathbf{w}$?

2. Suppose instead that we overestimated on our selection of λ. What do we expect to be the magnitude of:

   (a) The error on the training set?

   (b) The error on the validation set?

   (c) $\mathbf{w}$?

   (d) The number of nonzero elements of $\mathbf{w}$?

3. Suppose that instead of L_1 regularization, we use L_2 regularization, such that

$$\mathcal{L}(\mathcal{D}; \boldsymbol{\theta}, \lambda) = \sum_{n=1}^{N} (y_n - (\mathbf{w}^T \mathbf{x}_n))^2 + \lambda \|\mathbf{w}\|_2^2$$

   (a) How would this effect the number of nonzero elements of $\mathbf{w}$?

   (b) Compute the gradient with respect to $\mathbf{w}$ of the single-example loss $L(y_n, \hat{y}_n) = (y_n - (\mathbf{w}^T \mathbf{x}_n))^2$, where $\hat{y}_n = \mathbf{w}^T \mathbf{x}_n$ is the prediction on the n-th example.

   (c) Compute the gradient with respect to $\mathbf{w}$ of the *total* cost, $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}; \boldsymbol{\theta}, \lambda)$.

Problem 7: Artificial Neural Network

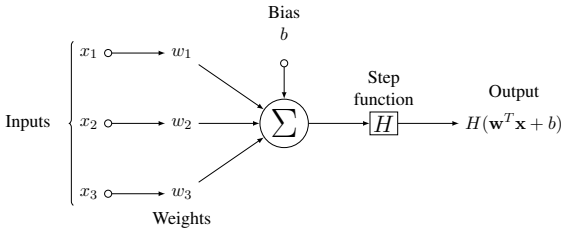
Consider the following Boolean function known as the *exclusive OR* or *XOR* gate.

$\mathbf{x}$		y
0	0	0
0	1	1
1	0	1
1	1	0

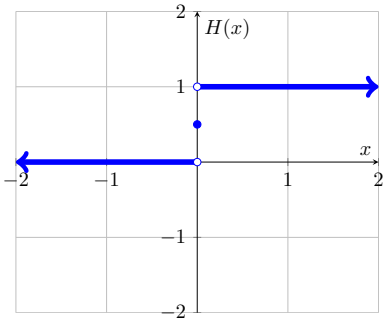
In this problem, you will attempt to *construct* a feedforward artificial neural network that computes this function (with zero loss). You will do this using composable simple functions analagous to neurons in biology, or gates in a circuit, which compute simple non-linear functions of their inputs.

1. The linear network architecture in Figure 2 (Left) has three parameters. Are there values of the parameters such that the loss is zero? If so provide an example.

2. The layered architecture in Figure 2 (Right) has nine parameters. Are there values of these parameters such that the cost is zero? If so provide an example.



(a) Artificial neuron with three inputs, a bias term, and a non-linear activation function.



(b) The Heaviside step function.

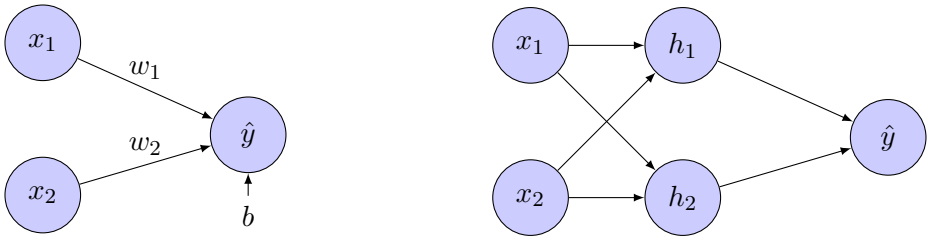


Figure 2: Two neural network architectures. Left: a network with two inputs and one output, where the output computes $\hat{y} = H(w_1x_1+w_2x_2+b)$. Right: a network with two hidden units (h_1, h_2, where $h_i = H(w_1^{(i)}x_1+w_2^{(i)}x_2+b^{(i)})$) and an output unit $\hat{y} = H(w_1^{(3)}h_1+w_2^{(3)}h_2+b^{(3)})$. The superscripts indicate that $w_1^{(1)}$ is different parameter from $w_1^{(2)}$. The weights and bias terms are not labeled in (b), but they are implied.

Programming

Problem 8: (Scikit-Learn)

In this problem you will experiment with a machine learning classifier that has already been implemented for you. The purpose of this exercise is to familiarize you with a basic software stack for doing ML in python with Scikit-Learn (sklearn), numpy, and matplotlib in the Jupyter environment. The task itself should not take long.

A Jupyter notebook file containing code is available at the following link. You can download and run the notebook on your machine, or you can click on the link to run it in Google Colab, a free service that allows users to run Jupyter notebooks on Google servers. If you install python on your machine, I recommend installing

Python 3 (not 2) using the Anaconda package manager.
→github.com/peterjsadowski/sklearn_examples/blob/master/sklearn_classifier.ipynb

1. Preprocessing the features (e.g. feature *normalization*) can affect the performance of ML classifiers. Consider the simple act of scaling one of the features. What effects could this have on a linear classifier? What effects could this have on a K-nearest neighbors classifier?
2. Try to maximize the generalization accuracy of the classifier. You may use whatever models or techniques you wish. What is the (estimated) generalization accuracy for your best model? Briefly describe your strategy and types of things you tried.
3. Demonstrate that there is an 'optimal' setting for one of the hyperparameters. Using the ML classifier model of your choice, vary this hyperparameter and plot the validation performance (or cross-validation performance) for each value. (It is sufficient to describe what you did and/or draw a plot as a result; it is not necessary to print code or figures in this homework.)

Solutions

Problem 1

Trivial by definitions.

Problem 2

Computing some integrals.

Problem 3

The hard part in 1-3 is computing an expectation over a function of $f(X)$, which can be accomplished using the following decomposition:

$$E[f(X)] = \int_0^1 dX_2 \left(\int_0^{X_2} dX_1 f(X_1) + \int_{X_2}^1 dX_1 f(X_2) \right)$$

Problem 4

1. $$P(D^+|T^+) = \frac{P(T^+|D^+)P(D^+)}{P(T^+)} = \frac{P(T^+|D^+)P(D^+)}{P(T^+|D^+)P(D^+) + P(T^+|D^-)P(D^-)} = \frac{.001S}{.001S + (1-Q)(1-.001)}$$

2. $$P(D^+|T^+) = \frac{S0.001}{S0.001 + (1-Q)(0.999)} = \frac{0.99 \times 0.001}{0.99 \times 0.001 + 0.01 \times 0.999} \approx \frac{1}{10}$$

3.

$$P(D^+|T^+) = \frac{S0.001}{S0.001 + (1-Q)(0.999)} = \frac{0.99 \times 0.001}{0.99 \times 0.001 + 0.1 \times 0.999} \approx \frac{1}{100}$$

Problem 5

1.

$$\begin{aligned}
\mathbf{x}^T A\mathbf{x} &= \mathbf{x}^T (A\mathbf{x}) \\
&= x^T \begin{bmatrix} \sum_{j=1}^n A_{1j}x_j \\ \sum_{j=1}^n A_{2j}x_j \\ \cdots \\ \sum_{j=1}^n A_{nj}x_j \end{bmatrix} \\
&= x_1 \sum_{j=1}^n A_{1j}x_j + x_2 \sum_{j=1}^n A_{2j}x_j + \cdots + x_n \sum_{j=1}^n A_{nj}x_j \\
&= \sum_{i=1}^n \sum_{j=1}^n x_i A_{ij} x_j \\
&= \sum_{i=1}^n \sum_{j=1}^n A_{ij} x_i x_j \\
\frac{\partial \sum_{i=1}^n \sum_{j=1}^n A_{ij}x_i x_j}{\partial x_k} &= \sum_{j \neq k}^n A_{kj}x_j + \sum_{i \neq k}^n A_{ik}x_i + 2A_{kk}x_k \\
&= 2A_{kk}x_k + 2\sum_{j \neq k}^n A_{kj}x_j \quad \text{(A is symmetric)} \\
&= 2\sum_{j=1}^n A_{kj}x_j \\
&= 2A_{k\cdot}\mathbf{x}
\end{aligned}$$

and $\nabla_x(b^T\mathbf{x}) = b$, so we have

$$\nabla_x(\frac{1}{2}\mathbf{x}^T A\mathbf{x} + b^T\mathbf{x}) = A\mathbf{x} + b$$

2.

$$\nabla_x f(x) = \nabla_x g(h(x)) = \begin{bmatrix} \frac{d}{dh}g(h)\frac{\partial}{\partial x_1}h(x) \\ \cdots \\ \frac{d}{dh}g(h)\frac{\partial}{\partial x_n}h(x) \end{bmatrix}$$

3. From part 1:

$$\frac{\partial \mathbf{x}^T A\mathbf{x}}{\partial x_i} = 2\sum_{j=1}^{n} A_{ij}x_j$$

So

$$\frac{\partial \mathbf{x}^T A\mathbf{x}}{\partial x_i x_j} = 2A_{ij}$$

and

$$\nabla^2(\frac{1}{2}\mathbf{x}^T A\mathbf{x} + b^T\mathbf{x}) = A$$

4.

$$\begin{aligned}\nabla f(\mathbf{x}) &= g^{'}(\mathbf{a}^T\mathbf{x}) \cdot \frac{\partial \mathbf{a}^T\mathbf{x}}{\partial \mathbf{x}} \\ &= g^{'}(\mathbf{a}^T\mathbf{x}) \cdot \mathbf{a}\end{aligned}$$

$$\begin{aligned}\nabla^2 f(\mathbf{x}) &= \begin{bmatrix} \frac{\partial g^{'}(\mathbf{a}^T\mathbf{x})a_1}{\partial x_1} & \cdots & \frac{\partial g^{'}(\mathbf{a}^T\mathbf{x})a_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g^{'}(\mathbf{a}^T\mathbf{x})a_n}{\partial x_1} & \cdots & \frac{\partial g^{'}(\mathbf{a}^T\mathbf{x})a_n}{\partial x_n} \end{bmatrix} \\ &= \begin{bmatrix} g^{''}(\mathbf{a}^T\mathbf{x})a_1a_1 & \cdots & g^{''}(\mathbf{a}^T\mathbf{x})a_1a_n \\ \vdots & \ddots & \vdots \\ g^{''}(\mathbf{a}^T\mathbf{x})a_na_1 & \cdots & g^{''}(\mathbf{a}^T\mathbf{x})a_na_n \end{bmatrix} \\ &= g^{''}(\mathbf{a}^T\mathbf{x}) \cdot (\mathbf{a}\mathbf{a}^T)\end{aligned}$$

Problem 6

1. (a) decrease, (b) increase, (c) increase, (d) increase.
2. (a) increase, (b) increase, (c) decrease, (d) decrease.
3. (a) increase
   (b) Note the sign of the gradient.

   $$\nabla_{\boldsymbol{\theta}} L = 2(\hat{y}_n - y_n)\mathbf{x_n}$$

   (c) Note that the constant terms (2 and 1/N) are not particularly important because we end up tuning the learning rate anyways.

   $$\nabla_{\boldsymbol{\theta}} \mathcal{L} = \frac{1}{N}\sum_{n}^{N} 2(\hat{y}_n - y_n)\mathbf{x_n} + \lambda\mathbf{w}$$

0.1 Problem 7

1. No.
2. Yes. A possible solution is to make one hidden unit compute the AND function, and the other compute the OR function, then the output unit should turn on only when $h_{OR} = 1$ and $h_{AND} = 0$.

0.2 Problem 8

1. In a linear classifier, scaling one of the features would change the optimal corresponding weight, but wouldn't change the space of possible decision boundaries. In a KNN classifier, scaling one of the features changes the similarity function used to compute the neighbors, and could have a big impact on the resulting decision boundary.
2. The optimal accuracy is 81%. This is the accuracy one would achieve with an arbitrary amount of training data. (The data was generated from a secret data distribution.) Many will succumb to the temptation to overestimate their generalization performance.
3. Plotting the validation loss for varying values of K in K-Nearest Neighbors gives a good example.