Homework 2

Question One

Least mean square algorithm

We want to choose θ so as to minimize J(θ). To do so, lets use a search algorithm that starts with some "initial guess" for θ, and that repeatedly changes θ to make J(θ) smaller, until hopefully we converge to a value of θ that minimizes J(θ). Specifically, lets consider the gradient descent algorithm, which starts with some initial θ, and repeatedly performs the update:

$\theta_j := \theta_j - \alpha\, \partial / \partial\theta_j .\, J(\theta)$. (This update is simultaneously performed for all values of j = 0, . . . , n.)

Here, α is called the learning rate. This is a very natural algorithm that repeatedly takes a step in the direction of steepest decrease of J. In order to implement this algorithm, we have to work out what is the partial derivative term on the right hand side. Lets first work it out for the case of if we have only one training example (x, y), so that we can neglect the sum in the definition of J.

We have:

$\partial / \partial\theta_j\, J(\theta) = \partial / \partial\theta_j\, 1/2\, (h_\theta(x) - y)^2$

$= 2 \cdot 1\,2\, (h_\theta(x) - y) \cdot \partial / \partial\theta_j (h_\theta(x) - y)$

$= (h_\theta(x) - y) \cdot \partial / \partial\theta_j \left(\sum^{n}_{i=0} \theta_i X_i - y\right)$

$= (h_\theta(x) - y)\, x_j$

For a single training example, this gives the update rule:

$\theta_j := \theta_j + \alpha(y(i) - h_\theta(x^{(i)}))\; x_j^{(i)}$

Update rules:

Consider the objective function $f:Rd \rightarrow Rf:Rd \rightarrow R$ that takes any multi-dimensional vector $x=[x1,x2,\ldots,xd]^\top x=[x1,x2,\ldots,xd]^\top$ as its input. The gradient of f(x) with respect to x is defined by the vector of partial derivatives:

$\nabla x f(x)=[\partial f(x)/\partial x1,\partial f(x)/\partial x2,\ldots,\partial f(x)/\partial xd]^\top . \nabla x f(x)=[\partial f(x)/\partial x1,\partial f(x)/\partial x2,\ldots,\partial f(x)/\partial xd]^\top .$

To keep our notation compact we may use the notation $\nabla f(x)$ and $\nabla xf(x)$ interchangeably when there is no ambiguity about which parameters we are optimizing over.Each element $\partial f(x)/\partial xi$ of the gradient indicates the rate of change for f at the point x with respect to the input xi only. To measure the rate of change of f in any direction that is represented by a unit vector u, in multivariate calculus, we define the directional derivative of f at x in the direction of u as:

$Duf(x)=limh→0f(x+hu)−f(x)h,Duf(x)=limh→0f(x+hu)−f(x)/h,$

which can be rewritten according to the chain rule as

$Duf(x)=\nabla f(x)\cdot u.Duf(x)=\nabla f(x)\cdot u.$

Since Duf(x) gives the rates of change of f at the point x in all possible directions, to minimize f, we are interested in finding the direction where f can be reduced fastest. Thus, we can minimize the directional derivative Duf(x) with respect to u. Since

$Duf(x)=\| \nabla f(x)\| \cdot \| u\| \cdot \cos(\theta)=\| \nabla f(x)\| \cdot \cos(\theta)Duf(x)=\|\nabla f(x)\|\cdot \|u\|\cdot \cos(\theta)=\|\nabla f(x)\|\cdot \cos(\theta)$, where:

θ is the angle between $\nabla f(x)\nabla f(x)$ and u, the minimum value of $\cos(\theta)\cos(\theta)$ is -1 when $\theta=\pi$. Therefore, Duf(x) is minimized when u is at the opposite direction of the gradient $\nabla f(x)$. Now we can iteratively reduce the value of f with the following gradient descent update:

$$x:=x−\eta\nabla f(x),x:=x−\eta\nabla f(x),$$

where the positive scalar η is called the learning rate or step size.

Question Two

When faced with a regression problem, why might linear regression, and specifically why might the least-squares cost function J, be a reasonable choice?

Let us consider a set of probabilistic assumptions, under which least-squares regression is derived as a very natural algorithm. Again let us assume that the target variables and the inputs are related via the equation $y^{(i)} = \theta^T x^{(i)} + \varrho (i)$, where $\varrho (i)$ is an error term that captures either unmodeled effects (such as if there are some features very pertinent to predicting housing price, but that we'd left out of the regression), or random noise. Let us

further assume that the ϙ (i) are distributed IID (independently and identically distributed) according to a Gaussian distribution (also called a Normal distribution) with mean zero and some variance σ 2 .

We can write this assumption as "ϙ (i) ∼ N (0, σ2)." I.e., the density of ϙ (i) is given by:

$p(ϙ^{(i)}) = 1 \sqrt{2\pi\sigma} \exp(-(ϙ^{(i)})^2 / 2\sigma^2)$.

This implies that:

$p(y^{(i)} | x^{(i)} ; \theta) = 1 \sqrt{2\pi\sigma} \exp(-(y^{(i)} - \theta^T x^{(i)})^2 / 2\sigma^2)$.

The notation "p(y (i) |x (i) ; θ)" indicates that this is the distribution of $y^{(i)}$ given $x^{(i)}$ and parameterized by θ. Note that we should not condition on θ ("p(y (i) |x (i) , θ)"), since θ is not a random variable. We can also write the distribution of $y^{(i)}$ as as:

y (i) | x (i) ; θ ∼ N ($\theta^T x^{(i)}$, σ2). Given X (the design matrix, which contains all the x (i) 's) and θ, what is the distribution of the y (i) 's? The probability of the data is given by p(~y|X; θ). This quantity is typically viewed a function of ~y (and perhaps X), for a fixed value of θ. When we wish to explicitly view this as a function of θ, we will instead call it the likelihood function: L(θ) = L(θ; X, ~y) = p(~y|X; θ). Note that by the independence assumption on the ϙ (i) 's (and hence also the y (i) 's given the x (i) 's), this can also be written:

$L(\theta) = \Pi_{i=1}^m p(y^{(i)} | x^{(i)} ; \theta)$

$= \Pi_{i=1}^m 1/\sqrt{2\pi\sigma} \exp(-(y^{(i)} - \theta^T x^{(i)})^2 / 2\sigma 2)$.

Now, given this probabilistic model relating the y (i) 's and the x (i) 's, what is a reasonable way of choosing our best guess of the parameters θ?

The principal of maximum likelihood says that we should choose θ so as to make the data as high probability as possible. I.e., we should choose θ to maximize L(θ). Instead of maximizing L(θ), we can also maximize any strictly increasing function of L(θ). In particular, the derivations will be a bit simpler if we instead maximize the log likelihood ℓ(θ):

$\ell(\theta) = \log L(\theta)$

$= \log \Pi_{i=1}^m 1/\sqrt{2\pi\sigma} \exp(-(y^{(i)} - \theta^T x^{(i)})^2 / 2\sigma^2)$ S

$= \sum_{i=1}^m \log 1/\sqrt{2\pi\sigma} \exp(-(y^{(i)} - \theta^T x^{(i)})^2 / 2\sigma 2)$

$= m \log 1 \sqrt{2\pi\sigma} - 1/\sigma 2 \cdot 1/2 \sum_{i=1}^m (y^{(i)} - \theta^T x^{(i)})^2$.

Hence, maximizing ℓ(θ) gives the same answer as minimizing $1/2 \sum_{i=1}^m (y^{(i)} - \theta^T x^{(i)})^2$, which we recognize to be J(θ), our original least-squares cost function.

To summarize: Under the previous probabilistic assumptions on the data, least-squares regression corresponds to finding the maximum likelihood estimate of θ. This is thus one set of assumptions under which least-squares regression can be justified as a very natural method that's just doing maximum likelihood estimation.

Question Three

First, the derivative of ln(x): $d/dx \ln(x) = 1/x$

Second, the (very elegant) derivative of the sigmoid: $d\sigma(z)/dz = \sigma(z)(1-\sigma(z))$

Finally, the chain rule of derivatives. Suppose we are computing the derivative of a composite function $f(x) = u(v(x))$. The derivative of $f(x)$ is the derivative of $u(x)$ with respect to $v(x)$ times the derivative of $v(x)$ with respect to x:

$$d f/dx = du/dv \cdot dv/dx$$

First, we want to know the derivative of the loss function with respect to a single weight w_j (we'll need to compute it for each weight, and for the bias):

$$\partial LL(w,b)/\partial w_j = \partial/\partial w_j -[y \log\sigma(w\cdot x+b) + (1-y)\log(1-\sigma(w\cdot x+b))]]$$

$$= -[\partial/\partial w_j \, y \log\sigma(w\cdot x+b) + \partial/\partial w_j \, (1-y)\log[1-\sigma(w\cdot x+b)]$$

Next, using the chain rule, and relying on the derivative of log:

$$\partial LL(w,b)/\partial w_j = -y/\sigma(w\cdot x+b) \, \partial/\partial w_j \, \sigma(w\cdot x+b) - 1-y/1-\sigma(w\cdot x+b) \, \partial/\partial w_j \, 1-\sigma(w\cdot x+b)$$

Rearranging terms:

$$\partial LL(w,b)/\partial w_j = -[\, y/\sigma(w\cdot x+b) - 1-y/1-\sigma(w\cdot x+b) \,] \, \partial/\partial w_j \, \sigma(w\cdot x+b)$$

And now plugging in the derivative of the sigmoid, and using the chain rule one more time, we end up with Eq:

$$\partial LL(w,b)/\partial w_j = -[\, y-\sigma(w\cdot x+b)/\sigma(w\cdot x+b)[1-\sigma(w\cdot x+b)]]\sigma(w\cdot x+b)[1-\sigma(w\cdot x+b)]\partial (w\cdot x+b)/\partial w_j$$

$$= -[\, y-\sigma(w\cdot x+b) \, \sigma(w\cdot x+b)[1-\sigma(w\cdot x+b)]]\sigma(w\cdot x+b)[1-\sigma(w\cdot x+b)]x_j$$

$$= -[y-\sigma(w\cdot x+b)]x_j$$

$$= [\sigma(w\cdot x+b)-y]x_j$$

Question Four

X_1	X_2	y
0	0	2
0	1	3
1	0	3
1	1	4

Gradient Descent Iteration #1

Let's start with the initial values of 0 for both coefficients.

$\theta_0=0$

$\theta_1=0$

$\theta_2=0$

$y = \theta_0+\theta_1*x_1+\theta_2* x_2$

$y = 0+0*x_1+0* x_2$

We can calculate the error for a prediction as follows:

error = p(i) – y(i)

Where p(i) is the prediction for the i'th instance in our dataset and y(i) is the i'th output variable for the instance in the dataset.

We can now calculate he predicted value for y using our starting point coefficients for the first training instance:

$X_1=0$, $x_2=0$,y=2

p(i) = 0 + 0 * 0+0*0

p(i) = 0

Using the predicted output, we can calculate our error:

error = 0 – 2

error = -2

We can now use this error in our equation for gradient descent to update the weights. We will start with updating the intercept first, because it is easier.

We can say that θ_0 is accountable for all of the error. This is to say that updating the weight will use just the error as the gradient. We can calculate the update for the θ_0 coefficient as follows:

$\theta_0(t+1) = \theta_0(t) – alpha * error$

Where $\theta_0(t+1)$ is the updated version of the coefficient we will use on the next training instance, θ_0 (t) is the current value for θ_0 alpha is our learning rate and error is the error we calculate for the training instance. Let's use a small learning rate of 0.5 and plug the values into the equation to work out what the new and slightly optimized value of θ_0 will be:

$\theta_0(t+1) = 0 – 0.5 * -2$

θ_0 (t+1) =1

Now, let's look at updating the value for θ_1. We use the same equation with one small change. The error is filtered by the input that caused it. We can update θ_1 using the equation:

$\theta_1(t+1) = \theta_1(t) – alpha * error * x_1$

Where $\theta_1(t+1)$ is the update coefficient, θ_1 (t) is the current version of the coefficient, alpha is the same learning rate described above, error is the same error calculated above and x is the input value.

We can plug in our numbers into the equation and calculate the updated value for θ_1:

$\theta_1(t+1) = 0 – 0.5 * -2 * 0$

$\theta_1(t+1) = 0$

Now, let's look at updating the value for θ_2. We use the same equation with one small change. The error is filtered by the input that caused it. We can update θ_2 using the equation:

$\theta_2(t+1) = \theta_2(t) – alpha * error * x_2$

Where $\theta_2(t+1)$ is the update coefficient, θ_2 (t) is the current version of the coefficient, alpha is the same learning rate described above, error is the same error calculated above and $_{x2}$ the input value.

We can plug in our numbers into the equation and calculate the updated value for θ_2:

$\theta_2(t+1) = 0 – 0.5 * -2 * 0$

$\theta_2(t+1) = 0$

We have just finished the first iteration of gradient descent and we have updated our weights to be $\theta_0=1$ and $\theta_1=0$ $\theta_2=0$

Gradient Descent Iteration #2

$X_1=0$, $x_2=1$,$y=3$ and $\theta_0=1$, $\theta_1=0$ and $\theta_2=0$

$y = \theta_0+\theta_1*x_1+\theta_2* x_2$

$p(i) = 1 + 0 * 0+0*0$

$p(i) = 1$

$error = 1 – 3$

$error = -2$

$\theta_0(t+2) = \theta_{t+1}(t)$-alpha * error

$\theta_0 (t+2) =1– 0.5 * -2$

$\theta_0(t+2)=2$

$\theta_1(t+2) = \theta_{t+1}(t) –$ alpha * error * x_1

$\theta_1(t+2) = 0 – 0.5 * -2 * 0$

$\theta_1(t+1) = 0$

$\theta_2(t+2) = \theta_{t+1}(t) –$ alpha * error * x_2

$\theta_2(t+2) = 0 – 0.5 * -2 * 1$

$\theta_2(t+1) = 1$

Gradient Descent Iteration #3

$X_1=1$, $x_2=0$,$y=3$ and $\theta_0=2$, $\theta_1=0$ and $\theta_2=1$

$y = \theta_0+\theta_1*x_1+\theta_2* x_2$

$p(i) = 2 + 0 * 1+1*0$

$p(i) = 2$

$error = 2 – 3$

$error = -1$

$\theta_0(t+3) = \theta_{t+2}(t)$-alpha * error

$\theta_0 (t+3) =2– 0.5 * -1$

$\theta_0(t+3) = 2 +0.5$

$\theta_0(t+3) = 2.5$

$\theta_1(t+3) = \theta_{t+2}(t) – \text{alpha} * \text{error} * x_1$

$\theta_1(t+3) = 0 – 0.5 * -1* 1$

$\theta_1(t+3) = 0.5$

$\theta_2(t+3) = \theta_{t+2}(t) – \text{alpha} * \text{error} * x_2$

$\theta_2(t+3) = 1 – 0.5 * -1 * 0$

$\theta_2(t+3) = 1$

Gradient Descent Iteration #4

$X_1=1$ $x_2=1$,$y=4$ and $\theta_0=2.5$, $\theta_1=0.5$ and $\theta_2=1$

$y = \theta_0+\theta_1*x_1+\theta_2* x_2$

$p(i) = 2.5 + 0.5 * 1+1*1$

$p(i) = 4$

$\text{error} = 4 – 4$

$\text{error} = 0$

$\theta_0(t+4) = \theta_{t+3}(t)\text{-alpha} * \text{error}$

$\theta_0 (t+4) =2.5– 0.5 * 0$

$\theta_0(t+4) = 2.5$

$\theta_1(t+4) = \theta_{t+3}(t) – \text{alpha} * \text{error} * x_1$

$\theta_1(t+4) = 0.5 – 0.5 * 0* 1$

$\theta_1(t+4) = 0.5$

$\theta_2(t+4) = \theta_{t+3}(t) - \text{alpha} * \text{error} * x_2$

$\theta_2(t+4) = 1 - 0.5 * 0 * 1$

$\theta_2(t+4) = 1$