## Overview

In this reading, you will practice implementing gradient descent in Octave as a warm-up to your programming assignment, while learning about a simple trick which is often useful in implementing ML algorithms. We will be implementing inverse-vector products using gradient descent.

Don't worry too much if you don't fully grasp some of the mathematical details right away. The main objective of this reading is to practice writing some Octave/Matlab code.

## Matrix Inverse-Vector Products

A common operation in ML algorithms is finding the inverse of a matrix $A$ (often denoted by $A^{-1}$). However, it turns out that computing the inverse of a big matrix can be computationally intensive. Fortunately, in many situations, we can use a simple trick to avoid computing the inverse of any matrix at all.

The key to this trick is that we rarely ever need the actual inverse $A^{-1}$. What we often want is the matrix inverse multiplied by some vector $b$ (i.e., we want $A^{-1}b$), we call $A^{-1}b$ the **matrix inverse-vector product**.

A good example of this is when we want to use the normal equations to solve a linear regression problem $\theta = (X^T X)^{-1}(X^T y)$. In this case, $A = X^T X$ and $b = X^T y$.

One way of calculating such products is to find $A^{-1}$ and then multiply that by $b$. However, we're going to explore a different way of computing $A^{-1}b$.

## Computing Matrix Inverse-Vector Products Using Gradient Descent

We'll start by trying to gain some intuition about computing the matrix inverse-vector product using gradient descent.

Say we are interested in finding $x = A^{-1}b$, this is equivalent to finding a vector $x$ such that $Ax = b$. We are going to do this by defining a cost function $f(x)$ such that $f(x)$ is minimized when $x = A^{-1}b$. We can then use gradient descent to minimize $f(x)$ and obtain the value of $A^{-1}b$.

The cost function that we're going to use is the familiar squared error loss function:

$$f(x) = \|Ax - b\|^2 = \sum_i^n \left[(Ax)_i - b_i\right]^2$$

## Exercise 1 Cost Function:

Implement the cost function $f(x)$ in Octave below. Your function should take in $A$, $b$ and $x$ and return $\|Ax - b\|^2$.

**Hint:** The Octave function $\mathrm{norm}(v)$ gives you $\|v\|$ (where $v$ is any vector).

```
1  function cost = squaredErrorCost(A, b, x)
2    % Your code here
3    cost = norm(A*x-b)^2
4  endfunction
```

Run

Reset

```
Good job!
```

You may recognize that this function is convex, this means that it has a global minimum (when $x = A^{-1}b$) and that gradient descent will eventually find this minimum point. The gradient of $f(x)$ is

$$\frac{df(x)}{dx} = 2A(Ax - b)$$

Now that we have the gradient of $f(x)$ we can use gradient descent to minimize $f(x)$. Recall that a gradient update (or gradient step) is an operation where we move $x$ a little bit in the direction of steepest descent, or equivalently in the opposite direction of the gradient.

## Exercise 2 Gradient Descent:

Implement a function that computes $A^{-1}b$ using gradient descent.

This function should take in $A$, $b$, $x\_\mathrm{init}$ (an initial value for x) and $\mathrm{alpha}$ (the step size) and perform gradient updates on $x$ until $\|Ax - b\|^2 < 10^{-6}$. Recall that a gradient update or gradient step is defined as:

$$x_{\mathrm{new}} = x - \mathrm{alpha} \cdot \frac{df(x)}{dx}$$

```
function A_inv_b = matrixInverseVector(A, b, x_init, alpha)
  % Your code here
  x = x_init - alpha * (2*A*(A*x_init-b));
  err = norm(A*x-b)^2; % initialize error
  while err >= 10^(-6),
    x_new = x - alpha * (2*A*(A*x-b));
    err = norm(A*x_new-b)^2;
    x = x_new;
  end;
  A_inv_b = x;
endfunction
```

Run

Reset

```
Good job!

Test Case 1: Difference between pinv(A)*b and matrix inverse product is 9.51e-05
Test Case 2: Difference between pinv(A)*b and matrix inverse product is 9.47e-05
Test Case 3: Difference between pinv(A)*b and matrix inverse product is 9.54e-05
Test Case 4: Difference between pinv(A)*b and matrix inverse product is 9.50e-05
Test Case 5: Difference between pinv(A)*b and matrix inverse product is 9.74e-05
```

## Conclusion

In practice, we do not usually use gradient descent to solve for $x$ such that $Ax = b$ (we can compute $x$ using other more efficient methods). However, the general "trick" of avoiding the need to compute or store $A^{-1}$ is still applicable in many situations.

In fact, this is such a common operation that there is a built-in function in Octave/Matlab – the $\backslash$ operator – to do this for you. In Octave/Matlab $A\backslash b$ will give you the value of $A^{-1}b$. Of course, you can still also use the $\mathrm{pinv}$ function to compute $A^{-1}$ if you prefer.

Mark as completed