Note: this homework has been updated to remove the backprop problem, which now appears in homework 4.

Just like before, your homework submission must contain the following to receive full credit:

- A single script called "hmwk3.m" or "hmwk3.py" that requires no arguments and prints the things I ask for in bold below. Each output should be labeled.
- A pdf called hmwk1_results.pdf with your solutions to all theory problems and also the text output from the hmwk1 script. See the posted homework example.
- As many other code files as you wish - but only things you wrote or solutions to previous homeworks.

1. (a) Write a function with signature

```
function iscorrect = check_gradient(f, grad, x0)
```

Arguments:

- `f` : A function $\mathbb{R}^\Omega \to \mathbb{R}$ that maps an array or arbitrary dimensions into the reals.
- `grad` : A function $\mathbb{R}^\Omega \to \mathbb{R}^\Omega$ that maps an array or arbitrary dimensions into an array of the same dimensions.
- `x0` : A vector/array in $\mathbb{R}^\Omega$.

Returns: A boolean that is true if `grad` generates the gradient of `f`.

Details: Your code should test whether `grad` generates the gradient of `f`. It should do this by generating a random perturbation δ and then testing whether

$$f(x+\delta) - f(x) \approx \langle \delta, \nabla f(x) \rangle.$$

The method should generate a *random Gaussian* δ with the same norm as `x0`, check the gradient condition, and then replace $\delta \leftarrow \delta/10$. Do this for 10 different orders of magnitude of δ. For each order, print out the ratio $\|\delta\|/\|x0\|$, and print out the *relative* error between the left and right side of the above equation. Finally, the last line of output should print the minimum relative error achieved. All of the outputs should be labeled. The method returns `true` if the gradient is correct up to reasonable error, and `false` otherwise.

(b) **In your hmwk1 script, test your gradient checker on the function**

$$f(x) = \frac{1}{2}\|Ax - b\|^2$$

for random $A \in \mathbb{R}^{3\times 3}$ and $b \in \mathbb{R}^{3\times 5}$, and capture the output in the console.

2. All of your solutions to these problems should only require a few lines of code.

   (a) Write a function with signature

   ```
   function f = l1_eps(x,eps)
   ```

   that returns the smoothed ℓ_1 norm

   $$|x|_\epsilon = \sum_i \sqrt{x_i^2 + \epsilon^2}.$$

   Your function must handle arguments `x` of arbitrary dimensions (i.e. 1d, 2d, 3d, etc...).

   (b) Write a function with signature

   ```
   function f = l1_grad(x,eps)
   ```

   that returns the gradient of the smoothed ℓ_1 norm you created in part (a).

   (c) Write a function with signature

   ```
   function f = tv_objective(x, b, mu, eps)
   ```

   Arguments:

   - `x` : A 2d array.
   - `b` : A 2d array containing a noisy image.
   - `mu` : A scalar, non-negative scaling parameter.

   Returns: The value of

   $$f(x) = \mu|\nabla_d x|_\epsilon + \frac{1}{2}\|x - b\|^2$$

   where $|\cdot|_\epsilon$ is the smoothed ℓ_1 norm, and ∇ denotes the discrete gradient. This function should only be a few lines of code, and you should use the function `grad2d` from your last homework assignment.

   (d) Write a function with signature

   ```
   function f = tv_grad(x, b, mu, eps)
   ```

   that produces the gradient $\nabla f(x)$. You should use your `grad2d` and `div2d` functions from the last assignment. Remember - this should only be a few lines of code or you did something wrong.

   (e) Build a noisy test image `b`, a random 2d input `x`, and set `mu=0.5`. Build a function handle that accepts `x`, and returns the objective. Build another function handle that accepts `x`, and returns the gradient. **Using these handles, test your gradient by calling `check_gradient`, and capture results in the console.**

   If your method doesn't pass the gradient checker, then make sure you didn't make any of the following common errors:

   - You remembered to square the norm in the objective, right?
   - You're using the Frobenius norm, not the 2 norm, right?
   - You remembered to multiply by μ in both objective and gradient functions, right?
   - You remembered the factor of $\frac{1}{2}$ in the objective, right?

   The most common reason to fail the gradient test is because your implementations for ∇ and ∇^T are not correct adjoints. You should have verified your adjoint using your adjoint checker in the last assignment.

3. (a) Create a function with signature

```
function y = logreg_grad(x,D,c)
```

that produces the gradient of the function `logreg_objective` from your last homework assignment.

(b) **Test your gradient by calling `create_classification_problem(1000,10,5)` to produce test data and labels. Then feed the resulting function and gradient into your function `check_gradient`.**