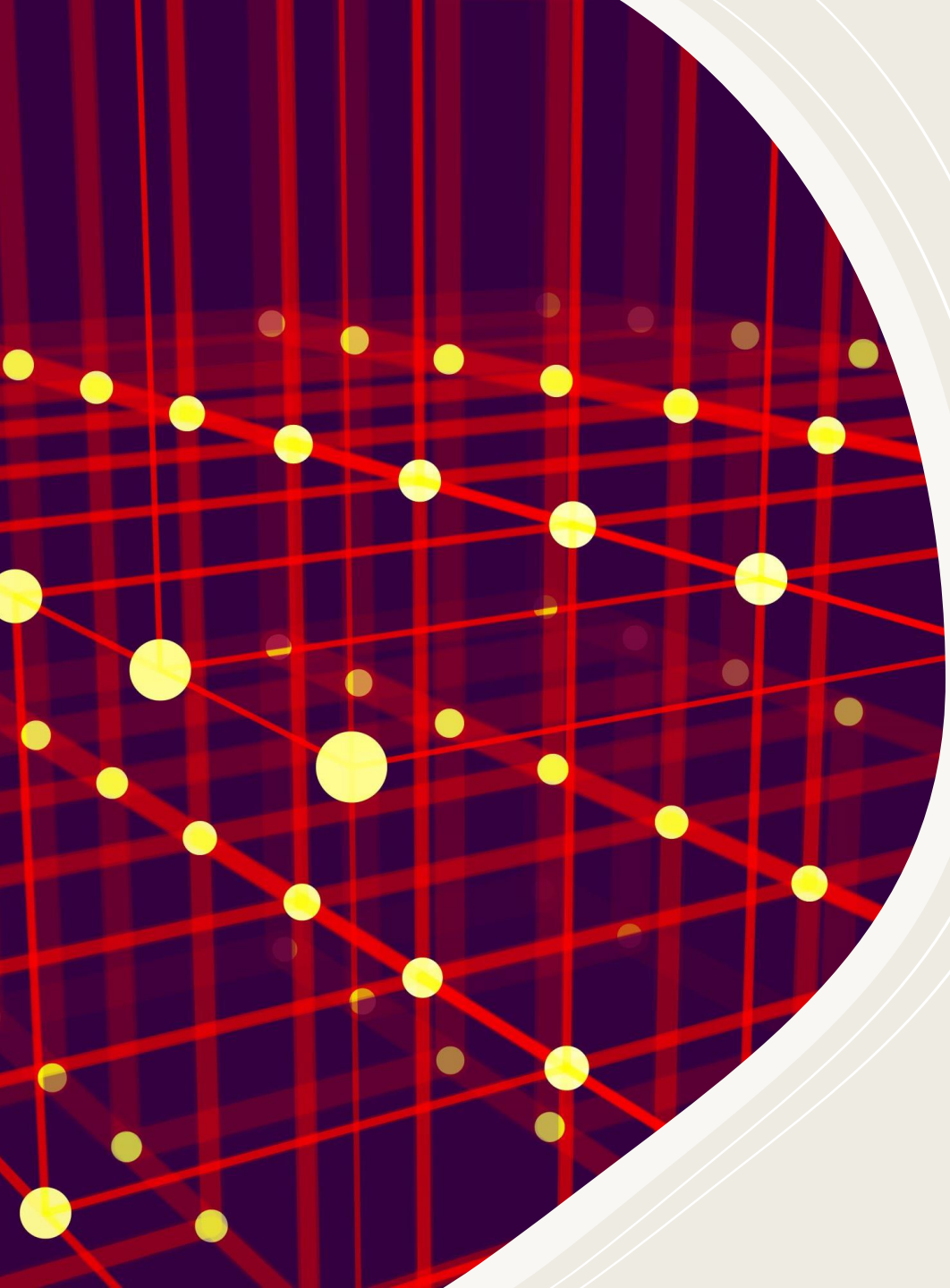

CPSC 340 – Tutorial 1

Lironne Kurzman
lironnek@cs.ubc.ca

University of British Columbia

September 13th, 2021

GITHUB – tl;dr

Clone a repo:

```
git clone <repo url>
```

See changes:

```
git status
git diff <file name>
```

Commit changes:

```
git add <file name>
git commit -m "<commit message>" git push
```

Pull remote changes:

```
git pull
```

Notation and Gradients

- Define: $\mathbb{R}^n$, $x^T Ax$, $\|x\|$, $\nabla f(x)$
- If $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, $A \in \mathbb{R}^{n \times m}$ find the dimensions of:
  - $X^T A$, $x^T Ay$, xx^T.
- If $a, x \in \mathbb{R}^n$, compute the gradients of:

  $f(x) = a^T x$,

  $f(x) = \log a^T x$.
- A helpful sanity check:
  If x is a length-n (column) vector, and $f : \mathbb{R}^n \to \mathbb{R}$ then
  $\nabla f(x)$ is a length-n (row) vector.
- Entropy: Given a probability vector $p = [p_1, \ldots, p_n]^T$,

$$entropy(p) = -\sum_{i=1}^{n} p_i \log(p_i).$$

Python/NumPy Basics

- numpy Arrays:

  `A = np.array(...)`
- Array slicing
- Element-wise operations:

  `+, -, *, /, np.amin, np.amax`
- Array shapes
- Matrix/vector operations:

  `np.transpose(), @`
- Other useful things:

  `np.ones, np.zeros`

Python Demo

Numpy Basics

Bonus: We can check that the gradient implementation is correct

Finite-difference approximation of the gradient

$\epsilon = 1e-4$

Gradient definion:

$$\frac{\partial f(x)}{\partial x_i} = \lim_{\epsilon \to 0} \frac{f(x_i + \epsilon) - f(x_i)}{\epsilon}$$

- Therefore,

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x_i + \epsilon) - f(x_i)}{\epsilon}$$

Bonus: Code to approximate the gradient

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x_i + \epsilon) - f(x_i)}{\epsilon}$$

Create a simple gradient check for single-variable function

```
def forward_diff(x, f):
    # Approximates gradient  eps = 1e-4
    g_approx = f(x + eps) - f(x)  g_approx /= eps
```

Bonus: Code to approximate the gradient

Using scipy:

```
import numpy as np
from scipy.optimize
import approx_fprime

approx = approx_fprime(x0, foo, 1e-4)
exact = foo_grad(x0)

print("My gradient     : %s" % exact)
print("Scipy's gradient: %s" % approx)

# Assert that the two are almost equal
np.testing.assert_almost_equal(approx, exact, 3)
```