Brief Review of Linear Algebra (Part 2)

Content and structure mainly from: http://www.deeplearningbook.org/contents/linear_algebra.html (http://www.deeplearningbook.org/contents/linear_algebra.html)

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Singular matrices are similar to zeros

- Informally, singular matrices are matrices that do not have an inverse (similar to the idea that 0 does not have an inverse)
- Consider the 1D equation $ax = b$
  - Usually we can solve for x by multiplying both sides by $1/a$
  - But what if $a = 0$?
  - What are the solutions to the equation?
- Called "singular" because a random matrix is unlikely to be singular just like choosing a random number is unlikely to be 0.

In [2]:
```
from numpy.linalg import LinAlgError
import sys
def try_inv(A):
    print('A = ')
    print(np.array(A))
    try:
        #np.linalg.inv(A)
        if np.linalg.cond(A) >= 1/sys.float_info.epsilon:
            raise LinAlgError('Singular matrix')
    except LinAlgError as e:
        print(e)
    else:
        print('Not singular!')
    print()

try_inv([[0, 0], [0, 0]])
try_inv(np.eye(3))
try_inv([[1, 1], [1, 1]])
try_inv([[1, 10], [1, 10]])
try_inv([[1, 1], [10, 10]])
try_inv([[2, 20], [4, 40]])
try_inv([[2, 20], [40, 4]])
```

```
A =
[[0 0]
 [0 0]]
Singular matrix

A =
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
Not singular!

A =
[[1 1]
 [1 1]]
Singular matrix

A =
[[ 1 10]
 [ 1 10]]
Singular matrix

A =
[[ 1  1]
 [10 10]]
Singular matrix

A =
[[ 2 20]
 [ 4 40]]
Singular matrix

A =
[[ 2 20]
 [40  4]]
Not singular!
```

In [3]:
```
# Random matrix is very unlikely to be 0
for j in range(10):
    try_inv(np.random.randn(2, 2))
```

```
A =
[[-1.19288666  0.57476849]
 [-0.40565994  1.88308994]]
Not singular!

A =
[[ 1.44548064  1.0585672 ]
 [-1.85994904 -0.00948361]]
Not singular!

A =
[[-0.61352128 -0.42713732]
 [-1.23231451  0.17235688]]
Not singular!

A =
[[ 0.47719912  0.47401289]
 [ 0.30125177 -0.68290875]]
Not singular!

A =
[[ 1.21666448  1.40321413]
 [ 0.15077117 -1.00406009]]
Not singular!

A =
[[ 0.43500445 -1.12063009]
 [ 0.53696971 -2.11952602]]
Not singular!

A =
[[-2.09804668 -0.43895893]
 [ 2.15475232  1.0557288 ]]
Not singular!

A =
[[ 0.25135967  1.47775309]
 [ 0.12797278 -0.61409801]]
Not singular!

A =
[[-0.72279829  0.81497235]
 [-0.5753405   0.87173775]]
Not singular!

A =
[[ 1.6028106  -0.18708103]
 [ 0.31560633  0.95107288]]
Not singular!
```

Norms: The "size" of a vector or matrix

- Informally, a generalization of the absolute value of a scalar
- Formally, a norm is an function f that has the following three properties:
  - $f(\mathbf{x}) = 0 \Rightarrow \mathbf{x} = \mathbf{0}$ (zero point)
  - $f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$ (Triangle inequality)
  - $\forall \alpha \in \mathbb{R}, f(\alpha \mathbf{x}) = |\alpha| f(\mathbf{x})$ (absolutely homogenous)
- Examples
  - Absolute value of scalars
  - L^p (also denoted ℓ_p) norm

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{d} |x_i|^p \right)^{\frac{1}{p}}$$

  - (Discussion) What does this represent when $p = 2$ (for simplicity you can assume $d = 2$)?
    - When $p = 2$, we often merely denote as $\|\mathbf{x}\|$.
  - What about when $p = 1$?
  - What about when $p = \infty$ (or more formally the limit as $p \to \infty$)?

In [4]:
```
x = np.array([1, 1])
print(np.linalg.norm(x, ord=2))
print(np.linalg.norm(x, ord=1))
print(np.linalg.norm(x, ord=np.inf))
```

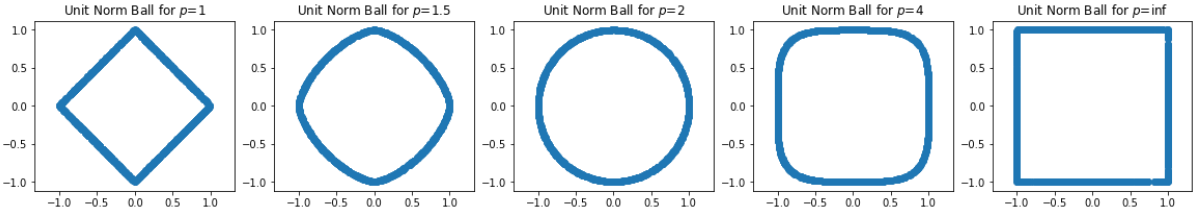
```
1.4142135623730951
2.0
1.0
```

Vectors that have the same norm form a "ball" that isn't necessarily circular

In [5]:
```
rng = np.random.RandomState(0)
X = rng.randn(1000, 2)

p_vals = [1, 1.5, 2, 4, np.inf]
fig, axes = plt.subplots(1, len(p_vals), figsize=(len(p_vals)*4, 3))

for p, ax in zip(p_vals, axes):
    # Normalize them to have the unit norm
    Z = (X.T / np.linalg.norm(X, ord=p, axis=1)).T
    ax.scatter(Z[:, 0], Z[:, 1])
    ax.axis('equal')
    ax.set_title('Unit Norm Ball for $p$=%g' % p)
```



If $p = 0.5$, then the equation below is not a norm.

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{d} |x_i|^p \right)^{\frac{1}{p}}$$

(Discussion) Can you provide a counterexample to one of the properties that does not hold for $p = 0.5$?

- $f(\mathbf{x}) = 0 \Rightarrow \mathbf{x} = \mathbf{0}$ (zero point)
- $f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$ (Triangle inequality)
- $\forall \alpha \in \mathbb{R}, f(\alpha \mathbf{x}) = |\alpha| f(\mathbf{x})$ (absolutely homogenous)

Squared L^2 norm is quite common since it simplifies to a simple summation

$$\|\mathbf{x}\|_2^2 = \left(\left(\sum_{i=1}^{d} |x_i|^2 \right)^{\frac{1}{2}} \right)^2 = \sum_{i=1}^{d} |x_i|^2 = \sum_{i=1}^{d} x_i^2$$

- Additionally, this can be computed as $\|\mathbf{x}\|_2^2 = \mathbf{x}^T \mathbf{x}$
- Informally, this is analogous to taking the square of a scalar number

In [6]:
```
x = np.arange(4)
print(np.linalg.norm(x, ord=2)**2)
print(np.dot(x, x))
```

```
14.0
14
```

Orthogonal vectors

- Orthogonal vectors are vectors such that $\mathbf{x}^T\mathbf{y} = 0$
- The dot product between vectors can be written in terms of norms and the cosine of the angle:

$$\mathbf{x}^T\mathbf{y} = \|\mathbf{x}\|_2\|\mathbf{y}\|_2 \cos\theta$$

- (Discussion) Suppose $\mathbf{x}$ and $\mathbf{y}$ are non-zero vectors, what must θ be if the vectors are orthogonal?

In [7]:
```
print(np.dot([0, 1], [1, 0]))
theta = np.pi/2
x = np.array([np.cos(theta), -np.sin(theta)])
y = np.array([np.sin(theta), np.cos(theta)])
print(x)
print(y)
print(np.dot(x, y))
```

```
0
[ 6.123234e-17 -1.000000e+00]
[1.000000e+00 6.123234e-17]
0.0
```

Special matrices: Orthogonal matrices

- Informally, an orthogonal matrix only rotates (or reflects) vectors around the origin (zero point), but does not change the size of the vectors.
- Informally, almost analagous to a 1 for matrices but more general
- A *square* matrix such that $Q^TQ = QQ^T = I$
- Or, equivalently $Q^{-1} = Q^T$
- Or, equivalently:
  - Every column (or row) is orthogonal to every other column (or row)
  - Every column (or row) has unit L^2 norm, i.e., $\|Q_{i,:}\|_2 = \|Q_{:,j}\|_2 = 1$

In [8]:
```
print('Identity matrix')
Q = np.eye(2) # Identity
print(Q)
print(np.allclose(np.eye(2), np.dot(Q.T, Q)))

print('Reflection matrix')
Q = np.array([[1, 0], [0, -1]]) # Reflection
print(Q)
print(np.allclose(np.eye(2), np.dot(Q.T, Q)))

print('Rotation matrix')
theta = np.pi/3
Q = np.array([
    [np.cos(theta), -np.sin(theta)],
    [np.sin(theta), np.cos(theta)]
])
print(Q)
print(np.allclose(np.eye(2), np.dot(Q.T, Q)))
```

```
Identity matrix
[[1. 0.]
 [0. 1.]]
True
Reflection matrix
[[ 1  0]
 [ 0 -1]]
True
Rotation matrix
[[ 0.5       -0.8660254]
 [ 0.8660254  0.5      ]]
True
```

Other special matrices: Symmetric, Triangular, Diagonal

- Symmetric matrices are symmetric around the diagonal; formally, $A = A^T$
- Triangular matrices only have non-zeros in the upper or lower triangular part of the matrix
- Diagonal matrices only have non-zeros along the diagonal of a matrix

In [9]:
```
A = np.arange(25).reshape(5, 5)+1
print('Symmetric')
print(A + A.T)
print('Upper triangular')
print(np.triu(A))
print('Lower triangular')
print(np.tril(A))
print('Diagonal (both upper and lower triangular)')
print(np.diag(np.arange(5) + 1))
```

```
Symmetric
[[ 2  8 14 20 26]
 [ 8 14 20 26 32]
 [14 20 26 32 38]
 [20 26 32 38 44]
 [26 32 38 44 50]]
Upper triangular
[[ 1  2  3  4  5]
 [ 0  7  8  9 10]
 [ 0  0 13 14 15]
 [ 0  0  0 19 20]
 [ 0  0  0  0 25]]
Lower triangular
[[ 1  0  0  0  0]
 [ 6  7  0  0  0]
 [11 12 13  0  0]
 [16 17 18 19  0]
 [21 22 23 24 25]]
Diagonal (both upper and lower triangular)
[[1 0 0 0 0]
 [0 2 0 0 0]
 [0 0 3 0 0]
 [0 0 0 4 0]
 [0 0 0 0 5]]
```