Correlated features

Consider the following set of examples with 2 features

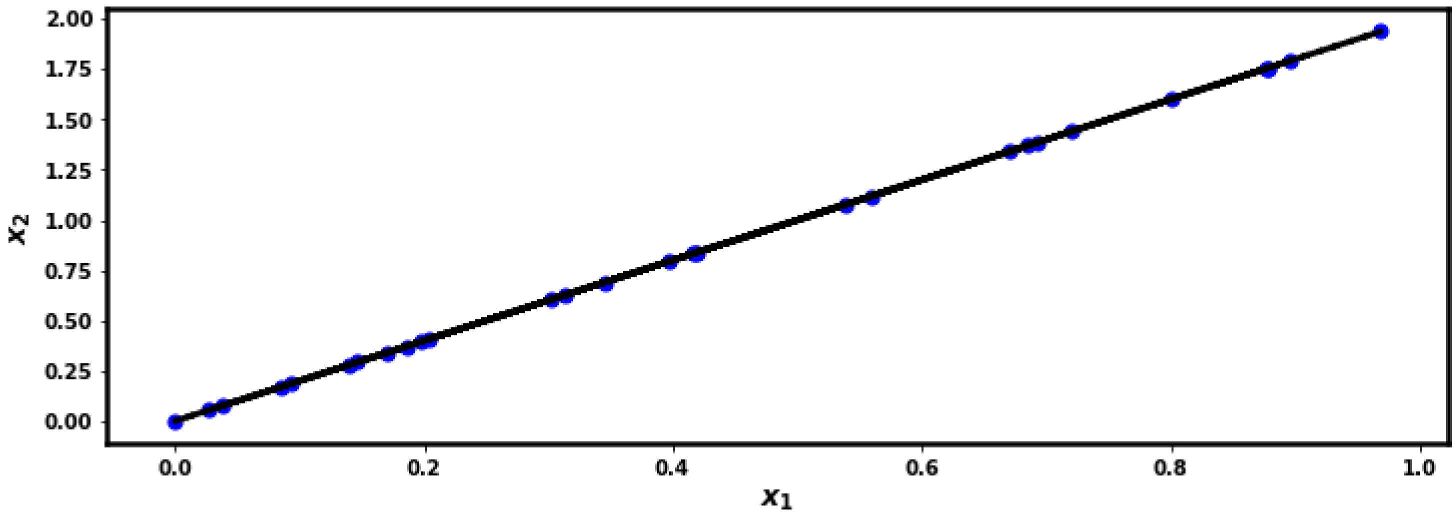
In [4]:

```
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline

m = 30
rng = np.random.RandomState(1)
x_1 = rng.rand(m)
x_2 = 2 * x_1

fig, ax = plt.subplots(1,1,figsize=(12,4))
_= ax.scatter( x_1, x_2, color="blue", s=50)
_= ax.plot( x_1, x_2, color="black", linestyle="dashed")
_= ax.set_xlabel("$x_1$")
_= ax.set_ylabel("$x_2$")
```



As you can see

- $\mathbf{x}_2$ is perfectly correlated with $\mathbf{x}_1$

$$\mathbf{x}_2^{(\mathbf{i})} = 2 * \mathbf{x}_1^{(\mathbf{i})}$$

A way to conceptualize $\mathbf{x^{(i)}}$

- As a point in the space spanned by unit basis vectors
$$\mathbf{u}_{(1)} = (1, 0)$$
$$\mathbf{u}_{(2)} = (0, 1)$$
- With $\mathbf{x^{(i)}}$ having exposure
$$\mathbf{x}^{(\mathbf{i})}_1 \text{ to } \mathbf{u}_{(1)}$$
$$\mathbf{x}^{(\mathbf{i})}_2 \text{ to } \mathbf{u}_{(2)}$$

So example $\mathbf{x^{(i)}}$ is

$$\mathbf{x^{(i)}} = \sum_{j'=1}^{2} \mathbf{x}^{(\mathbf{i})}_{j'} * \mathbf{u}_{(j')}$$

But because

$$\mathbf{x}_2^{(\mathbf{i})} = 2 * \mathbf{x}_1^{(\mathbf{i})}$$

we can create an *alternate* basis vector

$$\tilde{\mathbf{v}}_{(1)} = (1, 2)$$

such that example $\mathbf{x}^{(\mathbf{i})}$ is

$$\mathbf{x}^{(\mathbf{i})} = \tilde{\mathbf{x}}_1^{(\mathbf{i})} * \tilde{\mathbf{v}}_{(1)}$$

where $\tilde{\mathbf{x}}_1^{(\mathbf{i})} = \mathbf{x}_1^{(\mathbf{i})}$

That is, $\mathbf{x}^{(\mathbf{i})}$ has exposure $\tilde{\mathbf{x}}_1^{(\mathbf{i})}$ to the new, single basis vector.

So

- Rather than representing $\mathbf{x}^{(\mathbf{i})}$ as a vector with 2 features (in the original basis)
- We can represent it as $\tilde{\mathbf{x}}^{(\mathbf{i})}$, a vector with 1 feature (in the new basis)

This is the essence of dimensionality reduction

- Changing bases to one with fewer basis vectors

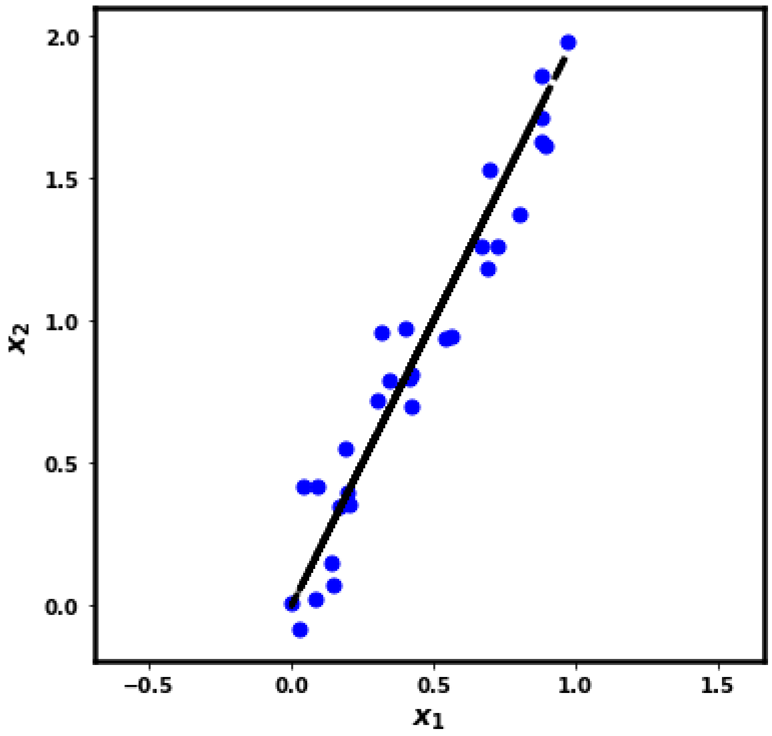
It is rarely the case for features to be perfectly correlated

In [5]:

```
eps = .01
x_2p = 2 * x_1 + .2 * rng.randn( x_1.shape[0] )

x_p = np.concatenate( [ x_1.reshape(-1,1), x_2p.reshape(-1,1)], axis=1)

fig, ax = plt.subplots(1,1,figsize=(6,6))
_= ax.scatter( x_p[:,0], x_p[:,1], color="blue", s=50)
_= ax.plot( x_1, x_2, color="black", linestyle="dashed")
_= ax.set_xlabel("$x_1$")
_= ax.set_ylabel("$x_2$")
_= ax.axis("equal")
```



In this case

- A second basis vector $\tilde{\mathbf{v}}_{(2)}$
- Orthogonal to the first

$$\tilde{\mathbf{v}}_{(1)} \cdot \tilde{\mathbf{v}}_{(2)} = 0$$

could approximate $\mathbf{x}^{(\mathbf{i})}$

$$\mathbf{x}^{(\mathbf{i})} = \sum_{j'=1}^{2} \tilde{\mathbf{x}}_{j'}^{(\mathbf{i})} * \tilde{\mathbf{v}}_{(j')}$$

In [6]:
```
from sklearn.decomposition import PCA
import math

pca_x2p = PCA()
#x_p = x_p - x_p.mean(axis=0)

pca_x2p_proj = pca_x2p.fit_transform(x_p)


def draw_vector(v0, v1, ax=None):
    arrowprops=dict(arrowstyle='->',
                    linewidth=2,
                    color="black",
                    shrinkA=0, shrinkB=0)
    _ = ax.annotate('', v1, v0, arrowprops=arrowprops)

    return ax

fig, ax = plt.subplots(1,1, figsize=(6,6))
mean = x_p.mean(axis=0)

maxp = np.sqrt( pca_x2p.explained_variance_[-1] )

_= ax.scatter( x_p[:,0], x_p[:,1], color="blue", s=10)

for i in range(0, 2):
    comp, length = pca_x2p.components_[i], pca_x2p.explained_variance_[i]
    v = comp  # *  np.sqrt(length)
    _= draw_vector( mean, mean + v , ax=ax)



_= ax.scatter( mean[0], mean[1], s=50, color="black")

_= ax.axis("equal")
```

The black lines represent the alternate basis vectors $\tilde{\mathbf{v}}_{(1)}, \tilde{\mathbf{v}}_{(2)}$.

As you can see:

- The variation along $\tilde{\mathbf{v}}_{(1)}$ is much greater than that around $\tilde{\mathbf{v}}_{(2)}$
- Capturing the notion that the "main" relationship is along $\tilde{\mathbf{u}}_{(1)}$

In fact, if we dropped $\tilde{\mathbf{v}}_{(2)}$ such that $||\tilde{\mathbf{x}}|| = 1$

- The examples would be projected onto $\tilde{\mathbf{v}}_{(1)}$
- With little information being lost

Subsets of correlated features

It may not be the case that a group of features is correlated across *all* examples

Consider the MNIST digits

- The subset of examples corresponding to the digit "1"
- Have a particular set of correlated features (forming a vertical column of pixels)
- Which *may not* be correlated with the same features in examples corresponding to *other* digits

Thus, a synthetic feature encodes a "concept" that occurs in many but not all examples

The "concept" will be *discovered*

- It may not necessarily be the pattern of features that corresponds to an entire digit
- It may be a partial pattern common to several digits
    - Vertical band (0, 1, 4, 7)
    - Horizontal band at top (5, 7, 9)

In [7]:

```python
print("Done")
```

```
Done
```