1.5. Broadcasting and the softmax function

- x_exp = np.exp(x)

x_sum = np.sum(x_exp, axis = 1, keepdims = True)

s = x_exp / x_sum → Works due to broadcasting

2. Vectorisation

- Vectorised implementation is cleaner and more efficient

2.1. L1 & L2 loss functions

- $L_1(\hat{y}, y) = \sum_{i=0}^{m} |y^{(i)} - \hat{y}^{(i)}|$

loss = np.sum(abs(y - yhat))

- $L_2(\hat{y}, y) = \sum_{i=0}^{m} (y^{(i)} - \hat{y}^{(i)})^2$

loss = np.sum((y - yhat)**2)

Notebook: Log. Reg. w/ a NN mindset

- Image recognition: cat classifier
- h5py: package to interact w/ a dataset that's stored on an H5 file
- train. set of m_train images (y=1 or y=0)
- test set of m_test images (cat or non-cat)
- image shape (num_px, num_px, 3) (square)
- _orig (for preprocessing)
- Reshape: X_flatten = X.reshape(X.shape[0], -1).T
- Common preprocessing steps:
  - Find dimensions and shapes of the problem (m_train, m_test, ...)
  - Reshape so that each example is a row of size (num_px * num_px * 3, ...)
  - "Standardise" the data