Summary so far

We have been looking at using linear models to do our modeling. So far, we have looked at:

- The line of best fit (using the normal equations)
- Linear functions of best fit (linear in the parameters)
- Linear functions of the general form:

$$\mathbf{y} = W\mathbf{x} + \mathbf{b}$$

  where the data points $\mathbf{x}$, $\mathbf{y}$ are known, and the matrix W and vector $\mathbf{b}$ are the unknowns. There are two ways of solving for the unknowns:

  - Batch training: Solve using least-squares error: $E = \sum_{k=1}^{p} \|\mathbf{t}_k - \mathbf{y}_k\|^2$ where $\mathbf{t}_i$ is the *desired* image of $\mathbf{x}_i$, and $\mathbf{y}_i$ is the image: $\mathbf{y}_i = W\mathbf{x}_i + \mathbf{b}$.

    In Matlab, we'll need to first change $W\mathbf{x} + \mathbf{b}$ to $\hat{W}\hat{\mathbf{x}}$ before solving using the slash command.
  - Online training: In this case, we approximate the least-squares solution by updating the weights and biases a small amount for each pair $\mathbf{x}_i, \mathbf{t}_i$ presented. This requires running through all of the data multiple times in order to get a good set of W, $\mathbf{b}$.

General Affine Maps

We saw that mappings of the form $\mathbf{y} = W\mathbf{x} + \mathbf{b}$ actually have a biological basis- the so-called "linear neural network". Under very simple assumptions, a neural network implements the affine map by physiologically determining weights W and biases $\mathbf{b}$ to give a desired input-output relationship[1].

From a biological viewpoint, we began with Hebb's rule (below, left)- Mathematically, it has shortcomings, and these led us to the Widrow-Hoff algorithm (below, right).

Hebb's Rule (Does not use target values)		Widrow-Hoff
$W^{\text{new}} = W^{\text{old}} + \alpha \mathbf{y}_i \mathbf{x}_i^T$	$\Rightarrow$	$W^{\text{new}} = W^{\text{old}} + \alpha\left(\mathbf{t}_i - \mathbf{y}_i\right)\mathbf{x}_i^T$

In these equations, we assume $\mathbf{t}_i$ are the **desired** outputs of our function, and $\mathbf{y}_i$ are the actual outputs, $\mathbf{y}_i = W\mathbf{x}_i + \mathbf{b}$.

[1] Actually, this statement is false. Why? (Hint: Do we expect all of our data to lie on a line or plane?)

Matlab Details

- For batch training, if we have p input vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$ where each vector is in $\mathbb{R}^n$, and p desired output vectors, $\mathbf{t}_1, \ldots, \mathbf{t}_p$ in $\mathbb{R}^m$, then we form matrices.

  The data matrix can be either $p \times n$ (each vector is a row) or $n \times p$ (each vector is a column), and similarly, the output data can be in rows or columns- It doesn't matter which way you do it, but try to be consistent.

  Example from the class notes (bottom of p. 10): We have 4 vectors in $\mathbb{R}^2$ for input and 4 vectors in $\mathbb{R}^1$ for output. To complicate matters, for the batch training we need to form $\hat{W}$ and $\hat{X}$ rather than W and $\mathbf{b}$. In the Matlab script file, $\hat{X}$ is constructed as a 3×4 matrix and our outputs Y are 1×4. When Matlab solves the system:

  $$\hat{W}\hat{X} = Y$$

  then $\hat{W} = [W|b]$. That is, `W=Y/X` produces a 1×3 matrix $\hat{W}$, and our model is:

  $$W(1)x_1 + W(2)x_2 + W(3) = y$$

- For online training (in the homework), we want to use `wid_hoff1.m`. The help file says that X must be input as "number of points by dimension", which means the data is in rows. Similarly, the desired output should also have size "number of points by dimension", or 4×1.

  Once this W and b are created, W should be 1×2 and b should be a scalar.

- For the problem on p. 10, since $y = 0$ is exactly between the two other outputs of $-1, 1$, we plot the line $W\mathbf{x} + \mathbf{b} = 0$, which is what you are looking at in the graph.

Homework from p. 13

Some details are missing from this exercise: The inputs are 8 vectors in $\mathbb{R}^2$. Each pair belongs to a different class, so there are 4 classes. The outputs are also vectors in $\mathbb{R}^2$ (as in the notes).

The inputs to `wid_hoff1.m` are data arrays that should be 8×2 (rather than 2×8 as in the notes).

Your goal in this problem is to look at how changing the parameters `lr, iters` changes the output- W in this case is 2×2 and $\mathbf{b}$ is 2×1. The model equation is:

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Therefore, we have two lines representing boundaries:

$$\begin{aligned} w_{11}x_1 + w_{12}x_2 + b_1 &= 0 \\ w_{21}x_2 + w_{22}x_2 + b_2 &= 0 \end{aligned}$$

One way to see if you have a "good" W and $\mathbf{b}$ would be to plot these lines together with the input array (see the code on pg. 12 to help).