Classification problem: given vectors \( \mathbf{x}_1, \ldots, \mathbf{x}_n \subseteq \mathbb{R}^d \)
and labels \( \mathbf{y}_1, \ldots, \mathbf{y}_n \subseteq \mathbb{R} \) or \( \mathbf{y}_i \in \{ -1, +1 \} \).

The \( \mathbf{x}_i \) are features and the \( \mathbf{y}_i \) are labels.

A classical example: Fisher's iris dataset. (1936)
- Collected samples of 150 flowers (iris set, specifically).
- For each, he measured sepal length, sepal width, petal length, and petal width.
- Each iris was from one of the three species: \{setosa, versicolor, virginica\}.
  (50 flowers from each).
- Goal: use measurements to predict the species of a particular iris.
- Data set in Matlab (fisheriris) also on wikipedia.

For \( i = 1, \ldots, 150 \) (each iris)

\[
\mathbf{x}_i \in \mathbb{R}^4 \quad \mathbf{x}_i = \begin{bmatrix}
\text{sepal length} \\
\text{sepal width} \\
\text{petal length} \\
\text{petal width}
\end{bmatrix}
\]

\( \mathbf{y}_i \in \{ -1, 0, 1 \} \) (label for each species)
there are many ways to perform classification. Today, we will see just one of them (more to come later!)

**Linear predictor**: assume each label is a linear combination of the features, i.e. we can find weights \( \{w_j\} \):

\[
Y_i \approx w_1 x_{i1} + w_2 x_{i2} + \ldots + w_d x_{id} + w_0 \quad \text{for all } i.
\]

\[
\begin{bmatrix}
Y_1 \\
Y_2 \\
\vdots \\
Y_n
\end{bmatrix} \approx \begin{bmatrix}
x_{11} & x_{12} & \cdots & x_{1d} & 1 \\
x_{21} & x_{22} & \cdots & x_{2d} & 1 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
x_{n1} & x_{n2} & \cdots & x_{nd} & 1
\end{bmatrix} \begin{bmatrix}
w_1 \\
w_2 \\
\vdots \\
w_d \\
w_0
\end{bmatrix}
\]

\[\Rightarrow Y \approx Xw \quad \text{where } X = \begin{bmatrix}
x_{11} & x_{12} & \cdots & 1 \\
x_{21} & x_{22} & \cdots & 1 \\
\vdots & \vdots & \ddots & \vdots \\
x_{n1} & x_{n2} & \cdots & 1
\end{bmatrix}
\]

in LS classification, we use least squares to find \( w \).

i.e. solve: minimize \( \| y - Xw \|^2 \)

solution (assuming \( X \) has full rank, i.e. \( \text{rank}(X) = n \)) is

\[
\hat{w} = (X^TX)^{-1}X^Ty
\]

predicted labels are: \( \hat{y} = X\hat{w} = X(X^TX)^{-1}X^Ty \)
typically for classification problems, \( n \gg d \).

(many more examples than features). This isn't always the case... we'll see cases where \( n < d \) later!

So in general, we will have \( \hat{y} \neq y \). For the Fisher case, we can plot \( \hat{y}_i \) as a function of \( i \).

![Graph showing estimated label \( \hat{y}_i \) and true label \( y_i \) as a function of examples.]

our estimated labels are

\[
\hat{y}_i = x_i^T \hat{w}
\]

\( \uparrow \) \( \downarrow \)

\( \mathbb{R} \) \( \mathbb{R}^d \) or \( \mathbb{R}^{d+1} \) if we have offset.

Given a new example \( \tilde{x} \) that is unlabeled, we can predict the label via:

\[
\tilde{y} = \tilde{x}^T \hat{w}
\]

\( \uparrow \) weights we already solved for.

but we need \( \pm 1 \) as prediction! So let:

\[
\tilde{y} = \begin{cases} 
+1 & \text{if } \tilde{x}^T \hat{w} \text{ close to } 1 \\
-1 & \text{if } \tilde{x}^T \hat{w} \text{ close to } -1 
\end{cases} = \begin{cases} 
+1 & \text{if } \tilde{x}^T \hat{w} > 0 \\
-1 & \text{if } \tilde{x}^T \hat{w} < 0 
\end{cases} = \text{sign(} \tilde{x}^T \hat{w} \text{)}.
\]
The decision boundary is the set of examples $x$ such that $x^T \hat{w} = 0$. This is a subspace!

Note: if there is no offset, this is a subspace. But if we use an offset, we must require the last component of $x$ be 1, i.e., we can write it as the set of all $x$ such that $x^T \hat{w} + \hat{w}_0 = 0$. This is an affine space.

The case with two features (sepal length and width).

$+$: setosa
$\circ$: versicolor.

Equation of decision boundary for plotting purposes:

$$x_1 \hat{w}_1 + x_2 \hat{w}_2 + \hat{w}_3 = 0$$

$$\Rightarrow x_2 = \left(-\frac{\hat{w}_3}{\hat{w}_2}\right) + \left(-\frac{\hat{w}_1}{\hat{w}_2}\right)x_1$$

classify as versicolor.

classify as setosa.
Cross-validation

The point of model is to help predict new outcomes when we are using previously unseen data.

Perfectly predicting the data used to train your model isn’t impressive at all!

Moral of the story: you shouldn’t use residuals, i.e., $\| y - \hat{y} \|$ to evaluate a model’s usefulness.

Example:

---

Fit a polynomial to this! Using higher degree produces better fits, obviously.

always a perfect fit with degree n-1.

but the fit looks like this:

which isn’t what we had in mind.
Solution: use cross validation:

1) split data points randomly into two groups:

   data set

   training testing

typically, training set is larger than testing set.

2) use training data to learn a classifier, \( \hat{w} \).

3) compute residual on testing data to evaluate.

   i.e.,

   \[
   X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} = \begin{bmatrix} X_{\text{train}} \\ X_{\text{test}} \end{bmatrix}, \quad \hat{w} = (X_{\text{train}}^T X_{\text{train}})^{-1} X_{\text{train}}^T Y_{\text{train}}. \\
   y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} Y_{\text{train}} \\ Y_{\text{test}} \end{bmatrix}, \quad \text{err} = \| Y_{\text{test}} - X_{\text{test}} \hat{w} \|.
   \]

result:

model too simple

\[ \text{sweet spot} \]

model too complex

---

Testing error vs degree
Alternative method (k-fold cross validation).

1) split data into k groups.
2) use 1st group as testing, other k-1 groups for training.
3) repeat k times (each group gets to be the testing group once).
4) average the error from all trials.

Compared to regular cross-validation, this method is:

1) more expensive (takes more time to calculate) if done naively, but there exist efficient ways to do it.
2) more robust to how you choose to subdivide your data. i.e. more repeatable if you run the experiment again with different groups.

Extreme ("leave-one-out cross-validation")

- Run (N-1) fold cross validation!
- can be computed efficiently