Regression as Machine Learning

Given

- A data universe \( X \).
- A sample set \( S \) where \( S \subset X \).
- Some target function \( f : X \rightarrow \mathbb{R} \).
- A training set \( D \), where \( D = \{(x, y) \mid x \in S \text{ and } y = f(x)\} \).

Compute a model \( \hat{f} : X \rightarrow \mathbb{R} \) using \( D \) such that,

\[
\hat{f}(x) \approx f(x),
\]

for all \( x \in X \).

**Observation:** Same as machine learning in classification except for the co-domains of the target function and the model.

**Question:** How do we compute the model?
Regression ANNs

The perceptron revisited
Regression ANNs

Recall

\[ E_k(w) = \frac{1}{2}(y_k - \hat{y}_k)^2 \]
\[ = \frac{1}{2}(y_k - t(w \cdot x_k))^2 \]
\[ = \frac{1}{2}(y_k - t(a_k))^2 \]

Learning defined in terms of numerical error instead of classification error - regression by default!

We turn the regression problem into a classification problem by applying thresholding.
Regression ANNs

We can now look at the gradient,

\[ \nabla E_k(w) = \frac{d}{dw} E_k(w) \]

\[ = \frac{1}{2} \frac{d}{dw} (y_k - t(a_k))^2 \]

\[ = -(y_k - t(a_k)) \frac{dt}{dw} (a_k) \]

\[ = -(y_k - \hat{y}_k) \frac{dt}{da_k} (a_k) \frac{da_k}{dw} \text{ (chain rule)} \]

\[ = -(y_k - \hat{y}_k) t'(a_k) \frac{d}{dw} (w \cdot \overline{x}_k) \]

\[ = -(y_k - \hat{y}_k) t'(a_k) \overline{x}_k \]

\[ = \delta_k \overline{x}_k \]

where \( \delta_k = -(y_k - \hat{y}_k) t'(a_k) \) is called the error.
Regression ANNs

Now recall our update rule,

\[ \overline{w} \leftarrow \overline{w} + \Delta \overline{w} \]

From before we have

\[ \overline{w} \leftarrow \overline{w} + \eta \nabla E_k(\overline{w}) \]

From our discussion above it follows that

\[ \overline{w} \leftarrow \overline{w} + \eta \delta_k \overline{x}_k \]

**Observation:** The weights are updated using a scaled version of the input vector. It is also easy to see that the weights are scaled proportional to the error.

This is called *back propagation.*

If we can take the derivative of the transfer function then we can easily extend this to multi-layer neural networks.