Formalize Weight Decay and Dropout

Just like any machine learning model, neural networks are prone to overfitting. There are two ways to regularize neural nets: weight decay and dropout.

> ☆ **Key Points**
>
> The two ways to regularize neural networks are weight decay and dropout.
>
> Weight decay requires changing the loss function.
>
> Dropout is specific to neural networks.

Weight Decay

This is essentially L2 regularization. Recall that to train neural networks, we try to minimize $\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{x}_i, y_i)$ where ℓ is the cross entropy loss for classification and MSE for regression. To use weight decay, we change the loss function to $\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{x}_i, y_i) + \lambda \sum_{j=1}^{l} ||W_l||_2^2$, where W_l are the weights of the neural networks at each layer and $||W||_2^2 = \sum_{i,j} w_{ij}^2$. As usual, λ is a hyperparameter that has to be tuned.

Dropout

This is specific to neural networks. Essentially, the idea is that during training, with some probability p, we switch off each neuron (or set the corresponding weights to zero).

You can imagine that we change the transition function to $\sigma(z)q$, where q is a Bernoulli random variable taking on 1 with probability p and 0 with probability $1 - p$. This random variable is drawn independently each time such a transition function is evaluated — however, **only during training the network**. This is done at all nodes of all layers and effectively forces the network to rely less on any given neuron, preventing it from simply memorizing the training set.