6.7900 Fall 2024: Lecture Notes 11

1 Recap

We view neural networks as linear transformations interleaved with nonlinear transformations. Example models:

- Linear model: $f(x;\theta) = w^T x + b$.
- Linear model with features: $f(x;\theta) = w^T \phi(x) + b$.
- Linear model with learnable linear model: $f(x;\theta) = w^T (W^{(1)} x + b^{(1)}) + b$.
- 1 hidden layer model: $f(x;\theta) = w^T \tanh(W^{(1)} x + b^{(1)}) + b$.
- Multi-layer perceptron (MLP): $f(x;\theta) = w^T \tanh(W^{(2)} \tanh(W^{(1)} x + b^{(1)}) + b^{(2)}) + b$.

Recall that large numbers of randomly initialized hidden units give meaningful feature expansions.

2 Learning Neural Networks

We can use stochastic gradient descent (SGD) to minimize empirical risk (squared errors for regression, cross-entropy for classification).

$$\frac{1}{N}\sum_{i=1}^{N} L(y^i, f(x^i;\theta)) \quad + \lambda \left\|\theta\right\|_2^2$$

Our update rule for a randomly chosen data point is

$$\theta \leftarrow \theta - \eta \nabla_\theta L(y^i, f(x^i;\theta)).$$

We use differentiable transformations in neural nets to ensure gradients are computable. The problem is the empirical risk per example is not convex. Initialization of parameters is also important.

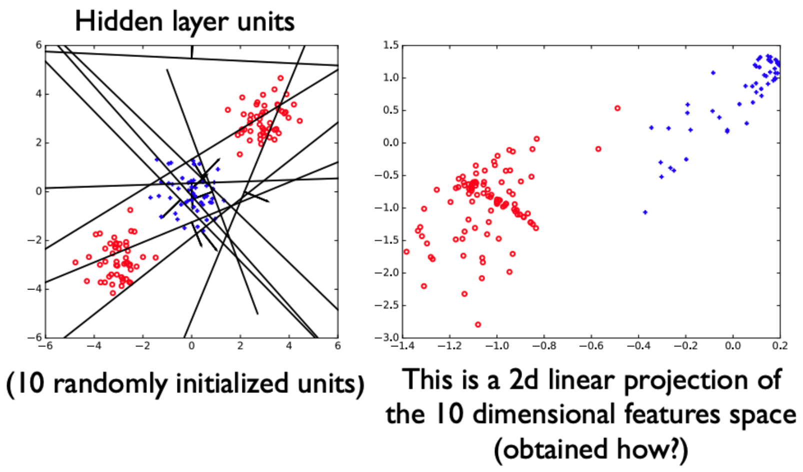


Figure 1: We classify red and blue points using randomly initialized hidden units. The separability is obtained by taking the classifier direction in the 10 dimensional space as one axis (the final linear layer), and any orthogonal direction as the other axis.

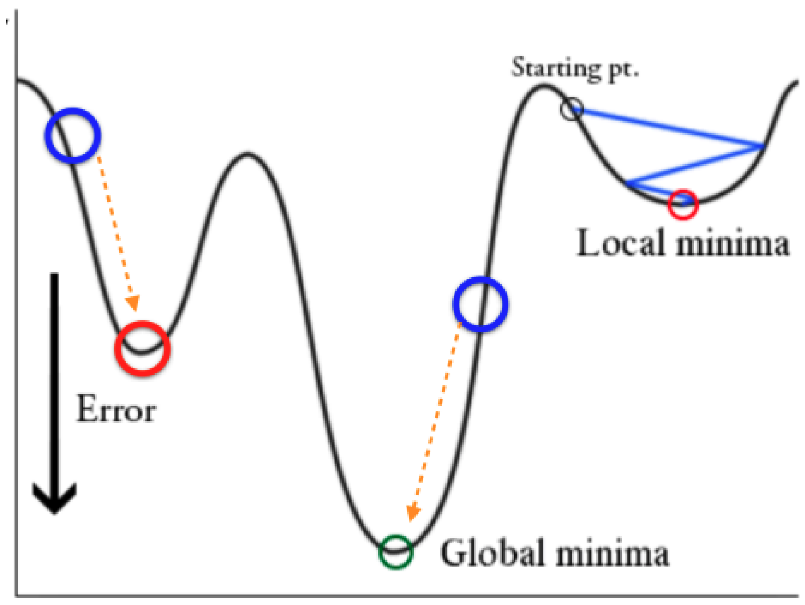


Figure 2: The loss landscape may not be nice.

Exercise: What is the problem with zero initialization?

Consider for an MLP with linear layers and ReLU activations if we use zero initialization. The output of the hidden layer is 0 for all hidden units. The output of the model is 0 for all data points. The gradient is 0 for all data points. The weights do not change.

3 Initialization, Learning Rate, and Loss Landscapes

To initialize, we can control fan-in variance. For a $d \times m$ matrix, we initialize $W_{ij} = \mathcal{N}(0, \sigma^2 I)$, where $\sigma^2 = \frac{1}{d}$. The output of neuron j is $\sum_{i=1}^{d} W_{ij} x_i = 1$, which has variance 1 under the assumption the input x_i has variance 1.

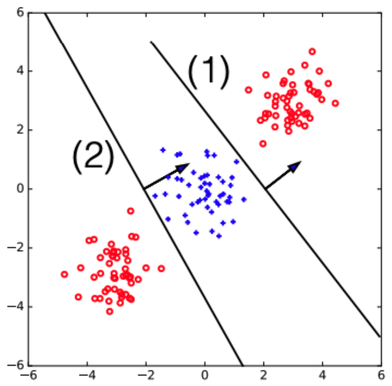
There are many learning rate schedules, which are often adaptive (e.g., Adam optimizer). Optimization parameters are often set as hyperparameters, which are selected based on validation performance.

Even with these choices, optimization is not easy. A simple MLP has many equivalent solutions in terms of weight matrices. For example, we can permute the nodes in each hidden layer. The matrices change, but the overall model would not.

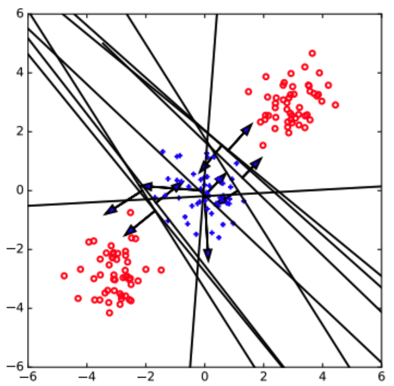
Furthermore, the high dimensional loss landscape is not well modeled by low dimensional figures. This means local minima obtained with different initialization might be connected with low loss paths.

3.1 Expressivity of Neural Networks

Say you have two hidden units, and that you linearly separate the data after training. You may also linearly separate with 10 hidden units, but the results may not be as interpretable.

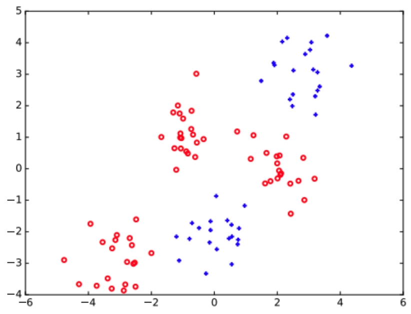


(a) 2 hidden units result.

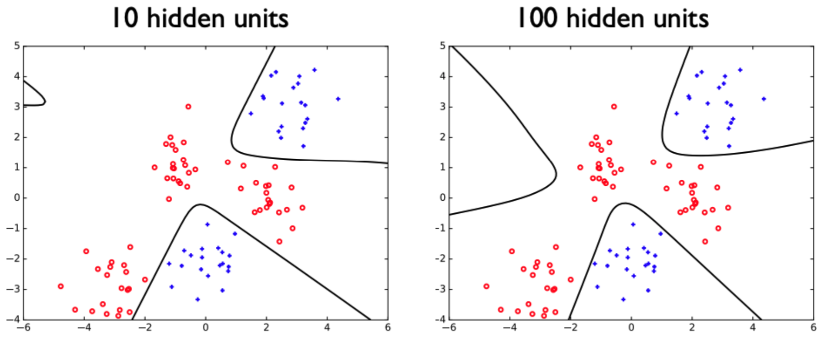


(b) 10 hidden units result.

Now consider a harder task. 2 hidden units can no longer solve the below task. Below, we first try to use 10 or 100 units, but with bad initialization. We show the



(a) A harder task not solvable with 2 hidden units.



(b) 10 and 100 hidden units with bad initialization.

effects of random offset initialization with 100 units.

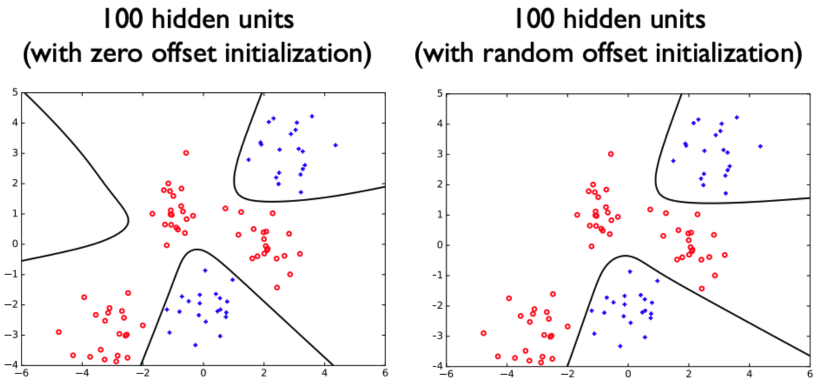


Figure 5: Random offset initialization with 100 units.

3.2 Size, Optimization

There are cases when less hidden units is sufficient, but it is hard to find a good solution. Using more hidden units means more overcapacity, making optimization easier.

4 Computation Graph, Backpropagation

We study how to evaluate the gradient with respect to all parameters for complicated models.

$$\begin{aligned} \theta &\leftarrow \theta - \eta \nabla_\theta L(y^i, f(x^i;\theta)) \\ f(x;\theta) &= w^T \tanh(W^{(2)} \tanh(W^{(1)}x + b^{(1)}) + b^{(2)}) + b \\ \theta &= \{w, b, W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}\}. \end{aligned}$$

We explain in the context of recurrent neural networks (RNNs) and its associated computation graph. We have a vocabulary size of V and state size of m (summary dimension).

Consider a state space model used to generate natural language sentences (one word at a time).

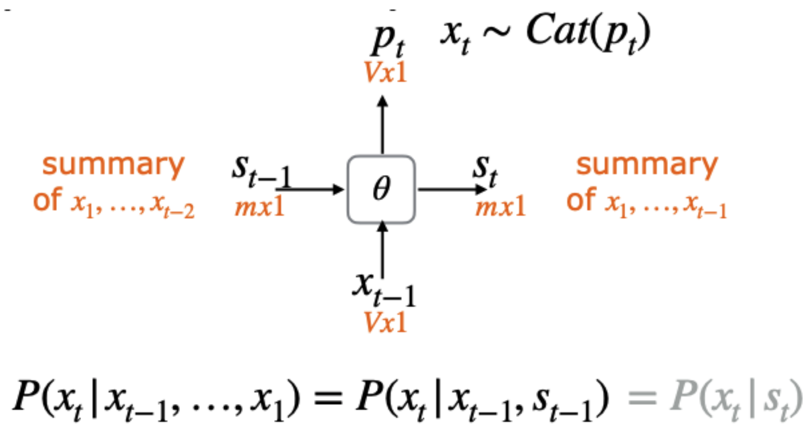

p_t $x_t \sim Cat(p_t)$
$Vx1$
summary of $x_1, \ldots, x_{t-2}$
s_{t-1}
$mx1$
θ
s_t
$mx1$
summary of $x_1, \ldots, x_{t-1}$
x_{t-1}
$Vx1$
$P(x_t | x_{t-1}, \ldots, x_1) = P(x_t | x_{t-1}, s_{t-1}) = P(x_t | s_t)$

Figure 6: RNN architecture.

Here s_t is a state that stores past information.

When learning these models, we do teacher forcing. We have an observed sequence $\hat{x}_1, \ldots$ and introduce losses at the outputs. We have

$$L(\hat{x}_t, p_t) = -\log p_t(\hat{x}_t) = -\log P(\hat{x}_t | \hat{x}_{t-1} \ldots, \hat{x}_1; \theta).$$

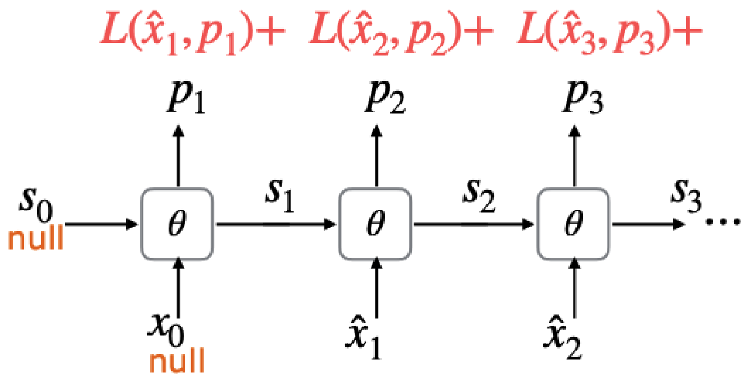


Figure 7: Teacher forcing.

We train this model so it has a high probability of producing sentences in training data.

We define the RNN weights as (offsets omitted for clarity)

$$s_t = \tanh(W^s s_{t-1} + W^x x_{t-1}), \theta = \{W^s, W^x, W^o\}$$
$$p_t = \text{softmax}(W^o s_t).$$

To compute gradients, we decompose into simple transformations, either with linear (with parameters) or non-linear (no parameters). We adjust the linear transformations in response to the desired output $\hat{x}_t$ by computing gradients.

We first compute gradients of linear transformations. Consider $z = Wx$. We have

$$\frac{\partial L}{\partial x} = \frac{\partial z}{\partial x}\frac{\partial z}{\partial z} = J^T \frac{\partial L}{\partial z} = W^T \frac{\partial L}{\partial z}.$$

Here J is the Jacobian. Any non-linear transformation acts the same with a different Jacobian. Using the chain rule, we can update all linear transformations as follows:

$$L(\hat{x}_t, p_t) = -\log p_t(\hat{x}_t), p_t(y) = \frac{\exp(z_y)}{\sum_j \exp(z_j)}$$
$$\frac{\partial L}{\partial z} = -(I(\hat{x}_t) - p_t)$$
$$\frac{\partial L}{\partial s_t} = (W^o)^T \frac{\partial L}{\partial z}$$
$$\frac{\partial L}{\partial a_t} = \text{diag}(1 - \tanh^2(a_t)) \frac{\partial L}{\partial s_t}.$$

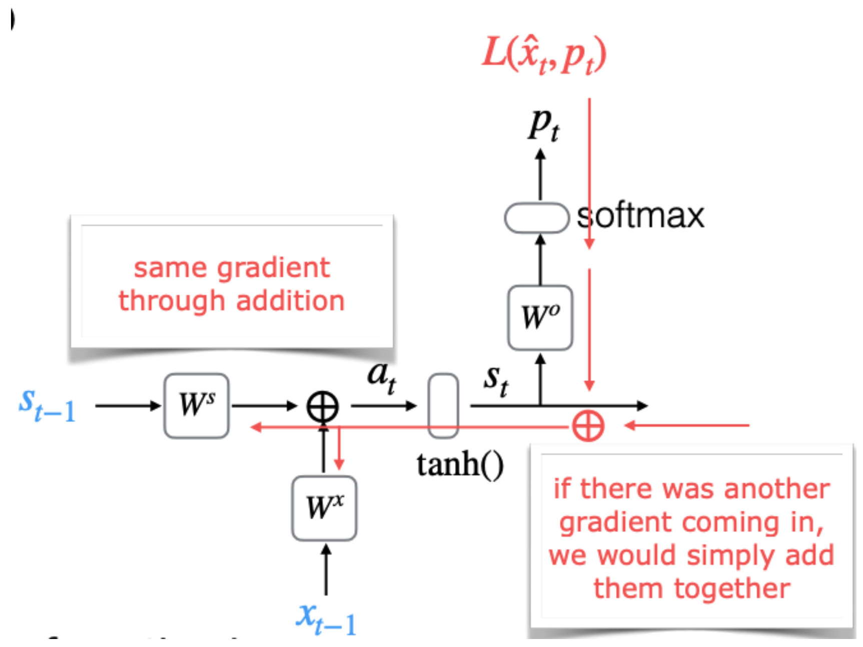

$L(\hat{x}_t, p_t)$
p_t
softmax
W^o
same gradient through addition
s_{t-1}
W^s
a_t
s_t
tanh()
W^x
x_{t-1}
if there was another gradient coming in, we would simply add them together

Figure 8: Backpropagation through RNN.