CS281: Advanced ML **September 27, 2017**

Lecture 8: Backpropagation & Directed Graphical Models

Lecturer: Sasha Rush *Scribes: Giridhar Anand, Michael Xueyuan Han, Ana-Roxana Pop*

8.1 Backpropagation in Neural Networks

8.1.1 Neural networks review

In the last lecture, we defined the mean parameter of a neural network as follows:

$$\mu = \sigma(w^T \text{ReLU}(Wx))$$

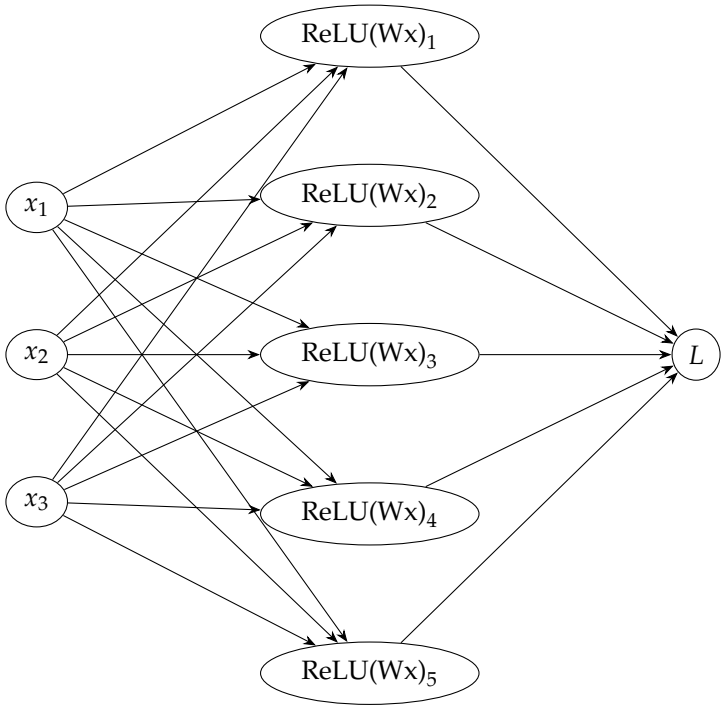
Here, μ parameterizes a Bernoulli distribution, $\text{Ber}(\mu)$. Suppose we want to find μ such that it maximizes the likelihood of a single data example (x, y). Then we compute

$$\mu = \underset{\mu}{\text{argmax}} \log p(y|x) = \arg\min_{\mu} (-\log p(y \mid x)) = \arg\min_{\mu} L$$

where L is the loss of the neural network.

8.1.2 Chain rule and backpropagation

We can represent the neural network graphically as follows:



In order to generate this graph, we must perform the following computational operations in order:

$$\begin{array}{ccccccccccc} v^{(0)} & \rightarrow & v^{(1)} & \rightarrow & v^{(2)} & \rightarrow & v^{(3)} & \rightarrow & v^{(4)} & \rightarrow & L \\ x & & Wv^{(0)} & & \text{ReLU}(v^{(1)}) & & w^T v^{(2)} & & \sigma(v^{(3)}) & & -\log v^{(4)} \end{array}$$

We would like to get the gradient terms $\dot{v}^{(i)} \equiv \frac{dL}{dv^{(i)}}$ for any i, which tell us how each part of the neural network affects our loss. We can do this by applying the chain rule (of calculus) to get a recursive solution (by convention, the derivative of a scalar with respect to a vector is represented as a column vector):

$$\frac{dL}{dv^{(i)}} = \left(\frac{dL}{dv^{(i+1)}}\right)^T \frac{dv^{(i+1)}}{dv^{(i)}}$$

$$\frac{\partial L}{\partial v_k^{(i)}} = \sum_j \frac{\partial L}{\partial v_j^{(i+1)}} \frac{\partial v_j^{(i+1)}}{\partial v^{(i)}}$$

Since the gradient of each term depends on the gradient of the subsequent term, we can compute the gradients in reverse while applying the chain rule. This method is known as backpropagation. For each backward step, we need to remember everything that was computed in the corresponding forward step, namely $v^{(i)}$, $\dot{v}^{(i+1)}$, and $\frac{dv^{(i+1)}}{dv^{(i)}}$:

$v^{(0)}$	$\rightarrow$	$v^{(1)}$	$\rightarrow$	$v^{(2)}$	$\rightarrow$	$v^{(3)}$	$\rightarrow$	$v^{(4)}$	$\rightarrow$	L
x		$Wv^{(0)}$		$\text{ReLU}(v^{(1)})$		$w^T v^{(2)}$		$\sigma(v^{(3)})$		$-\log v^{(4)}$
$\dot{v}^{(0)}$	$\leftarrow$	$\dot{v}^{(1)}$	$\leftarrow$	$\dot{v}^{(2)}$	$\leftarrow$	$\dot{v}^{(3)}$	$\leftarrow$	$\dot{v}^{(4)}$	$\leftarrow$	
...		...		$(\dot{v}^{(2)})^T W$		$\dot{\sigma}(v^{(3)})\dot{v}^{(4)}$		$-\frac{1}{v^{(4)}}$		

8.1.3 Exercise: Deriving parameter updates for NN with 1 hidden layer

Consider the neural network with one hidden layer shown in 8.1. The first layer applies the transformation $\boldsymbol{z}_n = g(\boldsymbol{V}\boldsymbol{x}_n)$, and the second layer applies the transformation $\hat{\boldsymbol{y}}_n = h(\boldsymbol{W}\boldsymbol{z}_n)$. Assume h is the canonical link function for the output GLM. Derive the gradient updates for the parameters $\boldsymbol{V}$ and $\boldsymbol{W}$.

$$\mathbf{x}_n \xrightarrow{\mathbf{V}} \mathbf{a}_n \xrightarrow{g} \mathbf{z}_n \xrightarrow{\mathbf{W}} \mathbf{b}_n \xrightarrow{h} \hat{\mathbf{y}}_n$$

Figure 8.1: Two-layer neural net. Image taken from Murphy.

Solution. Let the NLL for data n be denoted by J_n.

1. For the output layer, we have $\boldsymbol{W} = \{\boldsymbol{w}_k\}_{i=1}^K$. Taking the gradient of the NLL w.r.t. each $\boldsymbol{w}_k$ yields

$$\frac{\partial J_n}{\partial \boldsymbol{w}_k} = \frac{\partial J_n}{\partial b_{nk}} \frac{\partial b_{nk}}{\partial \boldsymbol{w}_k} = (\hat{\boldsymbol{y}}_n - \boldsymbol{y}_n)\boldsymbol{z}_n,$$

   where we derive $\frac{\partial J_n}{\partial b_{nk}} = \hat{\boldsymbol{y}}_n - \boldsymbol{y}_n$ by using the results for GLMs. We call this quantity the "error signal" for the output layer: $\hat{\boldsymbol{y}}_n - \boldsymbol{y}_n = \delta_{nk}^w$.

2. For the hidden layer, we have $\boldsymbol{V} = \{\boldsymbol{v}_j\}_{j=1}^H$. Taking the gradient of the NLL w.r.t. each $\boldsymbol{v}_j$ yields

$$\frac{\partial J_n}{\partial \boldsymbol{v}_j} = \frac{\partial J_n}{\partial a_{nj}} \frac{\partial a_{nj}}{\partial \boldsymbol{v}_j} = \delta_{nj}^v \boldsymbol{x}_n,$$

   where δ_{nj}^v is the error signal for the hidden layer. We can derive the error signal by applying the chain rule again:

$$\delta_{nj}^v = \frac{\partial J_n}{\partial a_{nj}} = \sum_{k=1}^K \frac{\partial J_n}{\partial b_{nk}} \frac{\partial b_{nk}}{\partial a_{nj}} = \sum_{k=1}^K \delta_{nk}^w \frac{\partial b_{nk}}{\partial a_{nj}}$$

Note that by definition,

$$b_{nk} = \langle \boldsymbol{w}_k, g(\boldsymbol{a}_n) \rangle = \sum_j w_{kj} g(a_{nj}) \rightarrow \frac{\partial b_{nk}}{\partial a_{nj}} = w_{kj} g'(a_{nj})$$

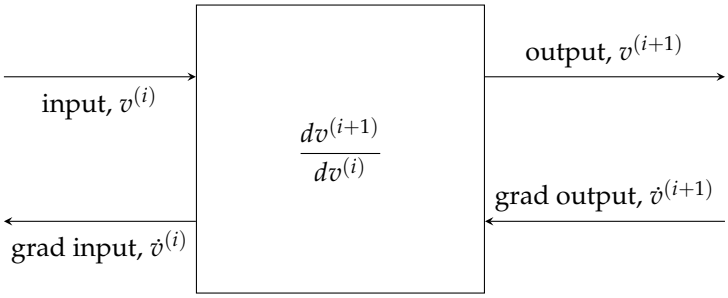
Combining, the error signal for the hidden layer is

$$\delta_{nj}^v = \sum_{k=1}^{K} \delta_{nk}^w w_{kj} g'(a_{nj})$$

8.1.4 Writing software for neural networks

- "blocks" style neural network (e.g. Torch)

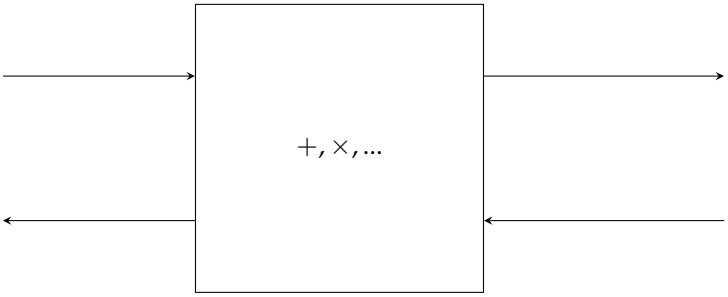
  Computation is done in "blocks" which are black boxes f that implement the following contract:



  We can also augment the black boxes. For instance, if we let f take in parameter W, we can also compute $\frac{dL}{dW}$ within this function.

- computational graph (e.g. Theano, TensorFlow)

  Everything is implemented in terms of primitives, so there are no black boxes:



  This allows us to optimize the neural network once and run it on many examples.

- imperative/autograd systems

  These are tape-based systems in which the computational graph can look different for different examples, but we can still compute gradients using backpropagation. Torch is built on an autograd core, but higher level functions like the Linear module take on a "blocks" style approach.

8.2 Graphical Models

8.2.1 Directed Graphical Models

The goal of using directed graphical models (DGMs) is to separate out two parts of a model:

1. Conditional Independence
2. Parameters and Parametrization

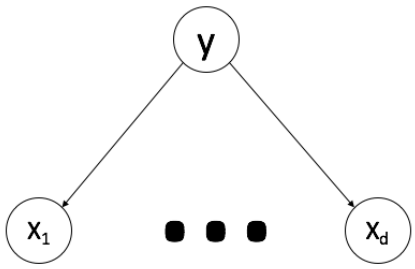


Figure 8.2: The graphical model of Naive Bayes

In the case of Naive Bayes (see Figure 8.2), we know from a previous lecture that:

- $p(y)$ is probably categorical.
- $p(x_j|y)$ could be one of many different distributions, including Categorical, Gaussian, Bernoulli, etc.

We are interested in the following distributions from the underlying data that we have:

- p(y, x): joint distribution
- $p(x_j)$: marginal distribution, or $p(y \mid x)$: conditional distribution

The structure of the model will often determine the difficulty of inference. This is the motivation of why we want to draw these graphs.
On a high level, given $p(A, B, C)$, we can always apply the chain rule (in probability):

$$p(A, B, C) = p(A \mid B, C)\, p(B \mid C)\, p(C)$$

However, if we write $p(A, B, C)$ in the way above, we basically assume that all variables depend on each other. In some cases, this is not necessarily true, and we want to find a factorization as below.

8.2.2 Factorization

If we have the case presented in Figure 8.3, we can rewrite $p(A \mid B, C) \rightarrow p(A \mid B)$. Having A only depend on one variable (B) is better than having it depend on two variables (B and C).

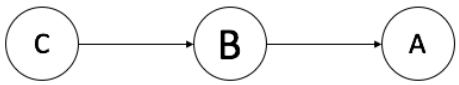


Figure 8.3: A graph where factorization is possible.

8.2.3 Formalism of DGMs

Formally, for directed graphical models (DGMs) (or *Bayes Nets*, or *causal graphs*) we have:

- A graph $G = (V, E)$ where $(s, t) \in E, s \neq t$ (V are vertices, E are edges)
- Each node is a random variable.
- Each edge is a conditioning decision.
- The graph is topologically ordered and it is a directed acyclic graph (DAG).
- Notation: $\text{pa}(x)$ represents x's parents.

8.2.4 Parents notation

Here, A and B are independent of each other, and they are C's parents (as in Figure 8.4). We can then write:

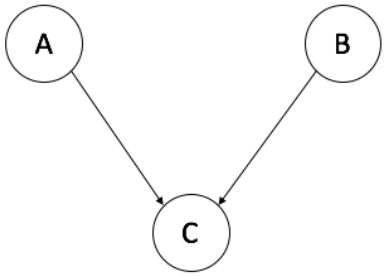


Figure 8.4: A graph to illustrate the use of parents notation.

$$p(A, B, C) = p(A)\, p(B)\, p(C \mid A, B) = p(A)\, p(B)\, p(C \mid \text{pa}(C))$$

8.2.5 Plate Notation

When we have lots of exchangeable variables (i.e., order is not important), we can use the *plate* notation. We want to graphically represent Naive Bayes on examples $(x_j^{(n)}, y^{(n)})$, in which

- y is parameterized on π: $p(y^{(n)} \mid \pi)$, and
- x_j depends on y and μ: $p(x_j^{(n)} \mid y, \mu)$

Since we have n samples of (x, y), we can use the plate notation, as shown in Figure 8.5.

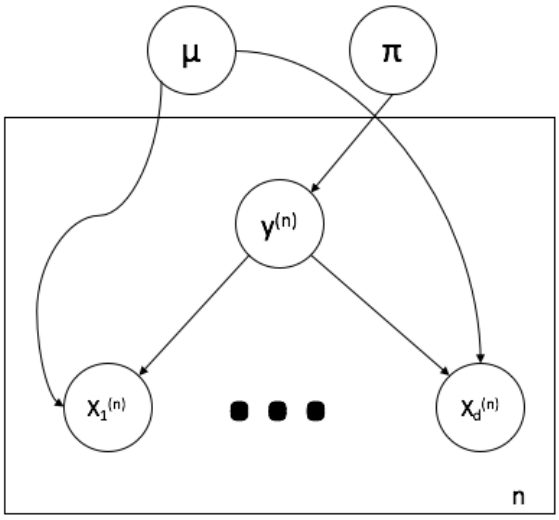


Figure 8.5: A graph to illustrate the use of plate notation.

8.2.6 Caching probabilities

You can save "probabilities" in the model, which is simply *caching* the values of probabilities with the graph. In this case (Figure 8.6), variables are discrete. We call C's probability table *conditional probability table (CPT)* since it is conditioned on A, B. Note that the values do not tell us anything about how the distribution is parameterized. Those are simply the probability values that you can read off from the graph.
Note also that the CPT of $p(x_i \mid x_1, \cdots, x_{i-1}) = O(\prod_i |x_i|)$ (i.e., exponential growth with the number of conditional terms).

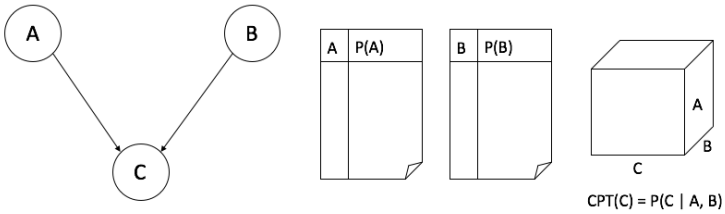


Figure 8.6: A model with probability tables. The CPT of C is a three-dimensional table.

8.2.7 Examples of Directed Graphical Models

Example 1 (Markov Chain)**.** Figure 8.7 shows an example of a Markov chain graphical model.

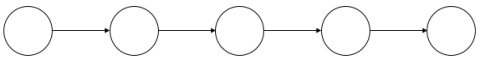


Figure 8.7: Markov Chain Graphical Model

Example 2 (Second Order Markov Chain)**.** Figure 8.8 shows an example of a second order Markov chain graphical model.

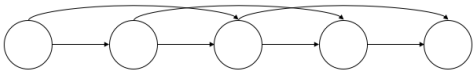


Figure 8.8: Second Order Markov Chain Graphical Model

Example 3 (Hidden Markov Model)**.** Figure 8.9 shows an example of a hidden Markov graphical model.

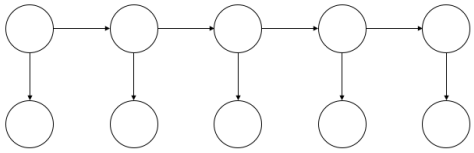


Figure 8.9: Hidden Markov Graphical Model

Example 4 (Navie Bayes)**.** Figure 8.10 shows an example of a Naive Bayes graphical model.

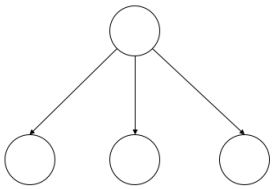


Figure 8.10: Naive Bayes Graphical Model

As we have seen in Figure 8.5, we can also incorporate parameters in the DGMs, as it is illustrated in the next example.

Example 5. In this case (Figure 8.11), we use the same Naive Bayes example with parameters. Here, we incorporate parameters $\alpha \sim$ Dirichlet. This is interesting because it combines two types of distributions: some of them are discrete, but in this example α and π are drawn from continuous distributions, as marked in the figure below.

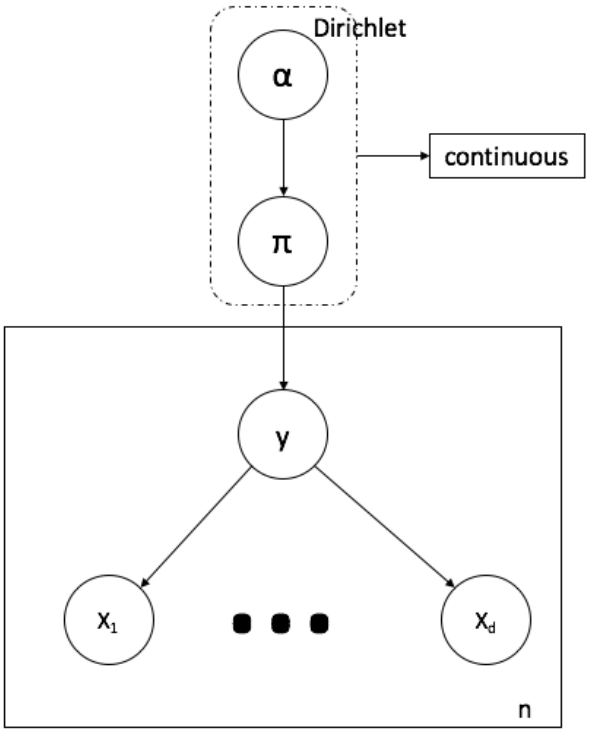


Figure 8.11: Naive Bayes Graphical Model with Parameters

Figure 8.11 corresponds to a single example. If we have multiple examples, we can use a plate-in-plate representation as in Figure 8.12.

8.3 Gaussian Directed Models

Gaussian directed models are a special case of DGMs where every one of the variables has the following distribution:

$$p(x_i \mid \text{pa}(x_i)) = \mathcal{N}(x_i \mid \mu_i + \sum_{j=\text{pa}(x_i)} W_{ij}(x_j - \mu_j),\ \ \sigma_i^2)$$

In the above equation, we transform each of the μ_i based on the starting mean plus a linear transformation of their parents (and to simplify things, we subtract the mean of each parent). We have an underlying generative process where each one of our random variables is a draw from a Gaussian and its children are a linear transformations of that draw.
This means that, we can rewrite x_i as:

$$x_i = \mu_i + \sum_j W_{ij}(x_j - \mu_j) + \sigma_i z_i \qquad \forall i \ \ z_i \sim \mathcal{N}(0,1)$$

Notice that $\sigma_i z_i$ is just Gaussian random noise.
If we define $S = \text{diag}(\sigma)$ (where each term will contribute a different corresponding σ_i), we can rewrite the above equation in a matrix form:

$$(x_i - \mu_i) = W(x - \mu) + Sz$$

where $\mu = [\mu_1, \cdots, \mu_d]$ is a vector containing each individual means.

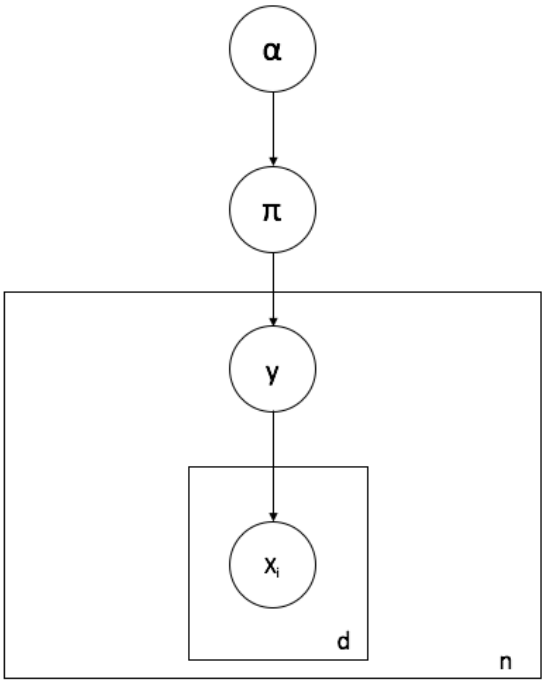


Figure 8.12: Naive Bayes Graphical Model with Parameters and Plate-in-Plate Notation

By rearranging the terms, we find:

$$\begin{aligned} Sz &= (I - W)(x - \mu) \\ x - \mu &= (I - W)^{-1} Sz \end{aligned}$$

This tells us how the x random variable differs from the mean at each of the different positions. We know that the Σ term for our covariant matrix is defined as:

$$\begin{aligned} \Sigma \equiv \operatorname{cov}[x - \mu] &= \operatorname{cov}\left[(I - W)^{-1} S\, z\right] \\ &= (I - W)^{-1} S \operatorname{cov}[z]\ S((I - W)^{-1})^T \\ &= (I - W)^{-1}\, S^2\, ((I - W)^{-1})^T \end{aligned}$$

which means that, in general, for Gaussian DGMs we have:

$$\text{Gaussian DGM} \sim \mathcal{N}(\mu,\ (I - W)^{-1}\, S^2\, ((I - W)^{-1})^T)$$

Remark. We will talk about *D-Separation* in the next lecture.