

# Week 6: Decision Trees and Variational Autoencoders

<http://mlvu.github.io>

March 13, 2019

## 1 Decision trees

Imagine you do a newspaper round to help you get through these lean times. On your round, you encounter a number of dogs that either bark or (try to) bite. The dogs are described by the following binary features: *Heavy*, *Smelly*, *Big* and *Growling*. Consider the following set of examples:

| Heavy | Smelly | Big | Growling | Bites |
|-------|--------|-----|----------|-------|
| No    | No     | No  | No       | No    |
| No    | No     | Yes | No       | No    |
| Yes   | Yes    | No  | Yes      | No    |
| Yes   | No     | No  | Yes      | Yes   |
| No    | Yes    | Yes | No       | Yes   |
| No    | No     | Yes | Yes      | Yes   |
| No    | No     | No  | Yes      | Yes   |
| Yes   | Yes    | No  | No       | Yes   |

**question 1:** What is the entropy of the target value *Bites* in the data?

**question 2:** Which attribute would the ID3 algorithm choose to use for the root of the tree (without pruning)?

**question 3:** What is the information gain of the attribute you chose in the previous question?

**question 4:** Draw the full decision tree that would be learned for this data using ID3 without pruning.

**question 5:** Suppose three new dogs appear in your round as listed in the table below. Classify them using the decision tree from the previous question.

| Dog    | Heavy | Smelly | Big | Growling | Bites |
|--------|-------|--------|-----|----------|-------|
| Buster | Yes   | Yes    | Yes | Yes      | ?     |
| Pluto  | No    | Yes    | No  | Yes      | ?     |
| Zeus   | Yes   | Yes    | No  | No       | ?     |

**question 6:** Someone proposes a new scheme to prevent overfitting: she suggests to set a pre-defined maximum depth for the decision trees. When the standard algorithm reaches this depth, it terminates. Could this help to prevent overfitting? Why (not)?

**question 7:** In the maximum depth scheme introduced above, how would you determine a good value for the maximum depth for a given data set?

**question 8:** Why can't we apply L1-regularization to this decision tree learning problem?

## 2 Variational autoencoders

The maximum likelihood principle tells us to optimize the quantity  $p(\mathbf{x} | \theta)$  as a function of  $\theta$  (the model parameters).

For complex models, this does not usually lead to a closed form solution.

Instead, we will rewrite the maximum likelihood objective using the following decomposition.

$$\ln p(\mathbf{x} | \theta) = L(\mathbf{q}, \theta) + \text{KL}(\mathbf{q}, p)$$

with

$\mathbf{q}(z | \mathbf{x})$  any distribution on  $z$

$\text{KL}(\mathbf{q}, p)$  the Kullback-Leibler divergence  
between  $\mathbf{q}(z | \mathbf{x})$  and  $p(z | \mathbf{x}, \theta)$

$$L(\mathbf{q}, \theta) = \mathbb{E}_{\mathbf{q}} \ln \frac{p(\mathbf{x}, z | \theta)}{\mathbf{q}(z | \mathbf{x})}$$

We will first prove that this equality holds. We start with the right hand side, fill in the components, and derive the left-hand side.

**question 9:** Fill in the blanks. We have written everything in terms of *expectations*  $\mathbb{E}$  to simplify the notation. The expectation is over the random variable  $z$ , while  $x$  has some definite value. Note that  $\mathbb{E}f(z) + \mathbb{E}g(z) = \mathbb{E}[f(z) + g(z)]$ .

$$\begin{aligned} L(\mathbf{q}, \theta) + \text{KL}(\mathbf{q}, \mathbf{p}) &= \mathbb{E}_{\mathbf{q}} \ln \frac{p(\mathbf{x}, z | \theta)}{q(z | \mathbf{x})} - \mathbb{E}_{\mathbf{q}} \ln \frac{p(z | \mathbf{x}, \theta)}{q(z | \mathbf{x})} \\ &= \dots \\ &= \mathbb{E}_{\mathbf{q}} \ln p(\mathbf{x}, z | \theta) - \mathbb{E}_{\mathbf{q}} \ln p(z | \mathbf{x}, \theta) \\ &= \mathbb{E}_{\mathbf{q}} \ln \frac{p(\mathbf{x}, z | \theta)}{p(z | \mathbf{x}, \theta)} = \dots \\ &= \mathbb{E}_{\mathbf{q}} \ln p(\mathbf{x} | \theta) = \ln p(\mathbf{x} | \theta) \end{aligned}$$

In the EM algorithm, we search by alternately optimizing  $L(\mathbf{q} | \theta)$  with respect to  $\theta$  and setting  $\mathbf{q}$  equal to  $\mathbf{p}$  (so that the KL term becomes zero).

**question 10:** For the variational autoencoder, we cannot (easily) perform this last step. Why not?

Instead, we *approximate*  $p(z | \mathbf{x}, \theta)$  with a neural network  $q_{\mathbf{v}}(z | \mathbf{x})$  that produces a distribution on  $z$  given some  $\mathbf{x}$ . We call the neural network computing  $p(\mathbf{x} | \mathbf{z}, \theta)$   $p_{\mathbf{w}}(\mathbf{x} | \mathbf{z})$ , to make the notation a little more friendly. Here,  $\mathbf{w}$ , stands for all parameters of the  $\mathbf{p}$  network, and  $\mathbf{v}$  stands for all parameters of the  $\mathbf{q}$  network.<sup>1</sup>

This gives us an auto-encoder-like structure. An input is mapped to a distribution  $q_{\mathbf{v}}(z | \mathbf{x})$  by the **encoder**. We sample a single  $z$  from this distribution and pass it through the **decoder**  $p_{\mathbf{w}}(\mathbf{x} | z)$  to produce a distribution on  $\mathbf{x}$  (see the **slides** for diagrams).

To find a way to train such an architecture, we turn again to our decomposition of the likelihood. In our new notation:

$$\ln p_{\mathbf{w}}(\mathbf{x}) = L(\mathbf{v}, \mathbf{w}) + \text{KL}(\mathbf{q}, \mathbf{p}) .$$

---

<sup>1</sup>We've turned  $\theta$  into  $\mathbf{w}$  and added parameters  $\mathbf{v}$  for our approximation  $\mathbf{q}$  on the conditional distribution on  $z$ . We've also taken the parameters out of the conditional, because we will always talk about the function "given the parameter"; we will never attempt to take the parameters out of the conditional.

The KL divergence term is difficult to compute: it's an expectation, and it contains the function  $p_w(z | x)$  which requires us to invert the decoder neural network (reason about the inputs given the outputs).

However, because the KL divergence is always positive, we know that

$$\ln p_w(x) \geq L(v, w)$$

for any  $q_v$  we choose. This is why  $L$  is called the variational *lower bound*.<sup>2</sup> If we choose our parameters  $w, v$  to maximize  $L$ , we are also, indirectly, maximizing  $\ln p_w(x)$ .<sup>3</sup>

To do so, we rewrite  $L(v, w)$  into two separate terms: a KL divergence and an expectation:

$$L(v, w) = -\text{KL}(q_v(z | x), p_v(z)) + \mathbb{E}_{q_v} \ln p_w(x | z)$$

**question 11:** Show that this equation holds (i.e. rewrite the left part into the right).

$$\begin{aligned} L(v, w) &= \mathbb{E}_{q_v} \ln \frac{p_w(x, z)}{q_v(z | x)} \\ &= \dots \\ &= \dots \\ &= \dots \\ &= \mathbb{E} \ln p_w(x | z) - [\mathbb{E} \ln q_v(z | x) - \mathbb{E} \ln p_w(z)] \\ &= \mathbb{E} \ln p_w(x | z) - \text{KL}(q_v(z | x), p_w(z)) \end{aligned}$$

Thus, to optimize our variational autoencoder, we should maximize  $L$ . In other words,  $-L$  is our loss function. The only problem left to solve is that the second term is an expectation (which we cannot compute explicitly).

**question 12:** How is this solved in practice?

---

<sup>2</sup>The word *variational* comes from the fact that one of its arguments,  $q$ , is a function (the calculus of functions is called *variational* calculus). For our purposes, this distinction doesn't matter much, since the function  $q$  is defined by a set of parameters  $v$ , so ultimately we will take the derivative over those parameters, as we are used to.

<sup>3</sup>How close the lower bound  $L$  comes to the true value  $\ln p_w(x)$  depends on how well our encoder network  $q_v$  approximates the true conditional distribution on  $z$ :  $p_w(z | x)$ . I.e. how small the KL term in the original decomposition is.