SL10. Bayesian Learning

Introduction:

- We're trying to learn the best (most probable) hypothesis *H* given input data and domain knowledge.

Best == Most probable

- It's the probability of some hypothesis *h* given input data *D*:

$$P_r(h \mid D)$$

- We're trying to find hypothesis *h* with the highest probability P_r:

$$argmax_{h \in H}(P_r(h \mid D))$$

Bayes Rule:

- Bayes Rule for probability states that:

$$P_r(h \mid D) = \frac{P_r(D \mid h)P_r(h)}{P_r(D)}$$

- $P_r(h \mid D)$ → The probability of a specific hypothesis given input data.
- $P_r(D \mid h)$ → The probability of data given the hypothesis. It's the likelihood of seeing some particular labels associated with input points, given a world where some hypothesis *h* is true.
- $P_r(h)$ → The prior belief of a particular hypothesis. This values encapsulates our prior belief that one hypothesis is likely on unlikely compared to other hypotheses. This is basically the domain knowledge.
- $P_r(D)$ → The prior belief of seeing a particular set of data (A normalizing term).

Bayesian Learning:

- Bayesian Learning algorithm:

```
For each $h \in H$:
    Calculate $P_r(h \mid D) = \frac{P_r(D \mid h)P_r(h)}{P_r(D)}$
Output:
    $h = argmax_{h \in H}(P_r(h \mid D))$
```

- Since we're interested in finding the hypothesis with the highest probability, not the exact probability value for each hypothesis, we can ignore the normalization factor $P_r(D)$.

  We'll end up calculating:

$$P_r(h \mid D) \cong P_r(D \mid h)P_r(h)$$

- Using this approximate probability, we can calculate the Maximum a Posteriori:

$$h_{map} = argmax_{h \in H}(P_r(h \mid D))$$

- We can also drop $P_r(h)$ from the equation, ending up with the Maximum Likelihood:

$$h_{ml} = argmax_{h \in H}(P_r(D \mid h))$$

- We're not exactly dropping $P_r(h)$, we just assume that our prior belief is "all hypothesis are equally likely". This is called a "Uniform Prior".

- The problem with Bayes Learning is that it's not practical to perform direct computations for large hypotheses spaces.

Bayesian Learning in Action:

- Assume:
  - Given Noise-free training data $\{\langle x_i, d_i \rangle\}$ as examples of c.
  - $c \in H$
  - Uniform prior.
- We need to calculate $P_r(h \mid D)$:

$$P_r(h \mid D) = \frac{P_r(D \mid h) P_r(h)}{P_r(D)}$$

$$P_r(h) = \frac{1}{|H|}$$

$$P_r(D \mid h) = \begin{cases} 1 \; if \; d_i = h(x_i) \; \forall x_i, d_i \in D \\ 0 \; otherwise \end{cases}$$

$$P_r(D) = \sum_{h_i \in H} P_r(D \mid h_i) \, P_r(h_i) = \sum_{h_i \in VS_{H,D}} 1 \, . \, \frac{1}{|H|} = \frac{|VS|}{|H|}$$

$$P_r(h \mid D) = \frac{1 \, . \, \frac{1}{|H|}}{\frac{|VS|}{|H|}} = \frac{1}{|VS|}$$

Bayesian Learning with Noise:

- Assume:
  - Given $\{\langle x_i, d_i \rangle\}$
  - $d_i = f(x_i) + \varepsilon$
  - $\varepsilon_i \sim N(0, \sigma^2)$ → IID (Independent and Identically Distributed)
- We need to calculate $P_r(h \mid D)$:

$$h_{ml} = argmax_{h \in H} (P_r(D \mid h))$$

$$h_{ml} = argmax_{h \in H} \prod_i P_r(d_i \mid h)$$

  - Given a Gaussian noise:

$$h_{ml} = argmax_{h \in H} \prod_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(\frac{-1}{2} \, . \, \frac{\left(d_i - h(x_i)\right)^2}{\sigma^2}\right)}$$

  - Since we're looking for the maximum:
    1. We can remove $\frac{1}{\sqrt{2\pi\sigma^2}}$ since we're looking for the maximum.
    2. We can take the *ln* to remove the exponential. Since the *ln* of a product is equal to the sum of the *ln*, we end up with the following function.

$$h_{ml} = argmax_{h \in H} \sum_i \frac{-1}{2} . \frac{\left(d_i - h(x_i)\right)^2}{\sigma^2}$$

- Again, since we're calculating the maximum, we can remove the $\frac{1}{2}$ and the σ^2:

$$h_{ml} = argmax_{h \in H} - \sum_i \left(d_i - h(x_i)\right)^2$$

- Maximizing a negative value is the same as minimizing the positive sum of this value:

$$h_{ml} = argmin_{h \in H} \sum_i \left(d_i - h(x_i)\right)^2$$

- This means: If you're looking for the maximum likelihood hypothesis, you should minimize the sum of squared error.

Minimum Description Length:

$$h_{map} = argmax_{h \in H}(P_r(D \mid h)P_r(h))$$
$$h_{map} = argmax_{h \in H}[\log P_r(D \mid h) + \log P_r(h)]$$
$$h_{map} = argmin_{h \in H}[- log\, P_r(D \mid h) - log\, P_r(h)]$$

- Information theory: The optimal code for some event w with probability p has a length of $-\log p$.
- This means that in order to maximize the Maximum a Posteriori hypothesis, we need to minimize two terms that can be described as length:
  - $log\, P_r(h)]$ → This is the length of the hypothesis, which is the number of bits needed to represent this hypothesis.
  - $log\, P_r(D \mid h)$ →This is the length of the data given a particular hypothesis. If the hypothesis perfectly describes the data, so we don't need any data. But if the hypothesis labels some points wrong, so we need the correct labels for these points to be able to come up with a better hypothesis.
    So basically this term captures the error.
- This means that the best hypothesis is the simplest hypothesis that minimizes error.

Bayesian Classification:

- The question in classification is "What is the best label?" not the best label.
- To find the best label, we need to do weighted vote for every single hypothesis in the hypotheses set, where the weight is the probability $P_r(h \mid D)$.
- Now we end up trying to maximize v_{map}

$$v_{map} = argmax_v \sum_h P_r(v \mid h)\, P_r(h \mid D)$$