Loss functions

Our treatment of Loss functions thus far has been somewhat superficial.

It's time to dive deeper.

To review

- The per example Loss is a measure of the success of a prediction on a **training** example
- Predictions are a function of parameters Θ
- "Fitting" or "training" a model is the process of find the best Θ
  - The optimal Θ is the one that minimizes average (across training examples) Loss

$$
\begin{array}{lll}
\mathcal{L}_\Theta^{(\mathbf{i})} & = & L(\ h(\mathbf{x}^{(\mathbf{i})}; \Theta), \mathbf{y}^{(\mathbf{i})}\ ) = L(\hat{\mathbf{y}}^{(\mathbf{i})}, \mathbf{y}) \\
\mathcal{L}_\Theta & = & \frac{1}{m} \sum_{i=1}^m \mathcal{L}_\Theta^{(\mathbf{i})}
\end{array}
$$

$$
\Theta^* = \operatorname*{argmin}_\Theta \mathcal{L}_\Theta
$$

The purpose of this module is to present a mathematical basis behind some of the common Loss functions in Machine Learning

- Mean Squared Error (MSE), used in Regression task
- Cross Entropy, used in Classification task
- Kullback Leibler (KL) Divergence

Likelihood

The statistical method known as Likelihood Maximization is the fundamental tool we will use.

Let us conceptualize our set of training examples

$$\langle \mathbf{X}, \mathbf{y} \rangle = [\mathbf{x}^{(\mathbf{i})}, \mathbf{y}^{(\mathbf{i})} | 1 \le i \le m]$$

as being *samples* from an unknown distribution

$$p_{\text{data}}(\mathbf{y} \mid \mathbf{x})$$

which we call the *true* or *actual* distribution

- *Conditional* distribution: target, conditional on feature

The distribution of the sample $\langle \mathbf{X}, \mathbf{y} \rangle$ is called the *empirical* distribution.

Given the actual distribution $p_{\text{data}}(\mathbf{x}, \mathbf{y})$,

- The likelihood (i.e, probability) of drawing the m particular examples in the training data
- Assuming independence, is

$$\mathbb{L}_{\text{data}} = \prod_{i=1}^{m} p_{\text{data}}(\mathbf{y^{(i)}} \mid \mathbf{x^{(i)}})$$

By taking the logarithm, we turn this product into a sum

$$\log(\mathbb{L}_{\text{data}}) = \sum_{i=1}^{m} \log\Big(p_{\text{data}}(y^{(\mathbf{i})} \mid \mathbf{x}^{(\mathbf{i})})\Big)$$

called the *Log Likelihood* of the data.

Note that the true process that generates examples is unknown to us; all we have is a sample (the empirical distribution) from the actual distribution.

We can *hypothesize* the existence of some process that generates the training data

- A Linear Model generates examples $\hat{\mathbf{y}}^{(\mathbf{i})} = \Theta^T \cdot \mathbf{x}^{(\mathbf{i})}$
- There may be more complex models we can suggest

But, given our limitations, our hypothetical model does not exactly match our examples

$$\mathbf{y^{(i)}} = \mathbf{\hat{y}^{(i)}} + \epsilon^{(\mathbf{i})}$$

where $\epsilon^{(\mathbf{i})}$ is the error between our hypothesized $\mathbf{y^{(i)}}$ and the one we observe.

This means that the conditional distribution of targets is centered around the model (predicted) value $\mathbf{\hat{y}^{(i)}}$.

Our limitations

- Maybe we can only *measure* $\mathbf{y}^{(\mathbf{i})}$ with error
- There is a missing feature that caused the error
    - Had this feature been included, there would be no error
        - Example: from everything I know about you, I observe that you *never* buy coffee at night
        - One night you buy coffee -- to bring to a friend, which is not a feature we capture

Putting this all together

- The observed error is defined as
$$\epsilon^{(\mathbf{i})} = \mathbf{y}^{(\mathbf{i})} - \hat{\mathbf{y}}^{(\mathbf{i})}$$
- A linear hypothesis for the true distribution is
$$\hat{\mathbf{y}} = \Theta^T \cdot \mathbf{x}$$
- The observed targets differ from our hypothesis by the observed error

$$\mathbf{y}^{(\mathbf{i})} = \Theta^T \cdot \mathbf{x}^{(\mathbf{i})} + \epsilon^{(\mathbf{i})}$$

Do these equations look familiar ?

- These are exactly the equations for Linear Regression
- Only the story we told is different
    - We didn't start with the goal of approximating $\mathbf{y}$
    - We adopt the standpoint that $\mathbf{y}$ differs from the hypothesized $\hat{\mathbf{y}}$ because of some error

Our hypothesis (now referred to as the *model*) is parameterized by Θ.

It implies a conditional distribution of targets

$$p_{\text{model}}(\mathbf{y} \mid \mathbf{x}; \Theta)$$

called the *model* or *predicted* distribution

Predicted distribution $p_{\text{model}}(\mathbf{y} \mid \mathbf{x}; \Theta)$ is an *approximation* of actual distribution $p_{\text{data}}(\mathbf{y} \mid \mathbf{x})$.

We now refer to the model values $\hat{\mathbf{y}}^{(\mathbf{i})}$ as *predictions*

Maximum Likelihood Estimation (MLE)

We introduce the statistical concept known as Maximum Likelihood Estimation (MLE)

We will subsequently relate our Loss functions to MLE.

Suppose the true process $p_{\text{data}}(\mathbf{x}, \mathbf{y})$ is

$$\mathbf{y^{(i)}} = 2 * \mathbf{x^{(i)}}$$

If our model

$$p_{\text{model}}(\mathbf{y} \mid \mathbf{x}; \mathbf{\Theta})$$

is

$$\mathbf{y^{(i)}} = 1 + 3 * \mathbf{x^{(i)}} + \epsilon^{\mathbf{(i)}}$$

then the errors $\epsilon^{\mathbf{(i)}}$ are systematically incorrect.

- Mean error won't be 0
- σ, the standard deviation of errors, won't be small

What this means is that

- Under the model's poor assumptions for Θ
- It is *less likely* for us to draw the m particular examples in the training data
    - Compared to an assumption for Θ that is closer to the actual

The likelihood under the model is written

$$\mathbb{L}_{\text{model}} = \prod_{i=1}^{m} p_{\text{model}}(\mathbf{y}^{(\mathbf{i})} \mid \mathbf{x}; \mathbf{\Theta})$$

- With a poorly chose Θ, errors are large, and the likelihood is small.

Under this framework, the best model

- Is the choice of Θ
- That maximizes the likelihood of drawing the m particular examples in the training data

This is called *Maximum Likelihood Estimation*

$$\Theta^* = \operatorname*{argmax}_\Theta \sum_{i=1}^{m} \log(p_{\text{model}}(\mathbf{y^{(i)}} \mid \mathbf{x^{(i)}}; \Theta))$$

(Note that maximizing the *log* likelihood is the same as maximizing the likelihood).

[Deep Learning Book 5.5 (https://www.deeplearningbook.org/contents/ml.html)](https://www.deeplearningbook.org/contents/ml.html)

Loss functions for Machine Learning

We now show that our choice of Loss functions

- MSE for Regression
- Cross Entropy for Classification

can be justified in terms of **maximization of the log likelihood**.

Regression: Log Likelihood of Linear models with normal errors

Under the hypothesis of a Linear Model, we have

$$\begin{aligned}
\mathbf{y}^{(\mathbf{i})} &= \hat{\mathbf{y}}^{(\mathbf{i})} + \epsilon^{(\mathbf{i})} \\
\hat{\mathbf{y}}^{(\mathbf{i})} &= \Theta^T \cdot \mathbf{x}^{(\mathbf{i})} \\
\epsilon^{(\mathbf{i})} &= \mathbf{y}^{(\mathbf{i})} - \hat{\mathbf{y}}^{(\mathbf{i})}
\end{aligned}$$

We had not previously made any assumption about the nature of $\epsilon^{(\mathbf{i})}$

Suppose it is normally distributed

$$\epsilon^{(\mathbf{i})} = \mathcal{N}(0, \sigma)$$

This means $p(\mathbf{y}^{(\mathbf{i})} \mid \mathbf{x}^{(\mathbf{i})}; \Theta)$ is $\mathcal{N}(\hat{\mathbf{y}}^{(\mathbf{i})}, \sigma)$

Substituting the formula for Normal distribution, the conditional probability of $\mathbf{y^{(i)}}$ given $\mathbf{x^{(i)}}$ is

$$\begin{aligned}
p(\mathbf{y^{(i)}} \mid \mathbf{x^{(i)}}; \Theta) &= \frac{1}{\sigma\sqrt(2\pi)}\exp(-\frac{(\mathbf{y^{(i)}} - \mathbf{\hat{y}^{(i)}})^2}{2\sigma}) & p(\mathbf{y^{(i)}} \mid \mathbf{x^{(i)}}; \Theta) \text{ is } \mathcal{N}(\mathbf{\hat{y}^{(i)}}, \sigma); \\
& & \text{def. of Normal} \\
&= \frac{1}{\sigma\sqrt(2\pi)}\exp(-\frac{(\epsilon^{\mathbf{(i)}})^2}{2\sigma}) & \epsilon^{\mathbf{(i)}} = \mathbf{y^{(i)}} - \mathbf{\hat{y}^{(i)}} \\
&\propto \exp(-\frac{(\epsilon^{\mathbf{(i)}})^2}{2\sigma})
\end{aligned}$$

The Likelihood of the training set, given this model of the conditional probability, is just the product over the training set of $p(\mathbf{y^{(i)}}|\mathbf{x^{(i)}})$:

$$\mathbb{L}_{\text{model}} = \prod_{i=1}^{m} p(\mathbf{y^{(i)}} \mid \mathbf{x^{(i)}}; \mathbf{\Theta})$$

and the Log Likelihood is

$$\begin{aligned}
\text{l}_{\text{model}} &= \log\left(\prod_{i=1}^{m} p(\mathbf{y^{(i)}} \mid \mathbf{x^{(i)}}; \mathbf{\Theta})\right) \\
&= \sum_{i=1}^{m} \log\left(p(\mathbf{y^{(i)}} \mid \mathbf{x^{(i)}}; \mathbf{\Theta})\right) \\
&\propto \sum_{i=1}^{m} \log\left(\exp(-\frac{(\epsilon^{(\mathbf{i})})^2}{2\sigma})\right) \\
&= \sum_{i=1}^{m} -\frac{(\epsilon^{(\mathbf{i})})^2}{2\sigma} \\
&= -\frac{1}{2\sigma} \sum_{i=1}^{m} (\mathbf{y^{(i)}} - \mathbf{\Theta}^T \cdot \mathbf{x^{(i)}})^2 \\
&\propto -\sum_{i=1}^{m} (\mathbf{y^{(i)}} - \mathbf{\Theta}^T \cdot \mathbf{x^{(i)}})^2
\end{aligned}$$

You should recognize the (negative of) the Log Likelihood as the Mean Squared Error (MSE).

Thus minimizing MSE Loss function, which we originally presented without justification

- Is equivalent to finding the model that maximizes the likelihood of the actual distribution

Stated another way

- The MSE Loss
- Gives rise to the Θ obtained by MLE

Classification: Log Likelihood of Binary classification

Review of Binary Classification:

- We encode the Positive labels $\mathbf{y^{(i)}}$ with the number 1 and Negative labels with the number 0
- For example i we compute $\hat{p}^{\mathbf{(i)}} = p(\mathbf{y^{(i)}} = \text{Positive} \mid \mathbf{x^{(i)}})$

The conditional probability for a target $\mathbf{y}^{(\mathbf{i})}$ given the features $\mathbf{x}^{(\mathbf{i})}$ can be written as

$$\begin{aligned} p(\hat{\mathbf{y}}^{(\mathbf{i})} \mid \mathbf{x}^{(\mathbf{i})}; \Theta) &= p(\mathbf{y}^{(\mathbf{i})} = \text{Positive} \mid \mathbf{x}^{(\mathbf{i})}; \Theta) \quad + \quad p(\mathbf{y}^{(\mathbf{i})} = \text{Negative} \mid \mathbf{x} \\ &= p(\mathbf{y}^{(\mathbf{i})} = \text{Positive} \mid \mathbf{x}^{(\mathbf{i})}; \Theta)^{\mathbf{y}^{(\mathbf{i})}} \quad * \quad p(y^{(\mathbf{i})} = \text{Negative} \mid \mathbf{x} \end{aligned}$$

Substituting the above probability into the Likelihood

$$\mathbb{L}_{\text{model}} = \prod_{i=1}^{m} p_{\text{model}}(\mathbf{y^{(i)}} \mid \mathbf{x^{(i)}}; \boldsymbol{\Theta})$$

and taking the log $$ \begin{array}[lllll]\ \mathcal{l} & = & \log(\likeli*{\text{model}}) \ \mathcal{l} & = & \sum*{i=1}^m { \y^\ip * \log \left(\prc{\y^\ip = \text{Positive}}{\x^\ip; \Theta}\right)

- (1-\y^\ip) *\log \left(\prc{\y^\ip = \text{Negative}}{\x^\ip; \Theta}\right)} \ & = & \sum_{i=1}^m { \y^\ip* \log(\hat{p}^\ip) + (1-\y^\ip) * \log(1 - \hat{p}^\ip)} \ \end{array} $$

Recalling the per-example Loss function for Binary Classification

$$\begin{aligned}
\mathcal{L}_{\Theta}^{(\mathbf{i})} &= - \left(\mathbf{y}^{(\mathbf{i})} * \log(\hat{p}^{(\mathbf{i})}) + (1 - \mathbf{y}^{(\mathbf{i})}) * \log(1 - \hat{p}^{(\mathbf{i})}) \right) \\
\mathcal{L}_{\Theta} &= \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}_{\Theta}^{(\mathbf{i})}
\end{aligned}$$

You see that $\frac{1}{m}$ times the negative of the Log Likelihood is equal to the Binary Cross Entropy Loss.

Thus minimizing Binary Cross Entropy loss, which we originally presented without justification

- Is equivalent to finding the model that maximizes the likelihood of the actual distribution

Stated another way

- The Binary Cross Entropy Loss
- Gives rise to the Θ obtained by MLE

KL divergence

We can now motivate the KL divergence:

- The difference between the log likelihood of the empirical and model distributions.

Bayes Theorem relates joint and conditional probabilities

$$\begin{aligned}
p(\mathbf{y} \mid \mathbf{x}) &= \tfrac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{x})} \\
p(\mathbf{x}, \mathbf{y}) &= p(\mathbf{y} \mid \mathbf{x})\, p(\mathbf{x}) \quad \text{re-arrange the terms}
\end{aligned}$$

So we can re-write

$$\begin{aligned}
\log(\mathbb{L}_{\text{model}}) &= \textstyle\sum_{i=1}^{m} \log\big(p_{\text{model}}(\mathbf{x^{(i)}}, \mathbf{y^{(i)}}; \boldsymbol{\Theta})\big) \\
&= \textstyle\sum_{i=1}^{m} \log\big(p_{\text{model}}(\mathbf{y^{(i)}} | \mathbf{x^{(i)}}; \Theta)\big)\ p(\mathbf{x^{(i)}}) \\
&= \mathbf{E}_{\mathbf{x} \sim p_{\text{data}}} \log\big(p_{\text{model}}(\mathbf{y^{(i)}} | \mathbf{x^{(i)}}; \Theta)\big)
\end{aligned}$$

and similarly for $\log(\mathbb{L}_{\text{data}})$

The difference between the log likelihoods of the two distributions

$$\log(\mathbb{L}_{\text{data}}) - \log(\mathbb{L}_{\text{model}}) \quad = \quad \mathbf{E}_{\mathbf{x} \sim p_{\text{data}}}(\log(p_{\text{data}}(\mathbf{y}|\mathbf{x}))) - \mathbf{E}_{\mathbf{x} \sim p_{\text{data}}}(\log(p_{\text{model}}(\mathbf{y}|$$

The above difference is called the *KL Divergence* between the distributions.

It is a measure of the "closeness" of two distributions.

This means

- Minimizing KL Divergence between the actual and predicted distributions
- Is equivalent to minimizing the difference between the log likelihoods of the distributions

The optimal $p_{\text{model}}(\mathbf{y}|\mathbf{x};\Theta))$ is the one with smallest KL Divergence

Since only $p_{\text{model}}(\mathbf{y}|\mathbf{x};\Theta))$ is a function of Θ, the Θ that minimizes KL Divergence is found by minimizing $$

- \E_{\x \sim \pdata}{ \left(\log(\pmodel(\y \; | \; \x; \Theta)) \right) } $$ which is expression for the Cross Entropy Loss.

Thus minimizing Cross Entropy Loss, which we originally presented without justification

- Is equivalent to minimizing the KL Divergence between the actual and predicted distributions
- Is equivalent to minimizing the difference between the log likelihoods of the distributions

Unsupervised Learning

Although we have not yet covered Unsupervised Learning, we observe that Likelihood Maximization can be applied there as well

- Unsupervised Learning has no targets
- So training examples
$$\langle \mathbf{X} \rangle = [\mathbf{x^{(i)}} | 1 \le i \le m]$$
- The distribution is over features, not of targets conditional on features
$$p_{\text{data}}(\mathbf{x})$$

Loss functions for Deep Learning: Preview

The Loss functions for Classical Machine Learning were perhaps motivated by the desire for closed form solutions.

In Deep Learning, the optimization is typically solved via search.

This opens the possibilities of complex loss functions that don't require closed form solution.

As we will see in the Deep Learning part of this course, the key part of solving a task is in *defining* a loss function that mirrors the task's objective.

Thus, many loss functions are problem specific and often quite creative.

Cool loss functions: Neural Style Transfer

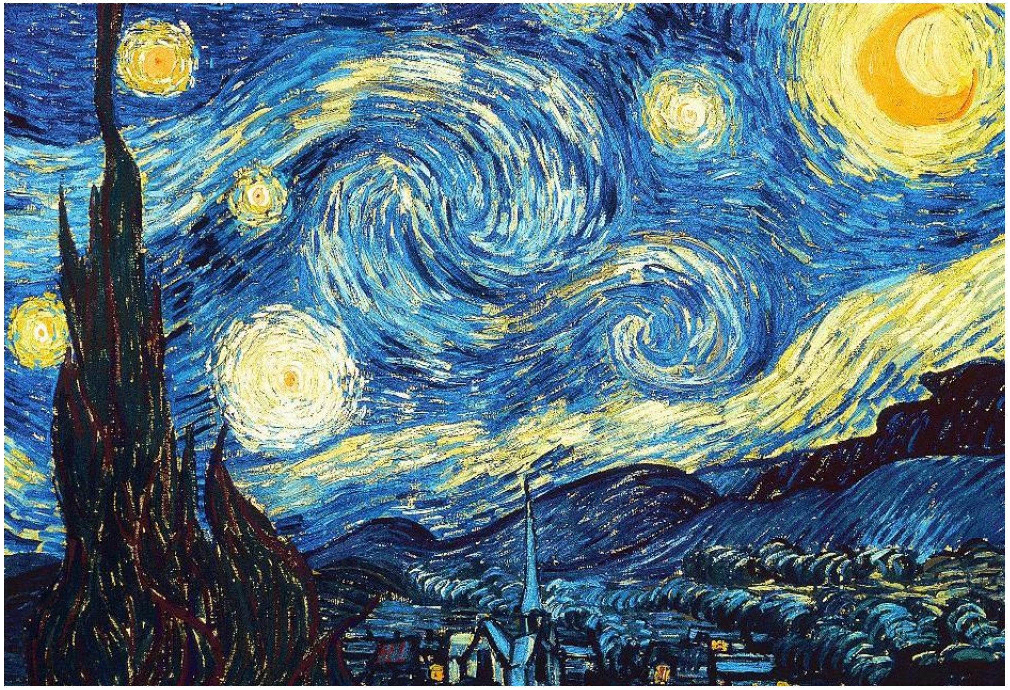
Neural Style Transfer

Given

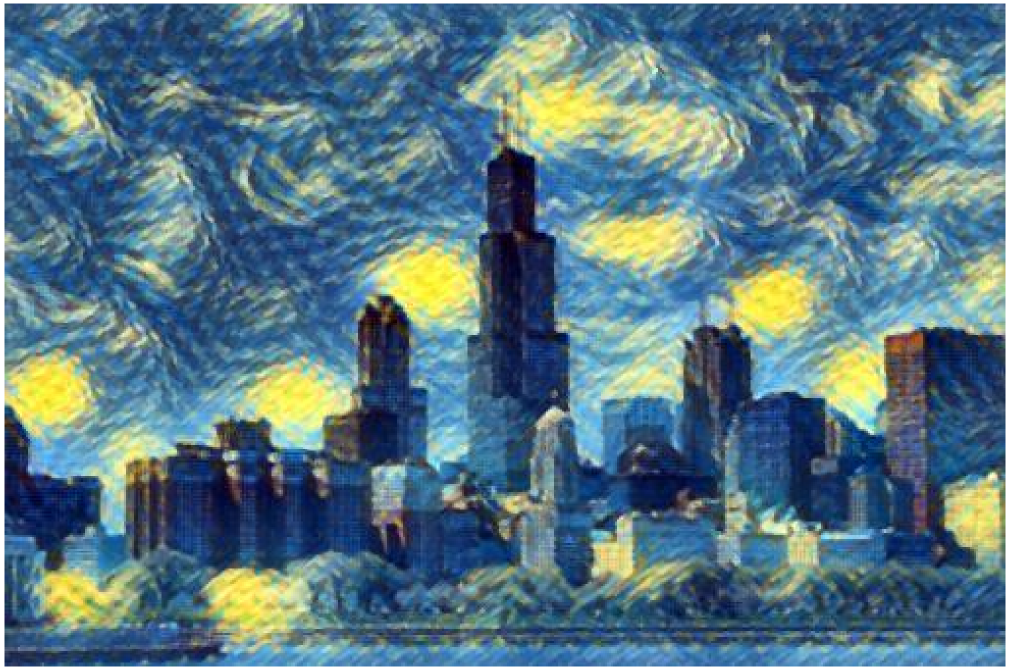
- a "Content" Image that you want to transform
- a "Style" Image (e.g., Van Gogh "Starry Night")
- Generate a New image that is the Content image redrawn in the style of the Style Image
    - Gatys: A Neural Algorithm for Style (https://arxiv.org/abs/1508.06576)
    - Fast Neural Style Transfer (https://github.com/jcjohnson/fast-neural-style)

Content image

Style image



Generated image



Loss function

Definitions:

- Style image, represented as a vector of pixels $\vec{a}$
- Content image, represented as a vector of pixels $\vec{p}$
- Generated image, represented as a vector of pixels $\vec{x}$

The Loss function (which we want to minimize by varying $\vec{x}$) has two parts

$$\mathrm{L} = \mathrm{L}_{\text{content}}(\vec{p}, \vec{x}) + \mathrm{L}_{\text{style}}(\vec{a}, \vec{x})$$

- a Content Loss
    - measure of how different the generated image $\vec{x}$ is from Content image $\vec{p}$
- a Style Loss
    - measure of how different the "style" of generated $\vec{x}$ is from style of Style image $\vec{a}$

Key: defining what is "style" and similarity of style

In [3]:
```
print("Done")
```

```
Done
```