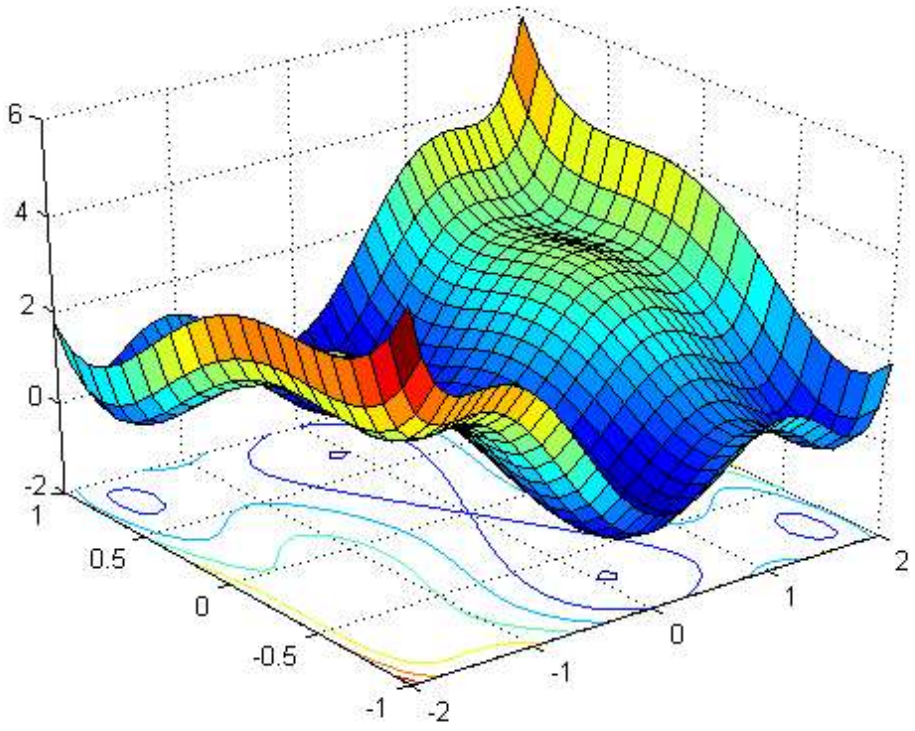
ALGORITHMIA

APRIL 30, 2018

Introduction to Loss Functions



The loss function is the bread and butter of modern Machine Learning; it takes your algorithm from theoretical to practical and transforms neural networks from glorified matrix multiplication into *Deep Learning*.

This post will explain the role of loss functions and how they work, while surveying a few of the most popular of the past decade.

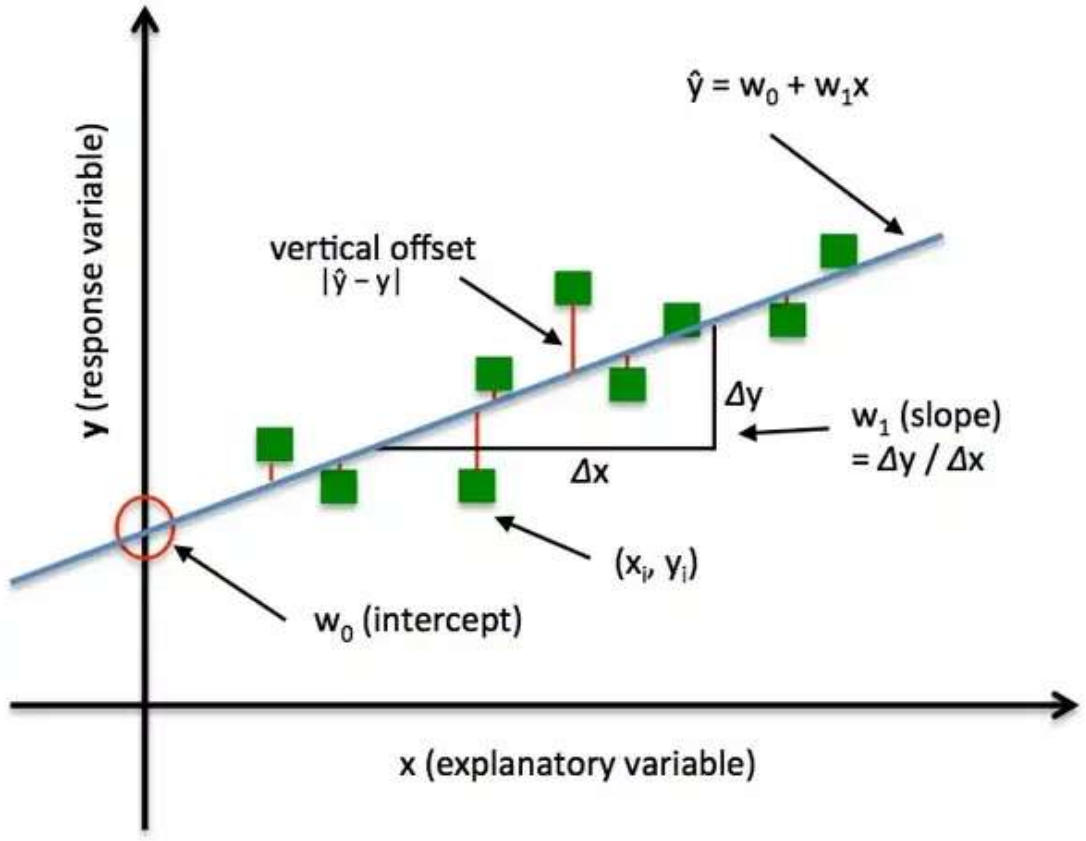
What's A Loss Function?

At its core, a loss function is incredibly simple: it's a method of evaluating how well your algorithm models your dataset. If your predictions are totally off, your loss function will output a higher number. If they're pretty good, it'll output a lower one. As you change pieces of your algorithm to try and improve your model, your loss function will tell you if you're getting anywhere.

In fact, we can design our own (very) basic loss function to further explain how it works. For each prediction that we make, our loss function will simply measure the absolute difference between our prediction and the actual value. In mathematical notation, it might look something like *abs(y_predicted – y).* Here's what some situations might look like with if we were trying to predict how expensive the rent is in some NYC apartments:

Our Predictions	Actual Values	Our Total Loss
Harlem: $1,000 SoHo: $2,000 West Village: $3,000	Harlem: $1,000 SoHo: $2,000 West Village: $3,000	0 (we got them all right!)
Harlem: $500 SoHo: $2,000 West Village: $3,000		500 (we were off by $500 in Harlem)
Harlem: $500 SoHo: $1,500 West Village: $4,000		2000 (we were off by $500 in Harlem, $500 in SoHo, and $1,000 in the West Village)

Notice how in the loss function *we* defined, it doesn't matter if our predictions were too high or too low? All that matters is how incorrect we were, directionally agnostic. This is *not* a feature of all loss functions: in fact, your loss function will vary significantly based on the domain and unique context of the problem that you're applying Machine Learning to. In your project it may be much worse to guess too high than to guess too low, and the loss function you select must reflect that.



Different Types And Flavors of Loss Functions

A lot of the loss functions that you see implemented in cutting edge Machine Learning today can get complex and confusing. Consider this paper from late 2017, titled *A Semantic Loss Function for Deep Learning with Symbolic Knowledge*. There's more in that title that I *don't understand* than I do. But if you remember the end goal of all loss functions--measuring how well your algorithm is doing on your dataset--you can keep that complexity in check.

We'll run through a few of the most popular loss functions currently being used, from simplest to most complex.

Mean Squared Error

Mean Squared Error (or MSE for short) is the workhorse of basic loss functions: it's easy to understand and implement, and generally works pretty well. To calculate MSE, you take the difference between your predictions and the ground truth, square it, and average it out across the whole dataset.

Implemented in code, MSE might look something like:

```
def MSE(y_predicted, y):
squared_error = (y_predicted - y) ** 2
sum_squared_error = np.sum(squared_error)
mse = sum_squared_error / y.size
return(mse)
```

Likelihood Loss

The likelihood function is also relatively simple, and is commonly used in classification problems. The function takes the predicted probability for each input example and multiplies them. And although the output isn't exactly human interpretable, it's useful for comparing models.

For example, consider a model that outputs probabilities of [0.4, 0.6, 0.9, 0.1] for the ground truth labels of [0, 1, 1, 0]. The likelihood loss would be computed as (0.6) * (0.6) * (0.9) * (0.9) = 0.2916. Since the model outputs probabilities for TRUE (or 1) only, when the ground truth label is 0 we take (1-p) as the probability. In other words, we multiply the model's outputted probabilities together for the actual outcomes.

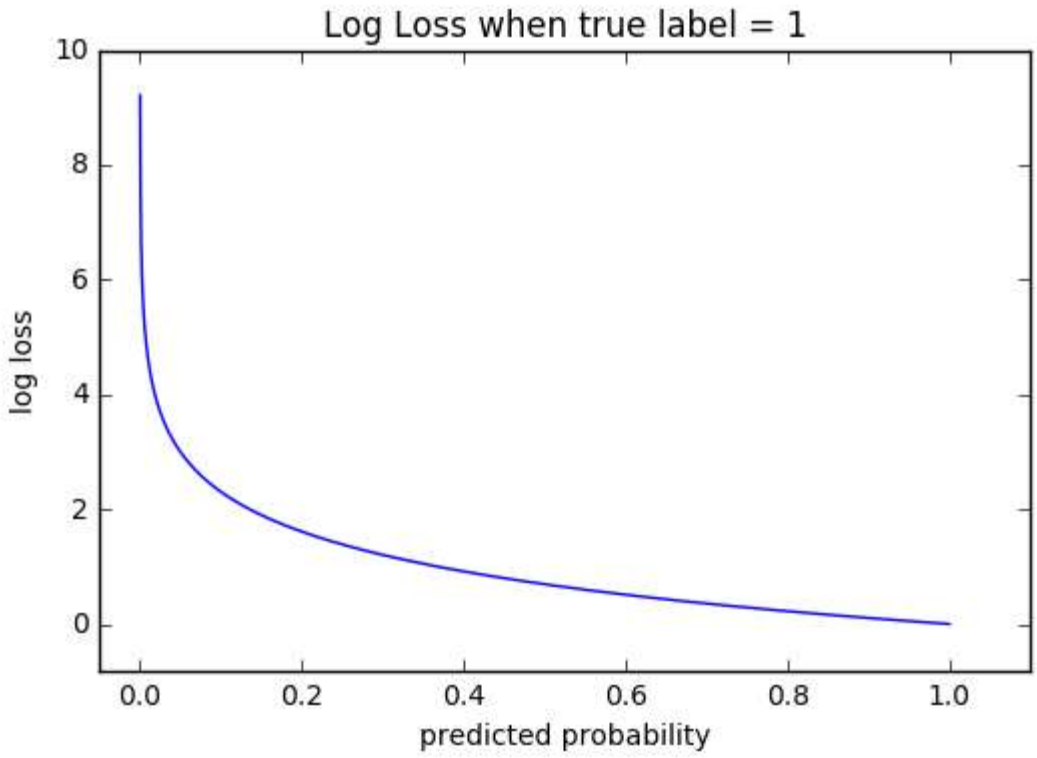
Log Loss (Cross Entropy Loss)

Log Loss is a loss function also used frequently in classification problems, and is one of the most popular measures for Kaggle competitions. It's just a straightforward modification of the likelihood function with logarithms.

$$-(y\log(p)+(1-y)\log(1-p))$$

This is actually exactly the same formula as the regular likelihood function, but with logarithms added in. You can see that when the actual class is 1, the second half of the function disappears, and when the actual class is 0 the first half drops. That way, we just end up multiplying the log of the actual predicted probability for the ground truth class.

The cool thing about the log loss loss function is that is has a kick: it penalizes heavily for being *very confident* and then *very wrong*. Predicting high probabilities for the wrong class makes the function go crazy. The graph below is for when the true label =1, and you can see that it skyrockets as the predicted probability for label = 0 approaches 1.



Source: Fast.ai

Loss Functions and Optimizers

Loss functions provide more than just a static representation of how your model is performing – they're how your algorithms fit data in the first place. Most Machine Learning algorithms use some sort of loss function in the process of optimization, or finding the best parameters (weights) for your data.

For a simple example, consider Linear Regression. In traditional "least squares" regression, the line of best fit is determined through none other than Mean Squared Error (hence the least squares moniker)! For each set of weights that the model tries, the MSE is calculated across all input examples. The model then optimizes the MSE functions--or in other words makes it the lowest possible--through the use of an optimizer algorithm like Gradient Descent.

Just like there are different flavors of loss functions for unique problems, there is no shortage of different optimizers as well. That's beyond the scope of this post, but suffice it to say that the loss function and optimizer work in tandem to fit the algorithm to your data in the best way possible.

Further Reading

Choosing the Right Metric for Evaluating Machine Learning Models – Part 1 (KDNuggets) – "*Each machine learning model is trying to solve a problem with a different objective using a different dataset and hence, it is important to understand the context before choosing a metric. In this post, I will be discussing the usefulness of each error metric depending on the objective and the problem we are trying to solve.*"

Bayesian Methods for Hackers: Would You Rather Lose an Arm or a Leg? (Informit) – "*The important point of loss functions is that they measure how bad our current estimate is: The larger the loss, the worse the estimate is according to the loss function. A simple, and very common, example of a loss function is the squared-error loss, a type of loss function that increases quadratically with the difference, used in estimators like linear regression, calculation of unbiased statistics, and many areas of machine learning.*"

Picking Loss Functions: A Comparison Between MSE, Cross Entropy, And Hinge Loss (Rohan Varma) – "*Loss functions are a key part of any machine learning model: they define an objective against which the performance of your model is measured, and the setting of weight parameters learned by the model is determined by minimizing a chosen loss function. There are several different common loss functions to choose from: the cross-entropy loss, the mean-squared error, the huber loss, and the hinge loss – just to name a few.*"

Some Thoughts About The Design Of Loss Functions (Paper) – "T*he choice and design of loss functions is discussed. Particularly when computational methods like cross-validation are applied, there is no need to stick to "standard" loss functions such as the L2-loss (squared loss). Our main message is that the choice of a loss function in a practical situation is the translation of an informal aim or interest that a researcher may have into the formal language of mathematics.*"

A More General Robust Loss Function (Paper) – "*We present a two-parameter loss function which can be viewed as a generalization of many popular loss functions used in*

robust statistics: the Cauchy/Lorentzian, Geman-McClure, Welsch/Leclerc, and generalized Charbonnier loss functions (and by transitivity the L2, L1, L1-L2, and pseudo-Huber/Charbonnier loss functions). We describe and visualize this loss and its corresponding distribution, and document several of their useful properties."

Videos and Lectures

Loss Functions And Optimization (Stanford) – *"Lecture 3 continues our discussion of linear classifiers. We introduce the idea of a loss function to quantify our unhappiness with a model's predictions, and discuss two commonly used loss functions for image classification: the multiclass SVM loss and the multinomial logistic regression loss. We introduce the idea of regularization as a mechanism to fight overfitting, with weight decay as a concrete example."*

Risk And Loss Functions: Model Building And Validation (Udacity) – *Part of the Model Building and Validation Course.*

Logistic Regression Cost Function (Coursera) – *Part of Andrew Ng's Machine Learning course on Coursera.*

Justin Gage

More Posts

Follow Me:

Search

Enter your query here...

SIGN UP

A.I. Topic Guides

Algorithm Spotlight

Blog Posts

Content Hub

Demos

Developer Spotlight

Emergent Future

Events

Guest Post

Integrations

Newsletter

Recipes

Algorithmia

AI in every application.