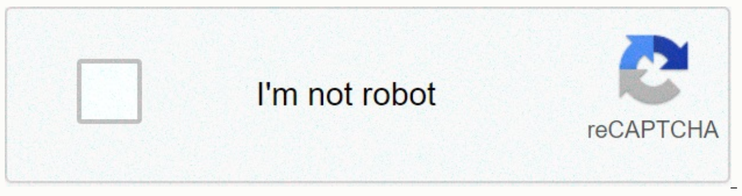

Continue

Classification report in machine learning

;Originally published in Andreessen Horowitz's AI Playbook.There are four main ways to train deep learning networks: supervised, unsupervised, semisurvised, and reinforcement learning. We'll explain the intuitions behind each of these methods. Along the way, we will share terms that you will read in literature in parentheses and point to more resources for the mathematically inclined. By the way, these categories cover both traditional machine learning algorithms and the newest and fancier deep learning algorithms. For mathematical leaning, see this Stanford tutorial that covers supervised and unsupply learning and includes code samples. Supervised learning Supervised learning trains networks using examples where we already know the correct answer. Imagine that we are interested in forming a network to recognize images from your photo library that have your parents in them. These are the steps we would take in this hypothetical scenario. Step 1: Creating and categorizing data sets We would start the process by going through your photos (the data set) and identifying all the images your parents have, tagging them. Then we would take the whole stack of photos and divide them into two stacks. We would use the first stack to train the network (training data) and the second stack to see the accuracy of the model when choosing photos with our parents (validation data). When the data sets are ready, we would feed the photos to the model. Mathematically, our goal is for the deep network to find a function whose entry is a photo and the output is a 0 when your parents are not in the photo or 1 when they are. This step is usually called a categorization task. In this case we are training for results that are yes-no, but supervised learning can also be used to generate a set of values, rather than just a 0 or 1. For example, we could train a network to generate the likelihood of someone returning a credit card loan, in which case the outing is between 0 and 100. These tasks are called regressions. Step 2: Training To continue the process, the model makes a prediction for each photo following rules (activation function) to decide whether to illuminate a particular node at work. The model works from left to right one layer at a time – let's ignore the most complicated networks for the time being. After the network calculates this for each node on the network, we will reach the right-hander node (output node) that lights up or not. Since we already know what images their parents have in them, they would be able to tell the model if their prediction is correct or incorrect. We would then support this information on the network. The algorithm uses this feedback, which is the result of a function that quantifies the extent to which the response is far from the prediction of the model. This is called cost function, also known as objective function, utility function or fitness function. The result of the function is then used to modify the strength of connections and bias biases nodes in a process called backpropagation since the information travels backwards from the results nodes. We would repeat this for each of the images, and in each case the algorithms try to minimize the cost function. There are a variety of mathematical techniques to use this knowledge of whether the model was right or wrong back in the model, but a very common method is gradient descent. Algobeans has a good explanation for how this works. Michael Nielsen adds the math involving calculus and linear algebra (and a friendly demon!). Step 3: VerifyOnce we have processed all the photos from our first stack will be ready to test the model. We would take the second stack of photos and use them to see how accurately the trained model can collect photos from your parents. Steps 2 and 3 would normally be repeated by tweaking various things about the model (hyperparameters), such as how many nodes there are, how many layers there are, what mathematical function to use to decide whether a node lights up, how aggressively to train weights during the backpropagation phase, and so on. This quora answer has a good explanation for the knobs it can turn. Step 4: UseFinally, once you have an accurate model, expand this model in the app. Expose the model as an API call, such as ParentsInPicture(photo), and you can call this method from your software, causing the model to make an inference and gives you the result. Let's go through this exact process later by writing an iPhone app that recognizes business cards. It can be difficult (i.e. expensive) to get a labeled data set, so make sure that the prediction value justifies the cost of getting the tagged data and forming the model first. For example, getting X-rays tagged from people who might have cancer is expensive, but the value of an accurate model that generates few false positives and few false negatives is obviously very high. LearningUnsupervised Learning Unsupervised It's for situations where you have a data set but no labels. Unsu supervised learning takes the set of entries and tries to find patterns in the data, for example by organizing them into clustering groups or finding outliers (detection of anomalies). For example: imagine that you are a T-shirt manufacturer, and you have a lot of people's body measurements. You would like a clustering algorithm that groups these measurements into a cluster set so you can decide how big your XS, S, M, L and XL shirts are. You're the WTO of a security startup and want to find anomalies in the history of network connections between computers: network traffic that seems unusual could help you find an employee downloading their entire CRM history because they're about to quit or someone transferring an abnormally large amount of money to a bank account If you're interested in this kind of thing, you'll like this survey of unsu supervised anomaly detection algorithms. You're on the Google Brain team, and you on YouTube videos. This is the very real story of YouTube's Cat Finder research that hailed the general public's enthusiasm for AI. In this article, the Google Brain team along with Stanford researchers Quoc Le and Andrew Ng describe an algorithm that groups YouTube videos into a sled of categories, including one that contained cats. No cats were set to be found, but the algorithm automatically grouped videos containing cats (and thousands of other objects from the 22,000 categories of objects defined on ImageNet) together without any explicit training data. Some unsupervised learning techniques you'll read in literature include: AutoencodingPrincipal components analysisRandom forestsK-means clusteringFor more information about unsupervised learning, try this Udacity class. One of the most promising recent developments in unsuvised learning is an idea by Ian Goodfellow (who was working in Yoshua Bengio's lab at the time) called generative adversary networks in which we faced two neural networks: a network, called a generator, is responsible for generating data designed to try to deceive the other network, called a discriminator. This approach is achieving some amazing results, such as AI that can generate photorealistic images from hand-drawn text strings or sketches. Semi-supervised learning by LearningSemi combines a lot of unlabeled data with a small amount of tagged data during the training phase. The trained models resulting from this set of workouts can be very accurate and less expensive to train compared to using all the tagged data. Our friend Delip Rao at intelligence consultancy IA Joostware, for example, built a solution using semi-supervised learning using only 30 labels per class that got the same precision as a trained model using supervised learning that required ~1360 tags per class. This allowed his client to scale his prediction capabilities from 20 categories to 110 categories very quickly. An intuition behind why using unlabeled data can sometimes help make models more accurate: even if you don't know the answer, you're learning something about what the possible values are and how often specific values appear. Math fans: Try Xiaojin Zhu's epic 135-slide tutorial and the document accompanying literature in 2008.Reinforcement LearningReinforcement learning is for situations where you don't have tagged data sets again, but you have a way of saying if you're getting closer to your goal (reward function). The classic hottest or coldest children's game (a variant of Huckle Buckle Beanstalk) is a good concept. Your job is to find a hidden object, and your friends will shout if you are getting hotter (closer) or colder (farther from) the object. Hotter/colder is the reward function, and the goal of the algorithm is to maximize the reward function. You may think that the reward function is a delay Scarce form of tagged data: instead of getting a correct/incorrect specific response with each data point, you'll receive a delayed reaction and only a hint of whether you're heading in the right direction. DeepMind published an article in Nature describing a system that combines reinforcement learning with deep learning to learn how to play a set of Atari video games, some with great success (such as Breakout) and others terribly (such as Montezuma revenge). The Nervana team (now on Intel) published an excellent explanatory blog post that scours the techniques in detail. A very creative Stanford student project by Russell Kaplan, Christopher Sauer, Alexander Sosa illustrates one of the challenges with reinforcement learning and suggests a smart solution. You'll see in the DeepMind newspaper that algorithms didn't learn to touch Montezuma's revenge. The reason for this is that, as Stanford students describe, reinforcement learning agents still struggle to learn in environments with measly rewards. When you don't get enough hotter or colder tracks, you have a hard time finding the hidden key. Stanford students basically taught the system to understand and respond to natural language tracks such as climbing the ladder or getting the key, making the system the highest-scoring algorithm in the OpenAI gym. Watch a video of the algorithm in action. Richard Sutton and Andrew Barto wrote the book on reinforcing learning. See the draft for the 2nd edition. Originally published in Andreessen Horowitz's AI Playbook.Join Hacker Noon Create your free account to unlock your personalized reading experience. Experience.