CS 763 09-18 summarization

2019-09-24

Robert Harlow
Shengwen Yang

1 Overview f attacks on the machine learning pipeline.

This week we are talking about two borad types of attacks.

A.) Adversarial example (also called test-time attacks). They target part 2.) of the machine learning pipeline.
–Find examples that lead to the "wrong" classification (a classification that clearly incorrect to a human) by taking advantage of the model's "blind spots".

B.) Data poisoning attacks (also called training-time attacks). The part 1.) of the machine learning pipeline.
–More over, they try to mainipute the trining example to effect the model that is learned.

1.1 Commonalities of adversarial examples and data poisoning attacks.

Both adversarial examples and data poisoning attacks work by messing with one of the core assumptions behind machine learning. That core assuption is: Training and test examples are drawn from some common distribution of examples. Moreover, training and test example are independently and identicailly drawn (iid) from that common distribution.

2 Adversarial Examples.

Justin discussed two papers on adversarial examples.

3 Paper #1: Intriguing properties of neural networks.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199.

3.1 Contribution #1

Contribution #1: An individual neuron in the second to last layer of a neural network no more detects an "important feature" than a any random linear combination of such neurons does.

–Feature layer assumption: The second to last layer of a neural network is commonly refered to as the feature layer. The name feature layer comes from a belief (now contested) that each neuron the in feature layer detects the presence of one "important feature". For example, if the neural network classifies images as either cats or fish, then the persence of legs could be one such "important feature".

–Szegedy et al. Test this assumption by running a large sets of images through a neural network (that was trained to classify such images). The outputs of each neuron in the feature layer are recorded for each image.

A. For each neuron in the feature layer, the images are ranked by the size of the output the image produced at that neuron.

B. For a set of linear combinations of neurons in the feature layer, the images are ranked by the size of the weighted sum of the outputs the image produced at those neurons.

By examining the top ranked images for various individual neurons and linear combinations of neurons and eyeballing the simularity of those images, the authors conclude that the top ranked images for a lienar combination of neurons are not gernerally more or less simular than the top ranked images for a particular neuron. This conclusion challenges validity of the Feature layer assumption (stated above).

3.2 Contribution #2: Adversarial Examples

–Szegedy et al. show that for a given neural network ("nn") and an image "x" that the network classifies as "c", it is usutally possible to find a small perturabtion of "x" (small meaning imperceptable to the human eye) such that the neural network classifies the purturbed image as "l" for every class "l" in the neural network's range (such that "l" is not equal to "c"). The perturbed image is called a "adversarial example".

–Szegedy et al. formulated an optimization problem that yields adversarial examples. Given image "x", "nn", "c", and "l" as defined in the paragraph above, the solution to the optimization problem is a image (a vector) "r" such that the l2 norm of "r" is "small" and "nn" classifies "x" + "r" as "l".

3.3 Contribution #2' (two prime): Transferability

–Szegedy et al. show that for two distinct (differing in one of the ways described below) models, "M1" and "M2", trained for the same classification task, adversarial examples for "M1" are often also adversarial examples for "M2". That is, "M2" classifies adversarial examples for "M1" incorrectly. Moreover,

"M1" and "M2" will often agree on the predicted class of such an adversarial example. Thus, adversarial examples are "transferable".

Szegedy et al. demonstrated "transferable" adversarial examples for all of the cases below:

A.) Model type: The two models differ in model type. (for example one may be a linear classifier and the other a convlutional neural network (CNN)).

B.) Model architecture: The two models differ in model architecture. (for example, both "M1" and "M2" might be convlutional neural networks (CNN) that differ in the quantity and size of hidden layers).

C.) Hyperparameters: The two models had different sets of hyperparameters during training. (for eaxmple, "M1" and "M2" might be trained with two different stochastic gradient descent (SGD) algorithms where the algorithms have different learning rates, number of training steps, and batch size.)

D.) Training examples: The two models were trained with different sets of training examples. (for example, "M1" and "M2" may be trained on disjoint sets of training data.)

4 Paper #2: Explaining and Harnessing Adversarial Examples.

Explaining and Harnessing Adversarial Examples

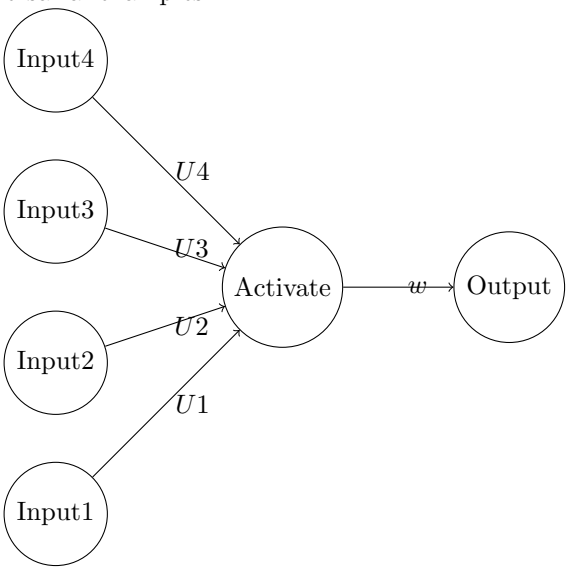
4.1 Contribution #1: Adversarial examples exist due to the linearity of neural networks.

Justin touched briefly touched on this contribution.

At a high level, Goodfellow et al. make a hieristic argument about why linearity of nerural networks lead to the existance of adversarial examples. The argument goes as follows:

–The activation function used in neural network are commonly peicewise linear (for example ReLU (rectified linear unit).)

–At a particular neuron, the output is thus often proportional to the dot product of the vector of the neurons weights "w" with the vector of the outputs of the previous layer "x". Consider the case where x is perturbed by a vector "r" with l-infinity norm less than epsilon (epsilon is choose to be small). If "r" is picked such that each term w_i * r_i has the same sign, than the net contribution of r on the dot product "w" DOT "x" + "r" will be large even though the l inifinty norm of r is small. (This is espeically true if the dimension of "x" is large.) The authors argue that this behavior leads to the existance of adversarial examples.



5 Contribution #2: The "fast gradient sign method".

–Goodfellow et al. introduce the "fast gradient sign method", a faster algorithm to find adversarial examples (faster than previously known algorithms.).

–Justin notes that the fast gradient sign method is no longer the strongest attack known today.

6 setup:

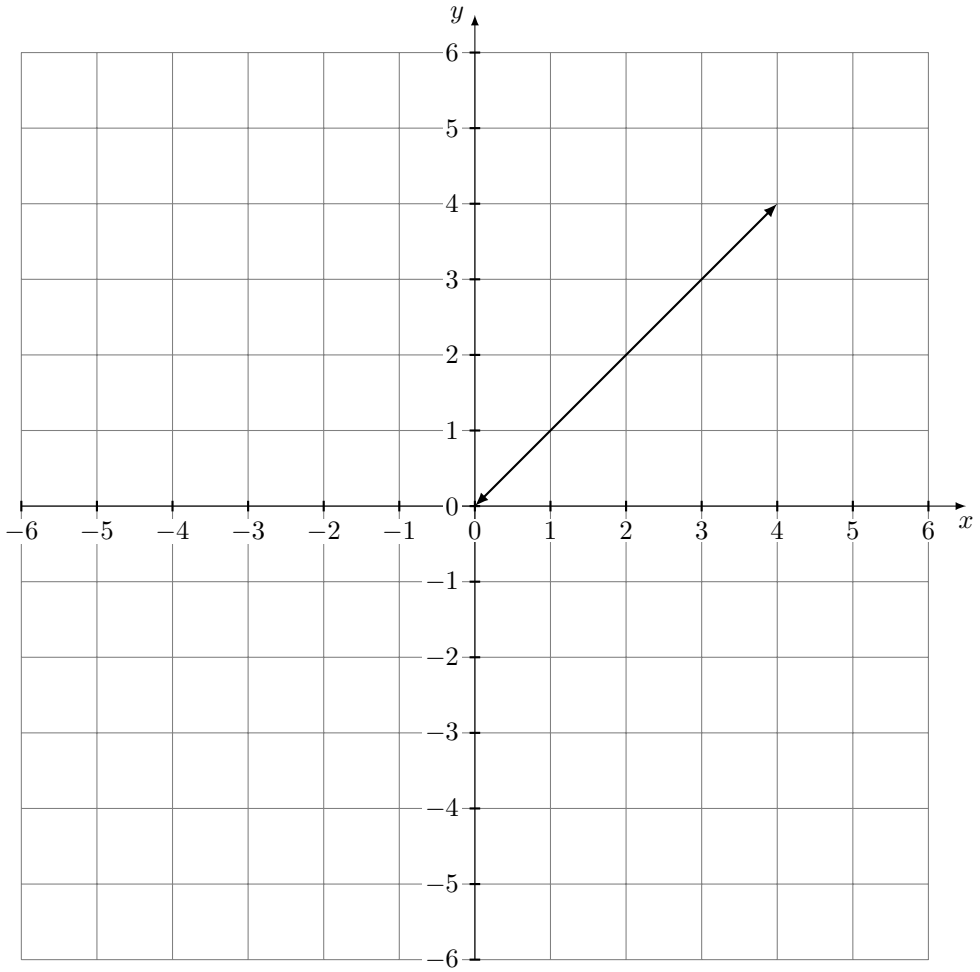
$$x \in Ximage, \tag{1}$$

fw is a model. perturbation r s.t.

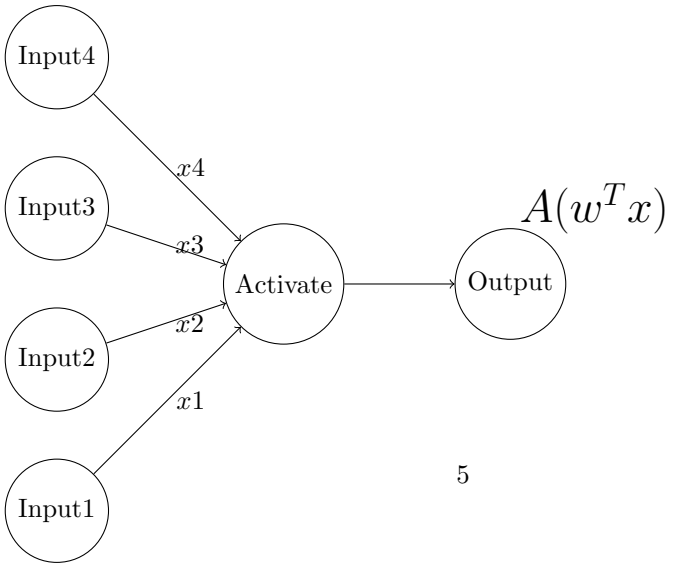
$$\|r\|_\infty \leq \epsilon \tag{2}$$

want:

$$f_w(x) \neq f_w(x+r). \tag{3}$$



6.1 Small perturbation → Large changes



$$A(w^T(x+r)) = A(w^T x + w^T r) \approx A(w^T x) + A(w^T r) \tag{4}$$

6.2 Fast gradient sign method (FGSM)

6.2.1 CostFunction:

$$J(w, x, y)(maybe|f_w(x) - y|) \tag{5}$$

6.2.2 Targeted attack

make x+r $\rightarrow$ y

$$f_w(x_*) = y_* \tag{6}$$

want to change prediction

6.2.3 set

$$r = \epsilon * sgn(\nabla_x * J(w, x, y_*)) * (X_*) \tag{7}$$

$$sgn(z) = \begin{cases} +1 & \text{if } z > 0 \\ 0 & \text{if } z = 0 \\ \text{-1} & \text{if } z < 0 \end{cases}$$

7 Full transcript.

So this might be SGD or something like that. And output somekind of model with parameters w. And then we can have the inference (or prediction). Some this means that I'll feed in some x, and I'll get out some example. So this is kind of the general machine learning pipeline that we're working with. And tyhe first thing we're going to talk about this week when we talk about attacks, there are two broad kinds of attacks that we're going to talk about so,...,the first one is called adversarial examples (that's today) and these are also called test-time attacks. So the idea with adversarial examples and test time attacks is that we're attacking this portion of the pipeline. Ok, so the training set is out there, some model is learned, and the model is kind of output. And then now we can kind of feed in different examples here and try to manipulate the prediciton that comes out of this specific model. So attacks against this part are adversarial examples.

The other attacks that we won't talk about today but we'll talk about on Friday are called data poisoning attacks. Data poisoning is also called training time attacks. So like you might guess they attack this other part of the pipeline. Where they try to manipulate the trying set by changing some examples or skewing them somehow in order to kind of effect which model is learned.

So these are two broad kind of attacks against machine learning pipeline. There are also other attacks that you can think of involving privacy concerns like model inversion. The also model theft like you're trying to figure out what the model is based on just this part or reverse enginerring certain things but at least two broad classes of attacks are data poisoning and adversarial examples.

So what's in common, one thing to keep in mind is that both of these attacks try to mess with some of the main assumptions behind machine learning. So they both try to play with the core asumption which is that: Assumption: (this is a common assumption) Training and test examples are iid (so trying and test examples are drawn from some common distribution). So what's a common asumption across almost all of machien learnign is that there's some hidden distribution that represents maybe the distribution over cat pictures. And then training and test examples are all drawn from the same distribution. Both kinds of these attacks kind of violate this assumption. Like the idea is like we're not taking a natural image or a natural training example; instead the attacker is kind of doing something on purpose in order to change the training example or the test image so that it's not actually from this distribution anymore, so this assumption kind of no longer holds, and then alot of the stuff that we kind of expect to work in machine learning doesn't work anymore.

Robert question: Justin, what is iid?

Justin answer: Let's expand on this. So assumption: training and test examples are drawn independently from some common distribution. This distribution is not known. (It's like this distirubjton over all cat pictures, so it's not mathematically specified) but the main assumption is that these training and test examples are coming from the same source (there drawing independently and identically from some distribution). Soo iid means independently and identically distributed.

Shashank question: So intuitiviely if a model that is trained to perfrom well in a domain shift senario, so would it automatically be more resistant to adversarial attacks?

Justin answer: So that a good question, um, I don't know if people have studied this before but even typically models that (I mean this would be good to investigate empirically also, like I don't actually know) but even models that are trained to resist domain shifts where maybe we don't assume there's a common distribution but maybe that distribution varies over time, there's usually some assuption about his this change is happening, it's not like an arbitrary change (it's not totally worst case) because if it's totally worst case is generally very difficult to do anything. So people will often assume that the shifts happen very slowly maybe, or there arn't too many shifts, like there might be one distribution up until some point and it will change to a different distribution just once. But generally these adversarial examples and data poisoning attacks kind of break these assumptions. Ok, the attackers not trying to behave in any very regular way but they're changing these examples. It's hard to model, they're kind of taking kind of a worst case.

Robert: So in short I guess that examples the attackers are pulling out are so low probability to occur in the distribution that so even models that are trained to be resistant to domain shift (we;; I don't know what a domain shift is) but is dosent't matter because the examples the attacker is pulling out are really low probability.

Justin answer: Uh, its a little bit hard to say. So right, all of this stuff right here (idd). (Like there's some distribution, but like a distribution not really a real thing, its just like a mathematical model for how these things are generated) so like it's hard to say that the things the attacker are pulling out are low probability. It's not clear what that means. Uh, they are certainly unnatural. Uh, they look like natural images but they have been perturbed in some way. And also all of these machine learning that we're mosly working with like these neural network based things, they don't actually have any gaurantees, so the fact that attackers are able to attack these models is not violate any mathematical theorm. Its just that this is kind of an assumption (iid) but is not like a really like a mathematical assumption, its more like a, it's like a philosophical assumption about how the world works or how, or like, its an assumption about when these models are believed to perform well in the real world. It is some assumption kind of like, uh, porgrams are kind of like turing machines, it's a philosophical assumption. But this is like in the real world most programs are like turing machines, they're not more powerful, uh, but in the real world these examples don't have to be coming from the same distribution. This assumption can be broken, by some adversary. It can be messed with. But its a challenge.

8 First paper overview

Okay, so today's adversarial examples day so we're going to talk about these three papers, um, right, so, okay. The first paper we'll talk about is the first one, it's called "Intriguing Properties of Neural Networks." This is a bunch of author but it appeared at ICLR 2014. This is the International Conference of Learning Representation so this is kind of, I think these day is a lot of deep learning and neural network research is happending here, but I guess a long time ago it was more about feature selection and how do you make exisiting machine learning systems perform better by selecting features better, but now it's mostly on neural networks.

Um and, right, so there are like two main observations in this paper. Um, the first one acutally, they label this first one, so usutally in these papers you lead with the contributions that are more interesting and the later ones are kind of less interesting, but in for this case like the second observation is the one that had kind of more impact. So the first one, which I'll kind of only briefly comment on. Uh, it's a little bit, maybe it's a little bit hard to understand. 1.) Their first contribution esentially says that there's no difference between looking at

individual neurons and looking at random combinations of neurons. So I'll try to explain this a little bit more. [justin writes it down] So, I'll talk about this in a second. Uh, the second main contribution which is the one that this paper is like really well known for is that um 2.) There are these things called adversarial examples. Which means that given some image that's classified as some class , uh, almost always I can find some very small perturbtion of the image that will lead it to be classified as something else. (Usually can change image to different class by adding a small perturbation). So this one here is what's called adversarial examples. So the idea is that if I have some picture of a cat and my image classifier says that it is a cat, most of the time there exists some very small perturbation image that will lead to the classifier classifing this thing as something else. (An ostrich or a or something). So the idea is that I can take an image that looks like a cat and add a tiny bit of imperceptible pertubation that will change the class. So that kind of means that neural networks...this prediction part is not robust because very small changes to the input can lead to changes to the class. So even small changes that a human can not detect, like if I showed you the perturbed image you would still think its a cat, uh, it looks like exactly the same picture, uh, but the prediction will give a completly differnt prediction so that's interesting or bad for various reasons.

9 Feature layer may not be detecting features after all (at least not in a separable way)

So, alright, so did you take a look at the two contributions? I kind of want to talk about the first one only very little, uh, because I think it's a little bit less interesting than the second one. This first on roughly says that..... So recall that the neural network looks something like this; so we have some inputs and a bunch of hidden layers and soem outputs. Ok, so this is the output layer, and this is the input. Um, so for a while it was believed that, um, this second to last layer here, which is sometimes often called the feature layer; it was beleived that each one of the units here was capturing some property of the original input, something that you could recognize. Right so if you remember like the older style machine learning where you would kind of craft these features by hand, so say, I want to have a classifier for cats, so I'm going to make a bunch of features, so these feature might be like number of legs, or you know like um, the orientation is like more horizontal or more vertical, uh, maybe number of eyes or something like this; I would craft these features by hand and label them. So given a picture I would fill in these features, and then I would use these features to try to predict something about the image, you know, whether its a cat or not. Right so, when neural networks where first kind of explored and kidn of took off; people beleived that maybe, they suspected they worked very well because they were somehow learning the features themselves. Uh, right, so the features were not hard coded anymore, like I don't have any features anymore, these might be just pixels in an image [refering to input layer]. I'm not trying to identify like

how many legs the image has, I just want to feed in the raw bits, or the pixels. Um, and the beleivf was that maybe by theis training process, each of the units in the last layer would encode some semantic information; right so maybe one of these way going to be like number of legs, maybe it would somehow learn this thing by itself. It would learn which are the important features of the image and then it will use these important features to predict whether its a cat or not.

The first contribution is trying to kind of go against this and says that is not how it works. So how to people try to test this thing. So they look at one unit right here [draws box] and they try to find all of the images that try to maximally activate this unit; they try to make the output of this unit as big as possible. They search through of different images and try to find the colelction of images that makes the output of this unit as big as possible. And they will look at the images asn say, oh these are all images where like there's a round part in the bottom left. Right, so they're like, oh, this neuron is probably encoding for a round part in the bottom left. We can kind of guess theis, and we can kind of recverse engineer maybe or try to explain what each of these units was doing by looking at all the pictures that try to maximally activate each unit. This is like round upper left, this is like strigt diagonal line maybe, this is like curve on upper right [referring to neurons in the feature layer] or somthing like this by looking at which of the iamges try to activate that neuron.

Un, so the first one, these author showed that you can also do this with just like a random combination of these units. So I can take a random combiantion and try to maximize the total output from this random combination, and looking at the images that maximize this random combination, they still look very simular. Right so somehow, so like taking a random comiantion there's still alot of simularity in the inputs that kind of maximize this combination, which seems to mean that its not the individal features are responsible for certain things but even looking at a random comiantion, it still captures some information about the input, and that was kind of suprising. Okay, so that individual units look kind of the same as random combinations in the training network.

Student question: So what does random combination mean?

Justin answer: Formalism on formula to maximize.

Robert question: How do they evaulate simularity.

Jusin answer: Just eyeballing (for example on third page). They kind of just eyeballed it (so like are they the same number) in generally its actually very difficult to identify simularity between two different images.

So this in kind of the like the smaller point of the paper. The bigger one is this adversarial examples point, so this is the one I will spend most of the time on.

10 Adversarial Examples

2.) Small perturbations to the input

Again this is kind of surpising at the time because image classification tasks, when people started using modern neural networks and they seemed to work very well, uh, people were very impressed and they thought that they could solve image classification. This paper pointed out that there are very significant blind spots in these systems.

The setup is like I have some image x, and I have some neural network and I predict a class. And, so these people wondered, can we perturb the image a little bit to change the class. Want to find some vector r such that. A.) want r to be kind of small, I don't want to change the image too much. (so l2 norm of r is small) but lets just say I want to find a small perturbation where small means its a small norm B.) and want f_w(x+r) equal to l

So if the original image was classified as a cat and I want it to be classified as a frog, can I find some small perturbation that will make this happen, that I can change the predition. Now if the neural network was really learning some robust structure of the image that it was kind of understanding or something, you now, like getting the structure of the image, you would expect this is probably not possible, right because this perturbation can be very very small such that you can't even see it with the eye, so any human would recognize the image as being the same thing (same class), but this paper found that it is actually possible to do this. You can set this up as an optimization problem which I'll describe shortly and you can solve the optimization problem to find some perturbation. They tried it form many different target classes, and it seems like almost always you can do this. And this is an intriguing property.

Descirbe optimiztion problem:

This is a constried problem but they change it to an unconstrined problem by using this kind of trick.

Discussed pictures in paper.

Robert: so C is some value with choose from experience, its some value we like?

Justin: Right, c is some coefficient. It's a fixed constant, you can pick anything you want for it. So I guess in optimizaiton this is often call a Lagrangian where you are taking these constraints and putting them... these [refering to the top] are hard constraints, so this optimization problem, any solution must satisfy these constraints. But sometime optimization problems of this type are hard to solve. One way to make them easier is to say I'm going to make these hard constraints into soft constraints. So like, in these kind of version of the problem [with soft constraints] I can violate this constraint, this one here, if I incur some penalty, so if the predicted class is not equal to l, I will have some penalty, which will make my objective worse, and this C will conrol the tradeoff between the actual objective and the constraints. This is a common trick, and its not gauranteed to be the same solution here. This one is just easier to solve and hopefully will give you something close to what this solution is. So you can violate the constraints and you have to pay a little for that.

Shashank: Is is possible, instead of fixed a particular target label, to pose the optimization problem such that it could predict any label but just not the original label.

Justin: Right, so yes, you can definitiely do that too. So this is like a targeted attack where you are trying to make it pick a specific label, but you might be just happy with making it wrong, like uh you just want to make it predict any other label besides the actual one. Uh, I think that's what the second paper does, but this is rather a small change, its definitely possible. Their almost the same.

11 Transferbility

I also want to talk about one more thing in this paper, they were the first to point this out, its another interesting property. So, this is the setup of adversarial attacks, adversarial examples [pointing at optimization formulation] this if how you find them, you solve this optimization problem.

Another interesting property: 2' Transferability So the idea here is that people found these adversarial examples. They thought it is very strange, you can perturb and imge and change a class.

I'm going to train a different kind of model, maybe not a cnn, maybe a different wiring, an i want to see if the adversarial examples still work against the different model. Might change the hyperparameters. Maybe change on disjoint set of training data.

For example two sgd model, one with twice as many steps, see if adversarial examples still work.

And this paper found that in all these cases these adversarial examples do transfer. So if I can find some adversarial examples against one model, probably its also going to work against simular models. Ok, and this is a very mysterious feature, uh, they later went on in the second paper to try to explain more about why this is happening, but its very unexpected right. I train two different model, very differnt models with differnt training procedure and it kind of suggests that these adversarial examples are not very specific to any one more, they're more general somehow.

Question: not audible [what do you mean by simular]

Justin: Right set find adversarial exmaples aginst one model and test against other model. So maybe one model is a linear classifier and the other is a cnn based classifier or something like this. Or maybe the architecture of the neural network is ver different between these two models. So if the adversarial example was very specific to the model it was trained on (it was found against) I would expect that if you tried to use this example against other models it would not fool the other models, but this paper found that it usually does fool other models.

Unaudible qestion: []

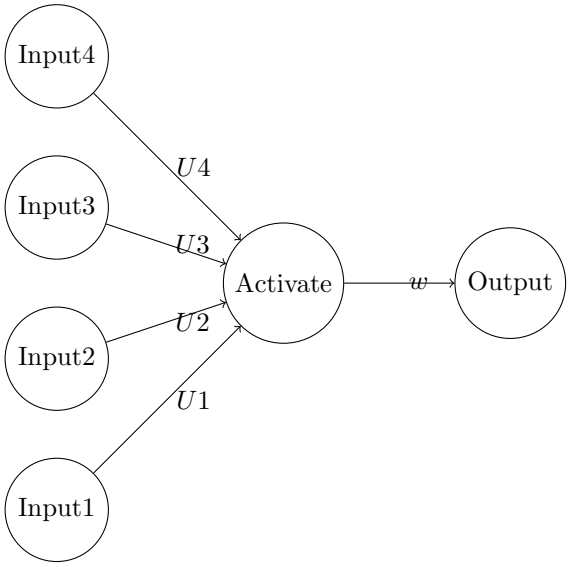
Justin: Uh, so this image is the same so most of the time, it will transfered. Thre is acutally an interesting picture and graph. If you look at the page 7, they have a table where they kind of trained adversarial examples against specific models and test them on similar models and non-similar models. They showed that if I trained some examples on my model then it will fool on 100% of the time that I can always make predict wrong for the thing. But when we take it in and tried on the other model, they also noticed that maybe not 100% of the time, but a substantial fraction of the time, the other model will also predict incorrectly. So then the adversarial examples can attack many kinds of models, even if it is just trained on some specific models.

12 Explaining and Harnessing Adversarial Examples

12.1 claim1

Adversarial examples come from linearity in Neural Network.

12.1.1



There are a lot of adversarial eamples

12.2 claim2

Faster algorithm to find adversarial examples

13 setup:

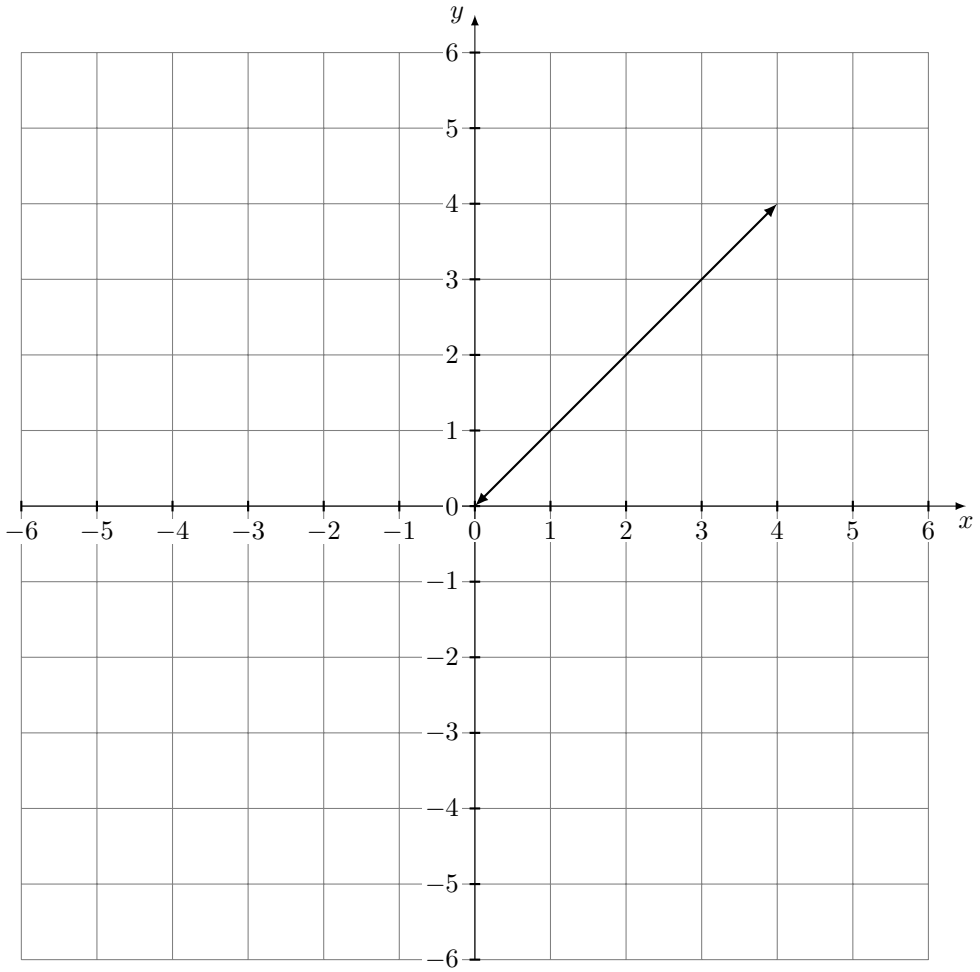
$$x \in Ximage, \tag{8}$$

fw is a model. perturbation r s.t.

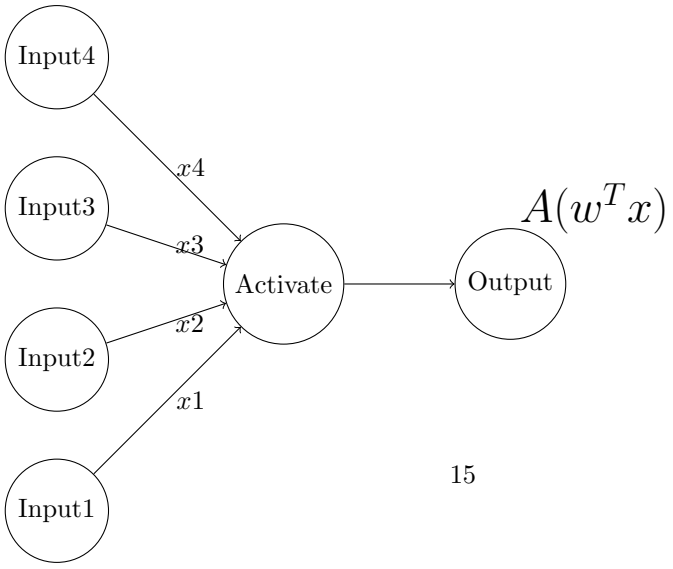
$$\|r\|_\infty \leq \epsilon \tag{9}$$

want:

$$f_w(x) \neq f_w(x+r). \tag{10}$$



13.1 Small perturbation → Large changes



$$A(w^T(x+r)) = A(w^T x + w^T r) \approx A(w^T x) + A(w^T r) \tag{11}$$

13.2 Fast gradient sign method (FGSM)

13.2.1 CostFunction:

$$J(w, x, y)(maybe|f_w(x) - y|) \tag{12}$$

13.2.2 Targeted attack

make x+r $\rightarrow$ y

$$f_w(x_*) = y_* \tag{13}$$

want to change prediction

13.2.3 set

$$r = \epsilon * sgn(\nabla_x * J(w, x, y_*)) * (X_*) \tag{14}$$

$$sgn(z) = \begin{cases} +1 & \text{if } z > 0 \\ 0 & \text{if } z = 0 \\ \text{-1} & \text{if } z < 0 \end{cases}$$

14 Questions and Answers

Question 1 Student: can we work and seen if adversarial examples if there is any gift in search change if we use some technologies we discussed last Friday with like noisy SGD and PATE, those kind of things?

Answer 1 Justin: Right. So, that is a question. You could think about differential privacy as making a some part of the distinct more robustic. It means that for differential privacy, it garantee the training set. For example, if I change the training example by 1, those two sets will not be too different. So, use differential privacy and the technique we discussed last time to protect the learning algorithm, but today we are protecting the inference and prediction part. It is possible that the model you learn are from the training set but it is not that robustic. We change the picture a little bit and we do change the class, so that part is not covered in differential privacy. But there is some work try to eject noise in the inference and prediction part, to make this part more robustic. Currently, there is no good answer yet. All neural network based model will have adversarial examples. There is no way that we currently know to know the example. They are have them and they are easy to find. The second paper have a lot of them. There are many many adversarial examples. It is not just a isolated phenomenon.

Question 2 Student: Does this perturbation have any pattern so that we can figure them out?

Answer 2 Justin: It does not seem like they do,so today we are talking about attacks first for this week and next week we are going to talk about defenses. As you can imagine, defence is much harder than attack. Attack you only need to find one weakness, but defence you have to defend the whole collection of attack, but the short answer is that it is not easy to figure this out. There is no acceptable pattern and people have tried different things, but there is no good solution yet.

Question 3 Student: Generating adversarial examples on data poisoning would require access to the parameters of the model would it require to be a white-box?

Answer 3 Justin: Right, that's right.There is also work on the black-box attacks,where you can make kind of queries.This one is white-box, right.The way they do this kind of stuff like they have to be able to have more structure to do. THis is the pertubation thing, it will take some weights and inputs to predict the class. So the parameter must to be known to evaluate the function. But there are also black-box attacks, black-box attacks are also possible. They have various of these attacks, they have single pixel attacks, one pixel and you can change the class. This is also possible, you don't have to change too much noise.

Question 4 Student: Based on the strong transferbility, can we just adversarial exapmles from the white-box and fly to the black-box?

Answer 4 Justin: Yes, you could expect that you could do this too. That would be one way to doing this. I think if you only get the black-box access you can still get query that you can learn something about the optimization function, even if you don't know the whole thing. You kind of just need to know how to perturbe the image and that we will see in the next article. If I make this input bigger, it is going to change something and you can try to query to find each pixel at a time, and find one to change.

Question 5 Student: So you can construct adcvesarial examples by using black-box?

Answer 5 Justin: Yes, you can.

Question 6 Student: The odd that we defind the model is for one specific example, of an image?

Answer 6 Justin: Right, so in this optimization problem. the w is the model, x is the image and the X is for the clear image. The odd is for every image.

Question 7 Student: For the two models, so I realize that they are very different, but do they have some similarities? For instance, their accuracy. So if one of the models can predict the cat at 90%, should we have the transferbility? Should we have another model that can predict the cat at 90%?

Answer 7 Justin: Right, they both have to be equally accurate for the same task.

Question 8 Student: Do they have a rigorous proof of the reason of the transferbility?

Answer 8 Justin: No, they don't have any expalination of the reason.

Question 9 Student: Intuitively, should there be any commonalities for the transferbility?

Answer 9 Justin: Intuitively,yes, but pratically no. They have to be the same task, but the archetecture could be very different. It is a very intruiging property.