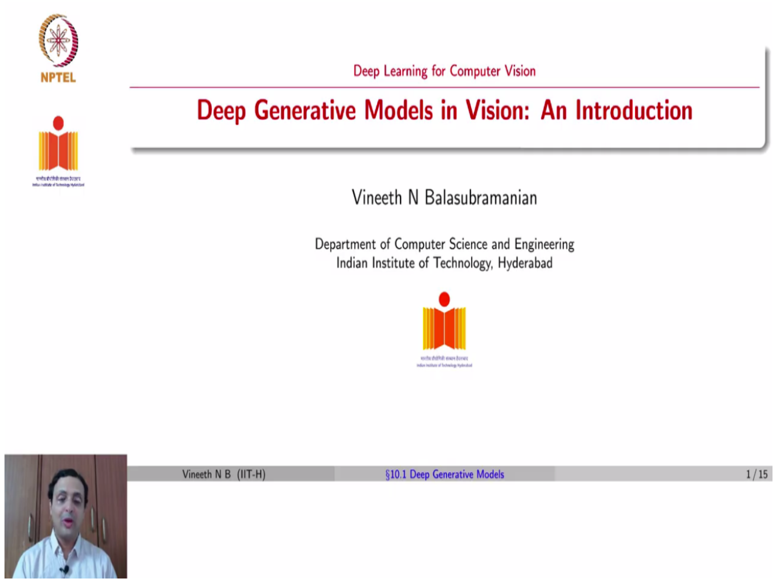
Deep Learning for Computer Vision
Professor. Vineeth N Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Lecture No. 61
Deep Generative Models: An Introduction

(Refer Slide Time: 00:14)



Deep Learning for Computer Vision

Deep Generative Models in Vision: An Introduction

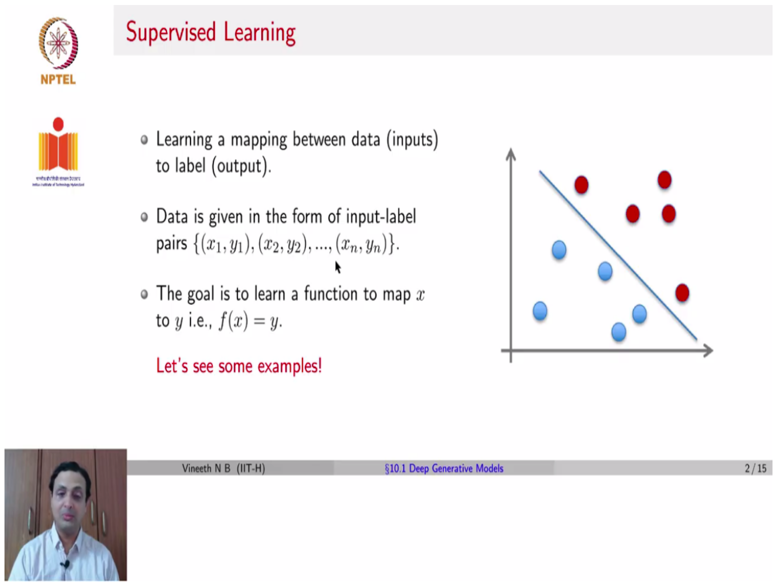
Vineeth N Balasubramanian

Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Vineeth N B (IIT-H) §10.1 Deep Generative Models 1 / 15

From Attention methods in Deep Learning Models and Transformers, we will now move to 2 weeks of a contemporary topic in Deep Learning, Deep Generative Models. Deep generative models include topics such as GANs, Generative Adversarial Networks, and Variational Auto Encoders, VAEs, which have been extremely successful over the last few years.

(Refer Slide Time: 00:49)



Supervised Learning

- Learning a mapping between data (inputs) to label (output).
- Data is given in the form of input-label pairs $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$.
- The goal is to learn a function to map x to y i.e., $f(x) = y$.

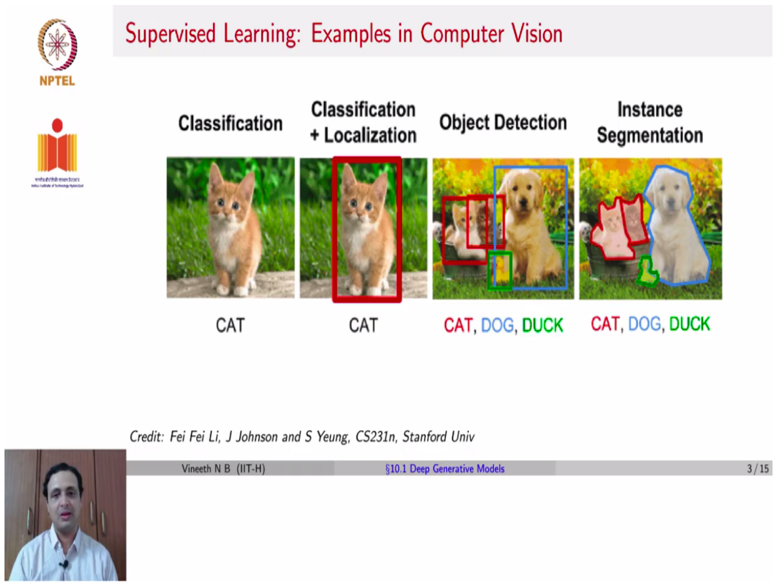
Let's see some examples!

Vineeth N B (IIT-H) §10.1 Deep Generative Models 2 / 15

Before we talk about the overall perspective of Deep Generative Models, let us talk about the landscape of machine learning itself and try to understand where deep generative models fit into the scheme of Machine Learning methods. If you considered Supervised Learning, supervised learning is about learning a mapping between inputs and outputs, that is, data and class labels. Your data is typically given in the form of input-label pairs, such as (x_1, y_1), (x_2, y_2), and so on till (x_n, y_n).

And the goal of a machine learning algorithm is to learn a model that can map x to y. So, it aims to learn a function f, which, when applied on x, gives us y, where we would like y to be as close to the original ground truth, y, that is provided. So, what are the supervised learning settings or problems that we have seen so far?

(Refer Slide Time: 2:00)



Supervised Learning: Examples in Computer Vision

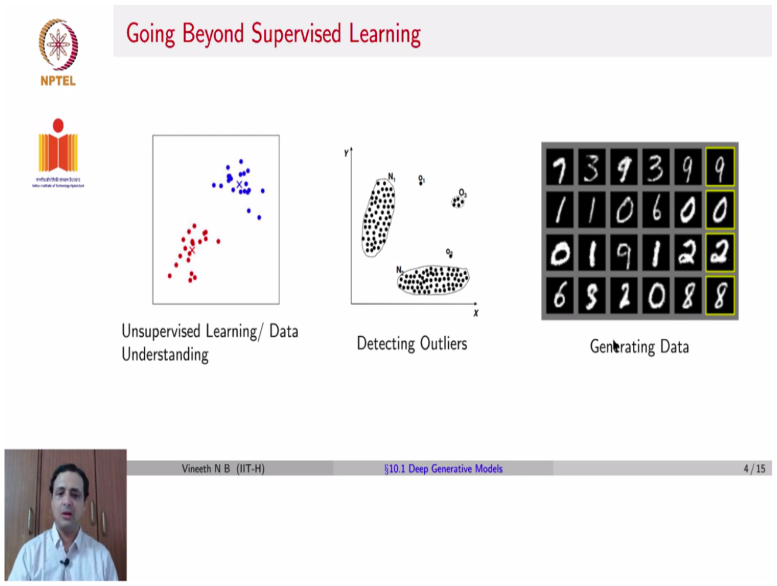
Credit: Fei Fei Li, J Johnson and S Yeung, CS231n, Stanford Univ

Vineeth N B (IIT-H) §10.1 Deep Generative Models 3 / 15

In computer vision, we have seen problems including Classification, Localization with Classification, Detection, Segmentation, and so on. These are supervised learning problems where there was an expected output for a given data point. Our deep neural network was trying to learn the function that takes us from input to output. The nature of the output varies with different kinds of tasks. For detection, the output looks different from the output for Classification.

For Segmentation, the output looks different from that of Detection and Classification. That resulted in different kinds of architectures, loss functions, and so on.

(Refer Slide Time: 02:57)

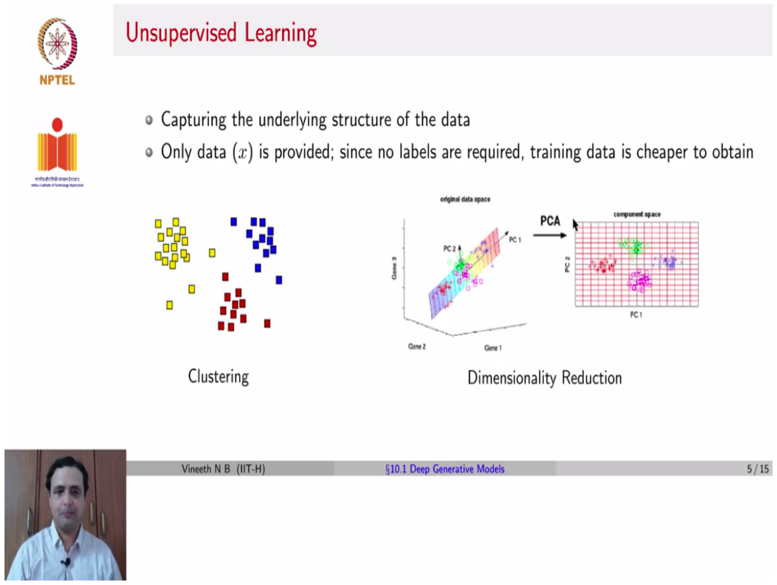


Going Beyond Supervised Learning

Vineeth N B (IIT-H) §10.1 Deep Generative Models 4 / 15

So, what are we talking about here, to go beyond Supervised Learning. So, if you think about paradigms beyond supervised learning, we talk about Unsupervised Learning, where the broad goal is about just understanding data. Another setting is about detecting outliers. That is another setting where one can use unsupervised learning. Finally, this week's lectures focus on generating data from past data, with no labels involved as it is, but just generating data.

(Refer Slide Time: 03:39)



Unsupervised Learning

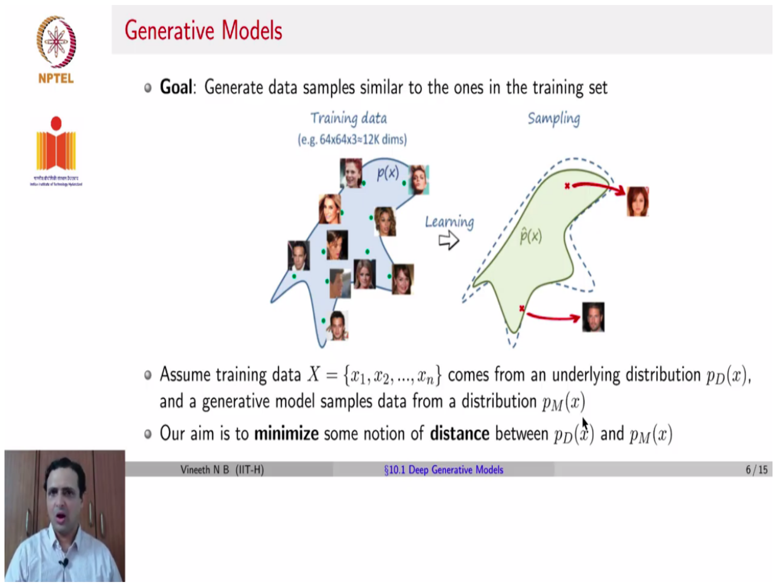
- Capturing the underlying structure of the data
- Only data (x) is provided; since no labels are required, training data is cheaper to obtain

Vineeth N B (IIT-H) §10.1 Deep Generative Models 5 / 15

So, suppose you considered unsupervised learning, which is perhaps the most popular form of going beyond supervised learning. The main idea is to capture the underlying structure of the data. You are given only the data; there are no labels. Since no labels are required, often training data is cheap to obtain. The most difficult part often can be to get annotations or labels for data, which can be challenging tasks depending on domains.

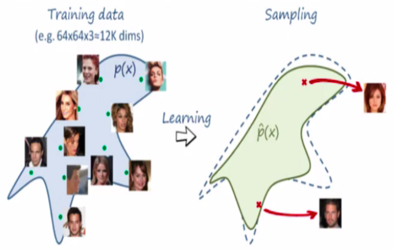
For example, in healthcare or genomics, or bioinformatics, getting labels for a data point may require experts to participate and may be a very costly affair both time-wise and money-wise. Unsupervised learning is generally categorized into clustering methods, such as K-means, DBSCAN, and dimensionality reduction methods, such as Principal Component Analysis, ISO map, and many other such methods.

(Refer Slide Time: 04:53)



Generative Models

- **Goal**: Generate data samples similar to the ones in the training set



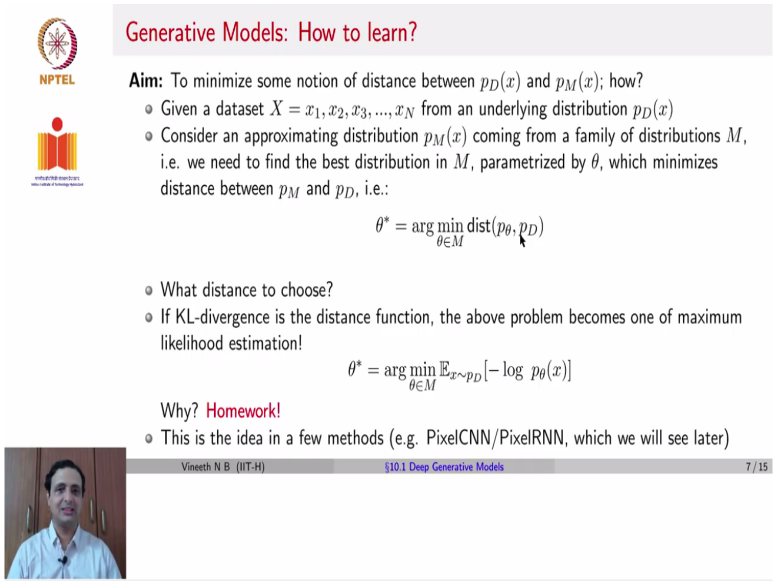
- Assume training data $X = \{x_1, x_2, ..., x_n\}$ comes from an underlying distribution $p_D(x)$, and a generative model samples data from a distribution $p_M(x)$
- Our aim is to **minimize** some notion of **distance** between $p_D(x)$ and $p_M(x)$

Vineeth N B (IIT-H) §10.1 Deep Generative Models 6 / 15

The focus of this week's lectures, as I just mentioned, is Generative Models, which is another variant of unsupervised learning. Given a set of data points, our objective is to generate more data samples, similar to those in the training set. What does that mean? Given a set of data points, as you see here, those are points that lie on some manifold. Remember, that manifold corresponds to a locally Euclidean surface, which you can look at as the intrinsic dimensionality of your data.

So, suppose data lies on a certain manifold, which corresponds to a distribution. In that case, we learn a model using some method, which can then sample from a distribution that the model has learned and generate new data points. What does this mean mathematically? If you had training data, x_1 to x_n, which comes from some underlying distribution, let us say p_D. We would like a generative model to sample data from a distribution $p_M(x)$ to minimize some notion of distance between p_D and p_M. We would like to learn p_M so that p_M becomes close to p_D in some notion of a distance.

(Refer Slide Time: 06:40)



Generative Models: How to learn?

Aim: To minimize some notion of distance between $p_D(x)$ and $p_M(x)$; how?

- Given a dataset $X = x_1, x_2, x_3, ..., x_N$ from an underlying distribution $p_D(x)$
- Consider an approximating distribution $p_M(x)$ coming from a family of distributions M, i.e. we need to find the best distribution in M, parametrized by θ, which minimizes distance between p_M and p_D, i.e.:

$$\theta^* = \arg\min_{\theta \in M} \text{dist}(p_\theta, p_D)$$

- What distance to choose?
- If KL-divergence is the distance function, the above problem becomes one of maximum likelihood estimation!

$$\theta^* = \arg\min_{\theta \in M} \mathbb{E}_{x \sim p_D}[-\log\ p_\theta(x)]$$

Why? Homework!
- This is the idea in a few methods (e.g. PixelCNN/PixelRNN, which we will see later)

Vineeth N B (IIT-H) §10.1 Deep Generative Models 7 / 15

How do you do this? Can we elaborate on this? Let us try to go a bit further. So, we said we would like to minimize some distance between p_D and p_M. Given a dataset from an underlying distribution, p_D, consider an approximating distribution p_M coming from a family of distributions M. Our objective is to find that parameterization in M or that distribution in M given by θ, the parameters of that distribution, which minimizes some notion of distance between p_M and p_M.

So, the objective function will be θ^* that is, our solution is given by argmin theta over M, some distance function of p_θ, and p_D. What distance function do we use? We choose KL divergence as the distance function, which is a loose distance function between two probability distributions. It does not satisfy all properties of a distance metric, but it is commonly used to measure the distance between two probability distributions.

If we replace this distance function with KL-divergence, it can be shown that the above problem becomes one of maximum likelihood estimation. θ^* becomes argmin over θ belonging to M, expectation with data points drawn from the distribution p_D , $-\ log(p_\theta(x))$. This is maximum likelihood estimation or minimization of negative log-likelihood. Why is this so? We are going to leave that as homework. It is fairly trivial to show that if you replace distance with KL divergence, you will automatically lead in two steps to negative log-likelihood. Please do work it out. We will discuss it in the next class.

This idea of learning p_M or p_θ in this case, using maximum likelihood estimation, is one way of learning generative models or deep generative models, in our case. A set of pixel CNN methods and pixel RNN methods explicitly use this objective to learn deep generative models. And we will see these methods a little later this week.

(Refer Slide Time: 09:32)

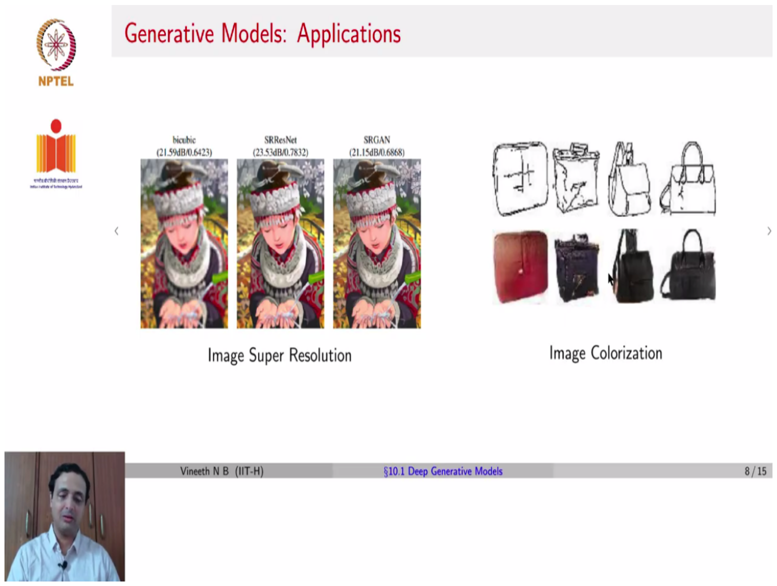

Generative Models: Applications

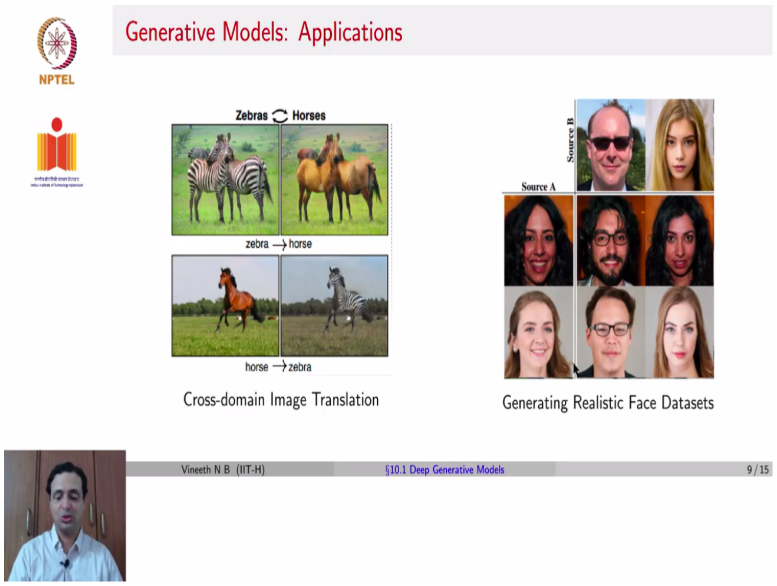
Image Super Resolution

Image Colorization

Vineeth N B (IIT-H) §10.1 Deep Generative Models 8 / 15

What are the applications? Why should we learn Deep Generative models? Deep generative models can be used for image super-resolution. Given an input image of low resolution, you can use generative adversarial networks or any other deep generative model to learn a super-resolved version or a high-resolution version of the original image. You could use such models to colourize images, go from sketches to color images, or go from black and white to color images.

(Refer Slide Time: 10:10)



Generative Models: Applications

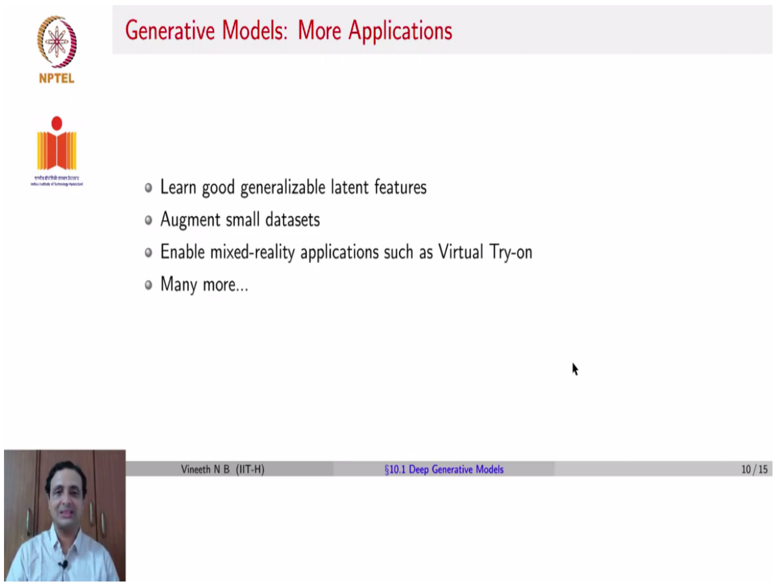
Cross-domain Image Translation

Generating Realistic Face Datasets

Vineeth N B (IIT-H) §10.1 Deep Generative Models 9 / 15

You could use this for cross-domain image translation to go from one domain to the other. In this case, we talk about taking a zebra picture and going to a picture of a horse. But you can also talk about going from a picture to a cartoon, and so on. Finally, you can also talk about generating realistic face data sets for performing face recognition, and so on.

(Refer Slide Time: 10:44)



Generative Models: More Applications

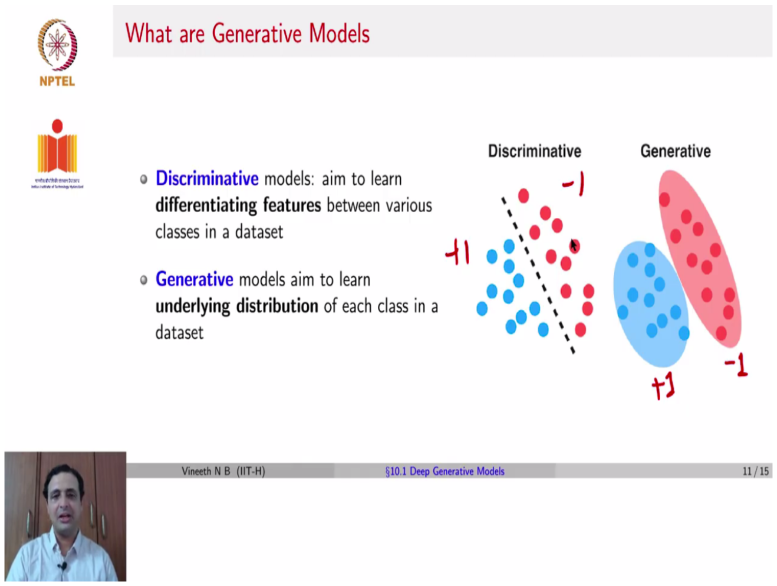
- Learn good generalizable latent features
- Augment small datasets
- Enable mixed-reality applications such as Virtual Try-on
- Many more...

Vineeth N B (IIT-H) §10.1 Deep Generative Models 10 / 15

There are many more applications; the goal can also be to learn good, generalizable latent features by learning to generate new images. The model also allows you to learn latent representations, by which we mean the hidden representations and certain layers of that model. Which can then be used for other downstream tasks such as, say, classification. You can also use this generation of new data to augment small datasets. You can then get larger data sets, which you can train for downstream tasks such as classification, again.

You can use such models for enabling mixed reality applications, such as Virtual Try-on. You can imagine going to any shopping website that lets us try on a shirt or a trouser before you purchase them. You could now take your image and see how you would look in that particular shirt, or how you would look with a specific kind of glasses, or sunglasses, for that matter. Many more applications that we will see this week, as well as in next week's lectures.

(Refer Slide Time: 12:05)



What are Generative Models

- **Discriminative** models: aim to learn **differentiating features** between various classes in a dataset
- **Generative** models aim to learn **underlying distribution** of each class in a dataset

Vineeth N B (IIT-H) §10.1 Deep Generative Models 11 / 15

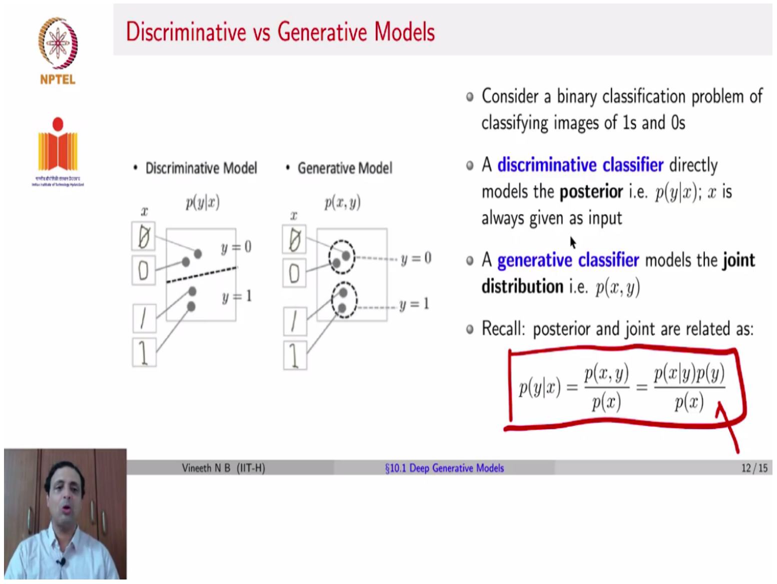
Before we move on with methods for generative models, what do generative models mean? Let us try to understand this from a conceptual perspective. Generative models can be distinguished from their complement, which is known as Discriminative models. In machine learning, discriminative models aim to learn differentiating features between different classes in a data set. In contrast, generator models aim to learn the underlying distribution of each class in a dataset.

Note that we are now trying to use an example from supervised learning to understand generative models better. All the examples that we saw so far of generative models as unsupervised learning. Generative models can also be used in supervised settings, where you can specify the class label, and the model will generate data from that specific class. So, for example, if you are generating a face image, you can say which person you want. The model can generate the face image for that person, which would correlate to a more supervised setting than an unsupervised setting.

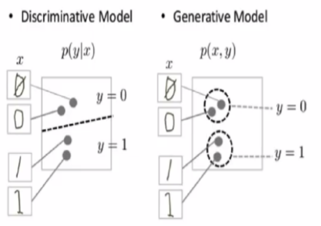
So, in supervised learning in Discriminative models, the goal is to learn a discriminator that separates two classes. An example would be support vector machines, which is a maximum margin classifier. On the other hand, in a generative model, the goal is to learn the parametrizations of distributions for each class. In a Discriminative model, one would finally do inference by asking if a test data point belongs to one side of a hyperplane or the other side. If it fell on one side of the hyperplane, you would say it belongs to class $+1$, and if it belongs to the other side of the hyperplane, you would say it belongs to class -1.

Now, in a Generative model, we try to find out that given a data point, what is the probability that the distribution of class $+1$ generated this data point? We will try to find out for the same data point what is the probability that the distribution of class -1 generated the same data point. You can see here that the entire approach to assigning a class label for the new point is generative and tries to understand how likely a certain distribution would have had a probability of generating a particular data point. That is the difference between discriminative and generative models more broadly in machine learning.

(Refer Slide Time: 15:25)



Discriminative vs Generative Models



- Consider a binary classification problem of classifying images of 1s and 0s
- A **discriminative classifier** directly models the **posterior** i.e. $p(y|x)$; x is always given as input
- A **generative classifier** models the **joint distribution** i.e. $p(x,y)$
- Recall: posterior and joint are related as:

$$\boxed{p(y|x) = \frac{p(x,y)}{p(x)} = \frac{p(x|y)p(y)}{p(x)}}$$

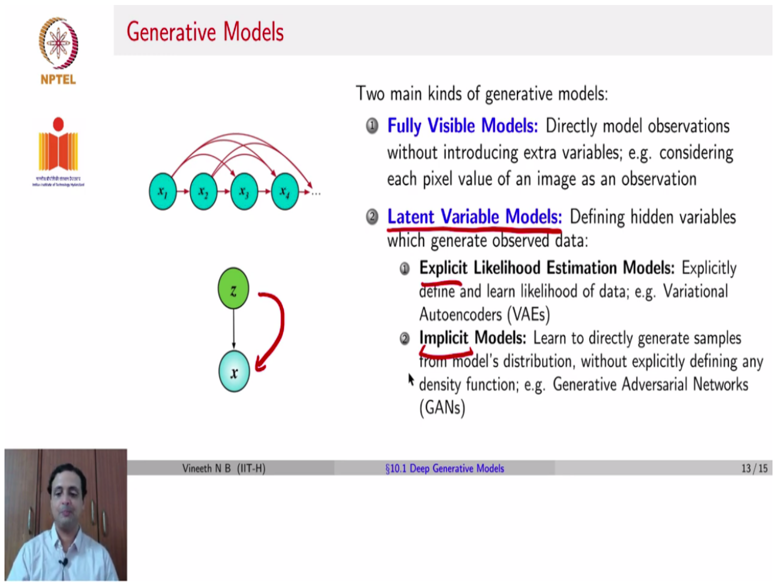
Vineeth N B (IIT-H) §10.1 Deep Generative Models 12 / 15

So, if you considered a binary classification problem of classifying, say images consisting of 1s and 0s. A discriminative classifier tries to model the posterior $p(y \mid x)$, a conditional distribution, where x is given as input. Your model has to give a particular probability for y given x. So, discriminative classifiers model this conditional distribution called the posterior. On the other hand, generative models model the joint distribution $p(x, y)$.

So, how would you use a generative model for assigning a class label? You would take a test data point and use this joint distribution to determine if this data point was assigned a class label $+\ 1$. What is the probability of having generated such a data point? If this data point was allotted a class label $-\ 1$, what is the probability of generating such a data point? And that would give you a way of getting a class label from this joint distribution.

Remember, the posterior, conditional distribution, and joint distribution is related to your probability basics. Where $p(y \mid x)$ is $p(x, y)$, your joint, by $p(x)$, which relates to the Bayes theorem, which states $p(y \mid x)$ is equal to $p(x \mid y)$ into $p(y)$ by $p(x)$. Some of these understandings are something that we will use when we talk about different deep generative models.

(Refer Slide Time: 17:27)



Generative Models

Two main kinds of generative models:

1. **Fully Visible Models:** Directly model observations without introducing extra variables; e.g. considering each pixel value of an image as an observation
2. **Latent Variable Models:** Defining hidden variables which generate observed data:
   1. **Explicit Likelihood Estimation Models:** Explicitly define and learn likelihood of data; e.g. Variational Autoencoders (VAEs)
   2. **Implicit Models:** Learn to directly generate samples from model's distribution, without explicitly defining any density function; e.g. Generative Adversarial Networks (GANs)

Vineeth N B (IIT-H) §10.1 Deep Generative Models 13 / 15

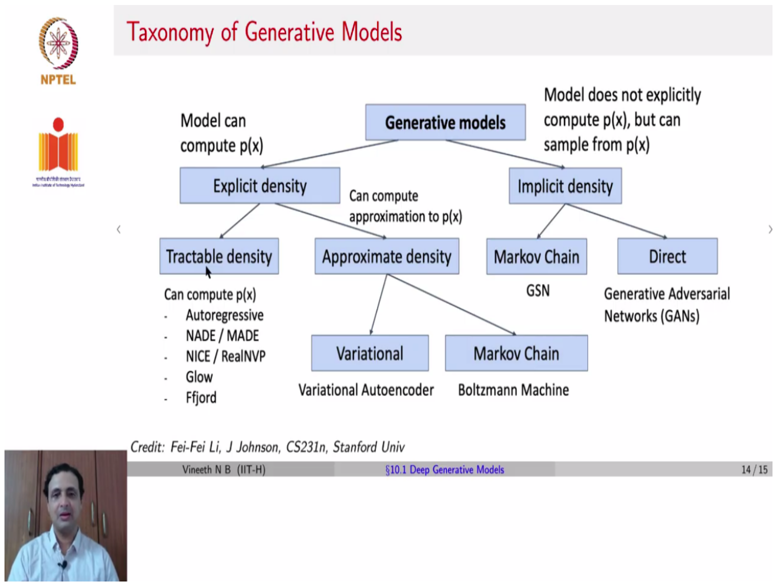
Existing deep generative models can be broadly divided into two kinds: Fully Visible models, which directly tries to model the observations. Here, the observations are x, and the model tries to capture these observations themselves without introducing any extra random variables. So, the only random variables considered are the excess of the data that you have. So, for example, you could be considering each pixel value of an image as an observation. Based on that, you will learn a generative model, which can generate new images.

The other category of methods is latent variable models. You introduce other kinds of random variables known as latent variables, or hidden variables, which we assume generate the data, which we try to learn as part of generating the latent part of learning the generative model. Within these kinds of models, latent variable models, there are very broadly speaking two kinds explicit and implicit models.

In explicit models, we try to explicitly define and learn a certain likelihood of data by defining a certain distribution. We may, for example, define a Gaussian on data or a mixture of Gaussians on data and then try to learn the means and covariances of those Gaussians as part of the learning procedure. On the other hand, in implicit models, there is no distribution assumed. No probability density function parameterization is assumed.

Given a set of data points, we do have some latent variables. But we would like to learn the latent variables so that we can generate data points that look similar to the data points that we have in our training data. We do not impose any specific distributional form on data or the model’s distribution in this particular case.

(Refer Slide Time: 20:11)



More generally speaking, there is an entire taxonomy of dividing existing approaches for coming up with generative models. So, as we just said, very broadly speaking, generator models can be divided into explicit methods and implicit methods. In explicit methods, you assume a certain PDF density function and try to learn the parameterization. In an implicit function, we do not assume any such distribution. Still, we just want to learn a certain density function, which can generate data similar to what we have seen in our original training data.

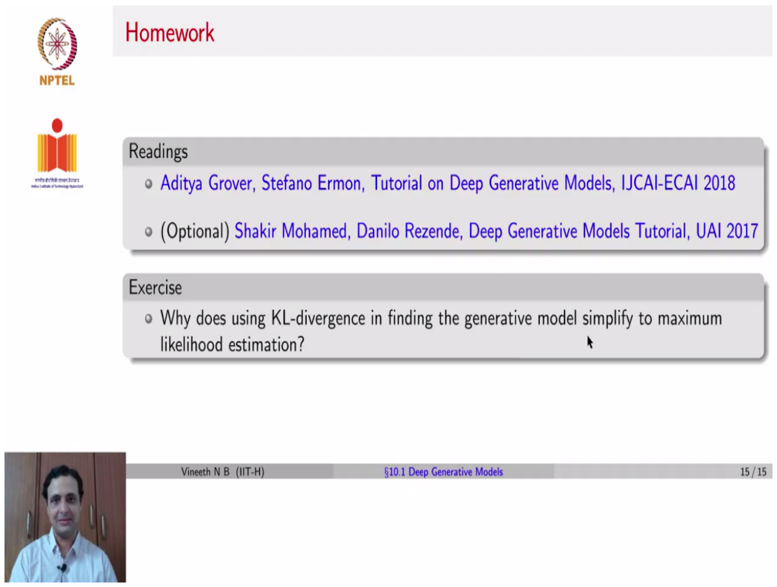
Within implicit density generative models, a popular example is GANs (Generative Adversarial networks) which tries to use the implicit density learned and generates data points. Another variant of implicit generator models is Markov chain based models such as generative stochastic networks. These are not as popular as GANs these days. But that is where they fit in into the overall landscape of generative models.

On the other hand, within explicit density generative models, you again have two kinds, one where the density is tractable, and one can estimate it and move forward to generate data points. Examples of methods here include Autoregressive models, NADE, MADE NICE, GLOW Ffjord, so on so forth. We will at least see some of these in later lectures this week.

Another kind is where you do have an explicit density function, but you cannot directly compute it. Still, you can compute it through an approximation, where the most popular method is through a method known as Variational Inference, which leads to a model called Variational Autoencoder. Another approach here is also again using Markov chains to approximate this density function, which leads to a family of methods known as Boltzmann machines.

We would not cover all of these methods in the lectures but cover the most popular ones, including GANs, VAEs, and some tractable density methods.

(Refer Slide Time: 22:50)



Homework

Readings

- Aditya Grover, Stefano Ermon, Tutorial on Deep Generative Models, IJCAI-ECAI 2018
- (Optional) Shakir Mohamed, Danilo Rezende, Deep Generative Models Tutorial, UAI 2017

Exercise

- Why does using KL-divergence in finding the generative model simplify to maximum likelihood estimation?

Vineeth N B (IIT-H) §10.1 Deep Generative Models 15 / 15

For more information, you can go through this excellent "*Tutorial on Deep Generative models*" by Aditya and Stefano, delivered at IJCAI 2018. If you would like to go further, another tutorial in a conference called UAI 2017, by Shakir and Danilo. Let us revisit one question that we left behind: why does using KL-divergence in finding the generator model simplify to maximum likelihood estimation?

As I already mentioned, it is a straightforward derivation. Please do work it out and try it before we discuss it.