# Discriminative and Generative Models

Linquan Jiang & Jingzhi Xu

2021/4/2

Generative and Discriminative models are two major streams of modelling relationship between variables in data analysis. The foundmental difference is the learning task, predictive or generative, written mathematically, $p(y|x)$ or $p(x, y)$, conditional or joint distribution. Discriminative relationship can be derived from generative models, as is the hypothetical data generating mechanism pouring out synthetic data $(\tilde{x}, \tilde{y})$ from which the conditional distribution of $p(y|x)$ can be easily calculated. This little essay aims at introducing basic concepts of generative and discriminative models and their applications in different scenarios with a few mostly wrong but somehow useful simulation illustrations for comparison.

Start from the definition. A generative model describes how a dataset is generated, in terms of a **probabilistic** model. By sampling from this model, we are able to **generate new data**. (Foster, 2019). The model has to be probablistic, or it will produce the same output everytime failing to generate new data, and this means there has to be some stochastic components. By intuition this class of models are trying to imitate some underlying distribution generating some observations more often than others in the training data. By comparison, when modeling discriminative relationships, everything is about labels and mapping to labels, odyssey towards giving correct labels to new data. At first glimpse this seems like supervised learning, and indeed it is, by the nature of the modelling process: get a series of $\{(x_i, y_i)\}$,fit whatever $f(x) = \hat{E}[y|x]$, calculate loss $l(y_i, \hat{y}_i)$, and maybe repeat this process serval times, if manually doing gradient descent for models without analyitical solution for optimal parameters. To this extend, discriminative modeling attempts to estimate the probability that an observation $x$ belongs to category $y$. Generative modeling doesn't care about labeling observations, as is sort of like unsupervised learning. But still it can discriminate like everyone, by generating observations from each distinct class.

The measure of goodness differs for the two classes of models. For the discriminative model, providing accurate predictions is plausible enough. But we want more from generative models, as is qualified if:

1. It can generate samples as if drawn from the true underlying distribution
2. It can generate examples that are suitably different from the observations in training data. In other words, the model shouldn't simply reproduce things it has already seen.

Empirically speaking, for models with same level of complexity, it takes more data to train generative model than its discriminative competitor, as is not surprisingly just the nature of supervised learning and unsupervised learning.

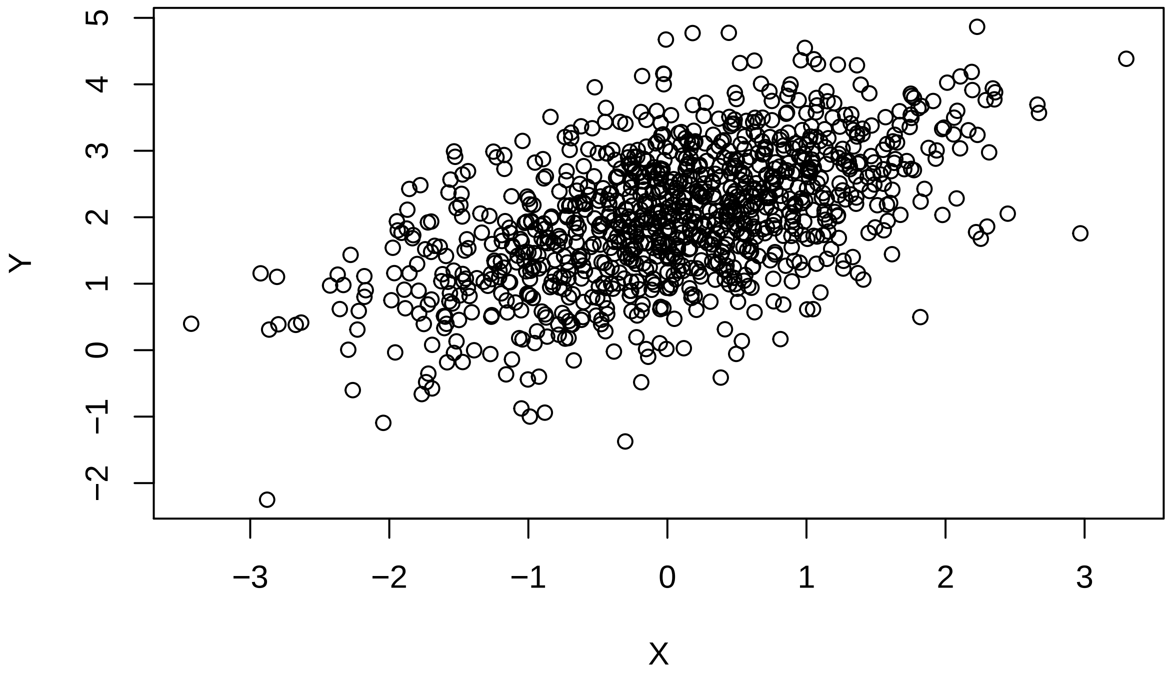
Now as we can roughly divide machine learning tasks into regression, classification, and structured learning, let's see what the two kinds of model do in each case.

## Regression

For this part we will use a gaussian mixture model and several regression models for comparison. Consider a simple bivariate normal distribution.

```r
library(MGMM)
mu_XY = c(0,2)
sigma_XY = matrix(c(1,0.5,0.5,1),2,2,byrow = TRUE)
N = 1000
```

```r
data = rGMM(N, d = 2, k = 1, means = mu_XY, covs = sigma_XY)
colnames(data) = c("X","Y")
plot(data)
```



To model the relationship between $X$ and $Y$, we can simply run a linear regression $Y = \beta_0 + \beta_1 X$

```r
linear_fit = lm(Y~X,as.data.frame(data))
summary(linear_fit)
```

```
##
## Call:
## lm(formula = Y ~ X, data = as.data.frame(data))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1982 -0.5474  0.0177  0.5785  2.6888
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.99073    0.02739   72.69   <2e-16 ***
## X            0.54799    0.02714   20.19   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8648 on 998 degrees of freedom
## Multiple R-squared:   0.29,  Adjusted R-squared:  0.2893
## F-statistic: 407.6 on 1 and 998 DF,  p-value: < 2.2e-16
```

Or we can fit a gaussian mixture model to capture the joint distribution of $X$ and $Y$.

```r
gmm_fit = fit.GMM(data, k = 1)
gmm_fit$Covariance
```

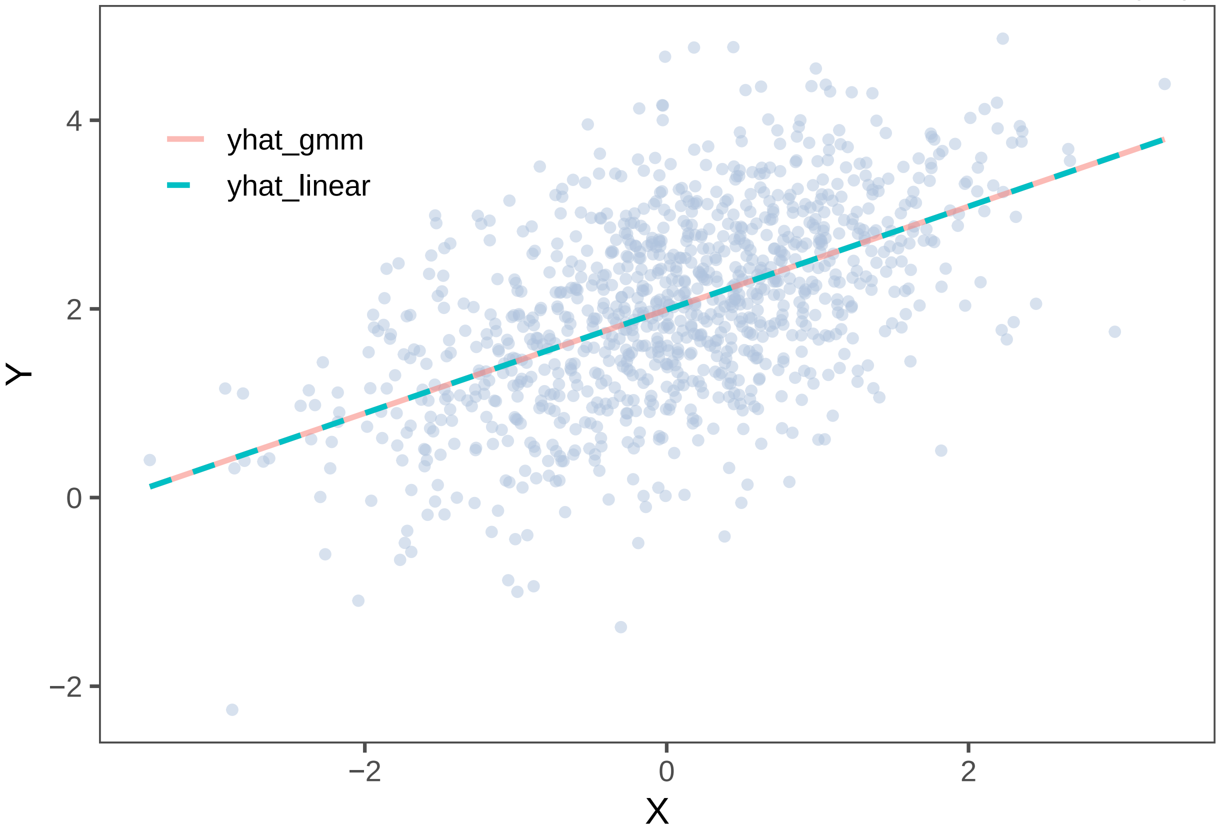
```
##           [,1]      [,2]
## [1,] 1.0160888 0.5568037
## [2,] 0.5568037 1.0522328
```

As we know by definition, $\hat{\beta}_1 = \frac{Cov(X,Y)}{Var(X)}$ and $\beta_0 = \bar{Y} - \hat{\beta}_1 \bar{X}$, we can simply plug in parameters of the joint distribution to derive the discriminative relationship as follows

```r
beta1 = gmm_fit$Covariance[1,2]/gmm_fit$Covariance[1,1]
beta0 = mean(data[,2]) - beta1*mean(data[,1])
gmm_predict_y = function(x){
  yhat = beta0 + beta1*x
}
print(c(beta0,beta1))
```

```
## [1] 1.9907263 0.5479872
```

For this simple case the two estimator have almost the same linear predictive relationship. Plotting together



The difference between the two models is not obivious in this simple setting, where the distribution is simply Gaussian instead of Gaussian Mixture. But the generative nature of GMM allows more applications. For example, a popular generative model, variation autoencoder(VAE) estimates an infinite conditional gaussian mixture model in the latent space of encoded input $x$ on $z$ with $x|z \sim N(\mu(z), \sigma(z))$ to model randomness in decoded output.

## Classification

Now for the classification task, a typical competitive pair would be logistic regression and Naive Bayes. Consider a binary classification task with discrete data. $\mathcal{X} = \{0,1\}^k$ and $\mathcal{Y} = \{0,1\}$ (Think about the case of finding a partner for homeworks by checking if the guy satisfys all of you criteria: Registered in Prof.Mao's Microeconometrics course? Yes, $X[,1] = 1$, Loves coding in R? Yes, $X[,2] = 1$, always starts to write homework few hours before the ddl with Cardi.B's songs on? Holy crap no, then $X[,3] = 0, \ldots$, Prefer Marvel to DC?$X[,???] = 1$ . Given all these $X$ values you can decide if you want to work with that guy, i.e.

$Y=1$ or $Y=0$.) Now let there be a joint distribution over $\mathcal{X}\times\mathcal{Y}$ and we sample $\mathcal{D}=\left\{x^{(i)},y^{(i)}\right\}_{i=1}^{m}$ from that for training. Each of the observation is noted as $\mathbf{x}_i=[X_i,Y_i]$

For the generative model, as we do not know explicitly the underlying mechanism $p_{data}$, a direct way would be to model each possible combination of features, based on the data we have seen, which would means terrible amount of parameters to estimate. But since we don't have much time before the ddl, we need to start from a pretty strong assumption that, each feature is independent of other features, which is also known as Naive Bayes assumption.

$$p\left(x_j \mid x_k\right)=p\left(x_j\right)$$

And now by chain rules

$$\begin{aligned}
p(\mathbf{x}) &= p\left(x_1,\ldots,x_K,y\right)\\
&= p\left(x_2,\ldots,x_K,y \mid x_1\right)p\left(x_1\right)\\
&= p\left(x_3,\ldots,x_K,y \mid x_1,x_2\right)p\left(x_2 \mid x_1\right)p\left(x_1\right)\\
&= p(y \mid x_1,\ldots,x_{k-1},y)\prod_{k=1}^{K}p\left(x_k \mid x_1,\ldots,x_{k-1},y\right)
\end{aligned}$$

Applying Naive Bayes assumption

$$p(\mathbf{x})=p(y)\prod_{k=1}^{K}p\left(x_k\right)$$

Use sample frequency as MLE estimates for the parameters $\hat{\theta_{kl}}=\hat{p}\left(x_k=l\right)=\frac{\sum_{i=1}^{N}\mathcal{I}(x_k=l)}{N}$

The discriminative analog of naive Bayes is the logistic regression:

$$p(y=1 \mid x;\beta,\theta)=\frac{1}{\left(1+\exp\left(-\beta^T x-\theta\right)\right)}$$

Now again consider a very simple binary case

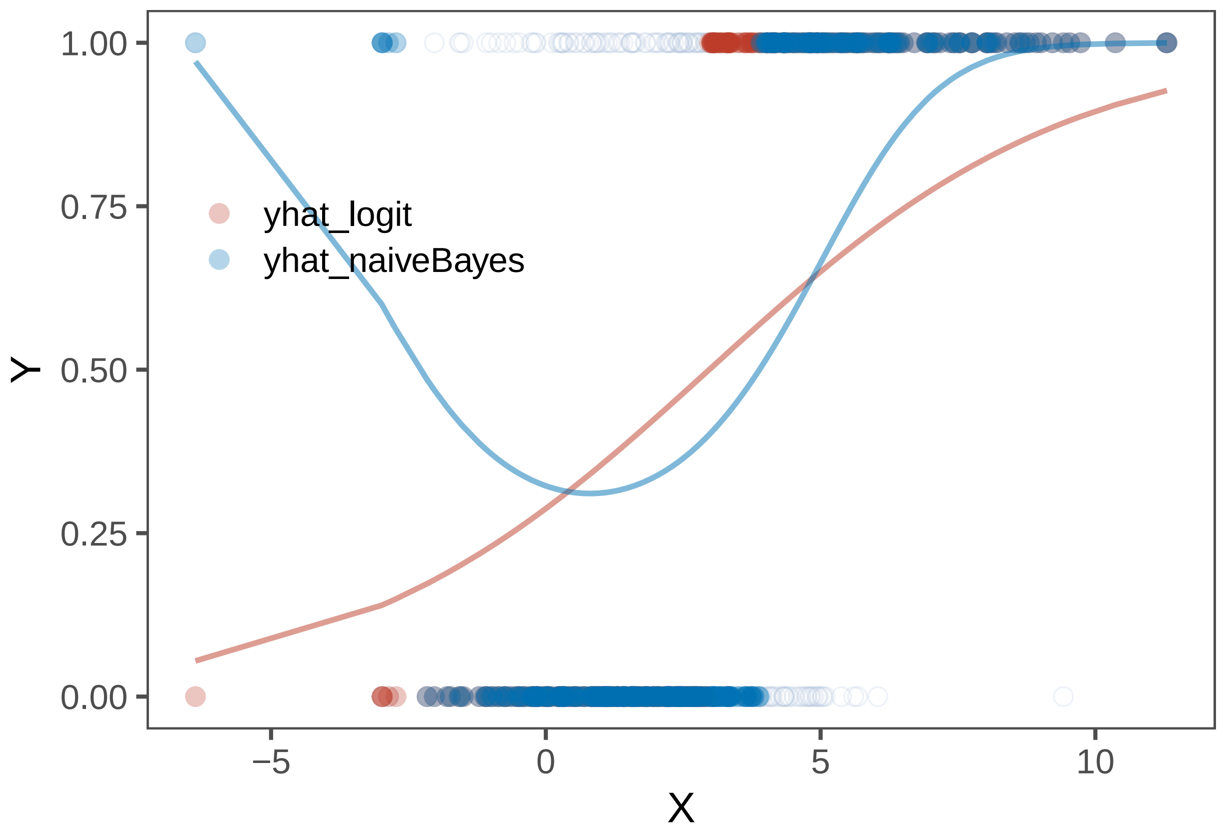
```r
require(AER)
require(e1071)
N = 200
x0 = rnorm(N,2,2) #x|y=0
x1 = rnorm(N,4,3)
Y = as.factor(c(rep(0,N),rep(1,N)))
X = c(x0,x1)
binarydata = data.frame(Y,X)
```

Fit the two models

```r
logit_fit = glm(Y ~ X, family = binomial,data = binarydata)
naiveBayes_fit = naiveBayes(Y~X, data = binarydata)
```

and make predictions on training set

plot together to see the results

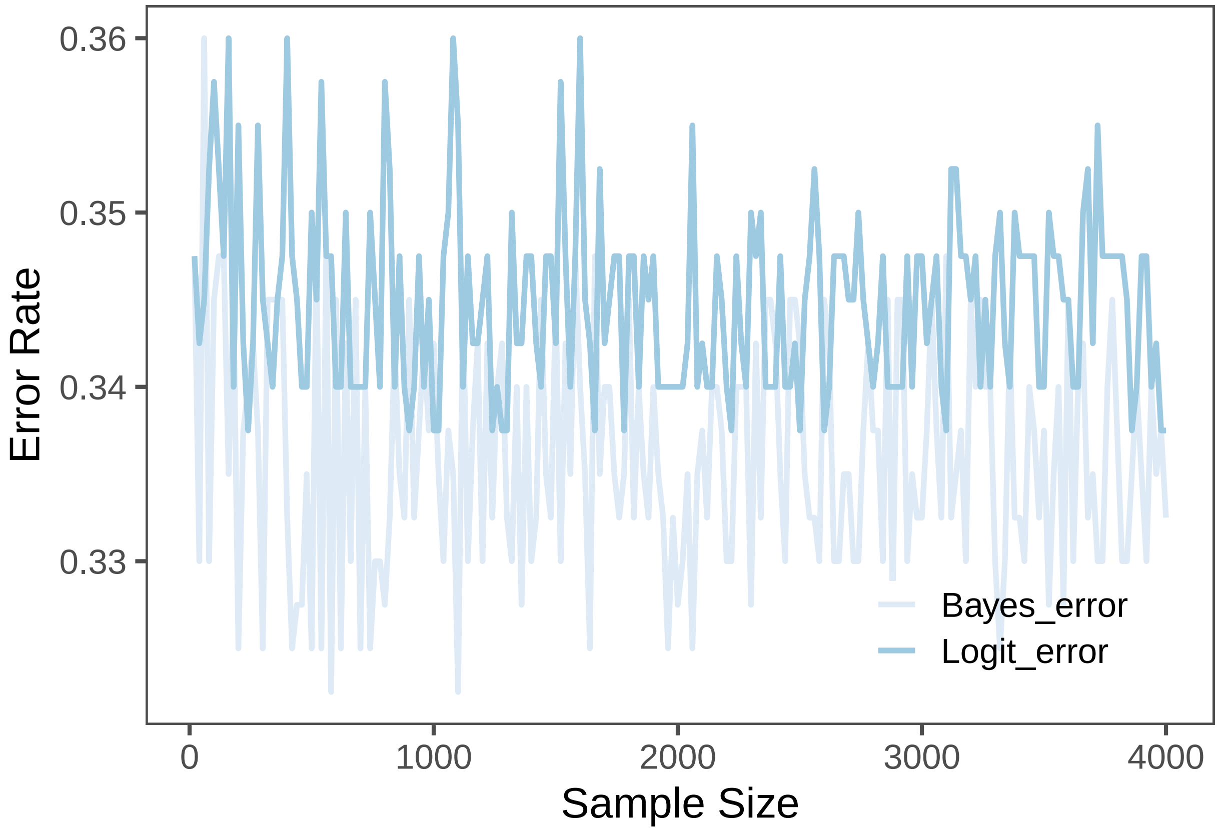


There seems little difference here between two models, however, when it comes to asymptotic error, even though the discriminative logistic regression algorithm has a lower one, the generative naive Bayes classifier may converge more quickly to its (higher) asymptotic error (Jordan, 2002). Thus, as the number of training samples is increased, one would expect generative naive Bayes to initially do better, but for discriminative logistic regression to eventually catch up to, and quite likely overtake, the performance of naive Bayes.

So continue with the settings above and change the sample size, we can derive the following relationship between sample size and predictive accuracy of the two models.

```
sample_size = seq(10,2000,10)
test_size = 200
test_set = data.frame(X = c(rnorm(test_size,2,2),rnorm(test_size,4,3)),Y = as.factor(c(rep(0,test_size)
L = length(sample_size)
Logit_error = rep(0,L)
Bayes_error = rep(0,L)
i=1
for (i in 1:L){
  N = sample_size[i]
  x0 = rnorm(N,2,2) #x|y=0
  x1 = rnorm(N,4,3)
  Y = as.factor(c(rep(0,N),rep(1,N)))
  X = c(x0,x1)
  binarydata = data.frame(Y,X)
  logit_fit = glm(Y ~ X, family = binomial,data = binarydata)
  naiveBayes_fit = naiveBayes(Y~X, data = binarydata)
  Logit_pred_phat = predict(logit_fit,newdata = test_set,type = "response")
  Logit_pred = factor(as.numeric(Logit_pred_phat>0.5))
  Logit_error[i] =  sum(Logit_pred != test_set$Y)/(test_size*2)
  NB_pred = predict(naiveBayes_fit,newdata=test_set,type="raw")
  NB_pred = factor(as.numeric(NB_pred[,2]>0.5))
  Bayes_error[i] = sum(NB_pred != test_set$Y)/(test_size*2)
}
```

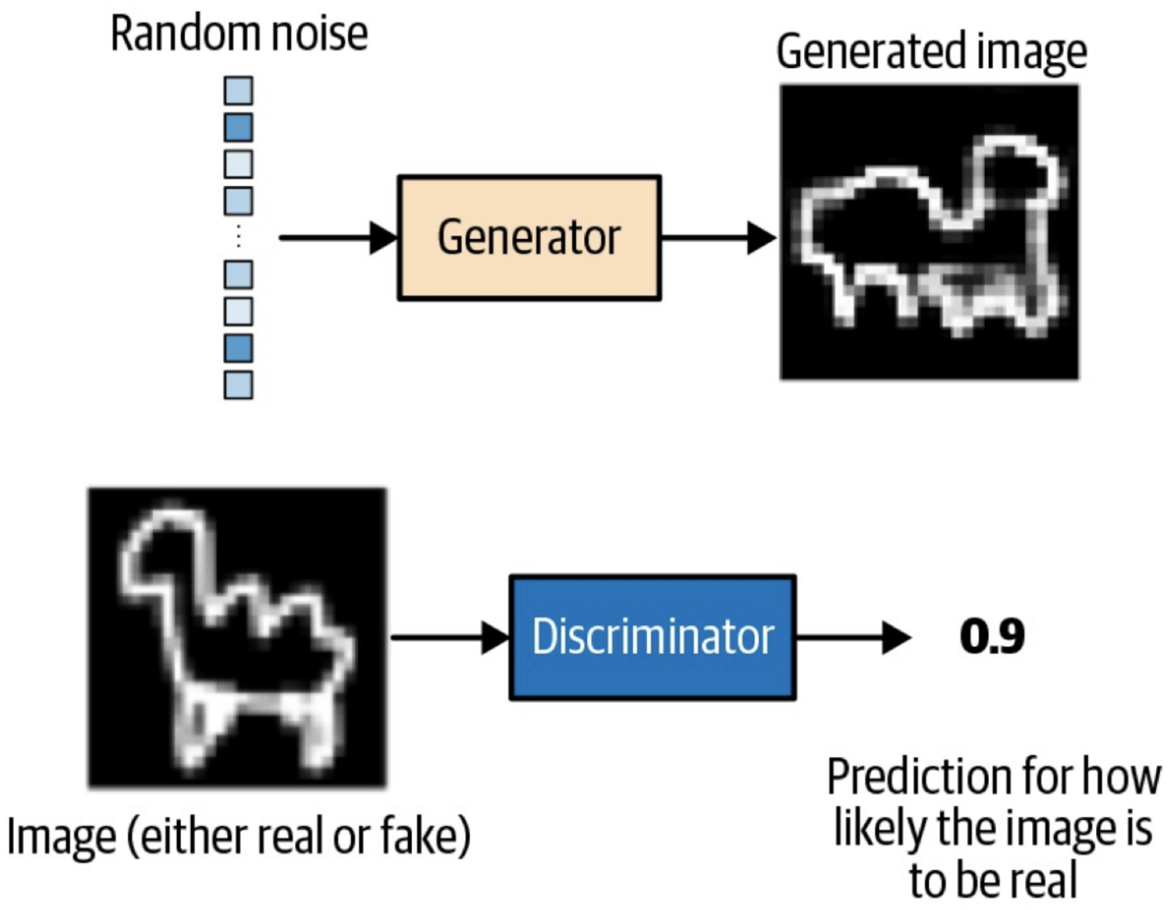
Plot the results



## Structured Learning and Combination of Generative Models and Discriminative Models

Another popular machine learning task in recent years is structured learning, where the target or feature has specific inner structure. A typical scenario is picture generation where the output is an image with certain color and shape structure. For example, consider the widely used (and p-hacked) dataset MNIST containing hand-written pics for numbers $0 \sim 9$. Conditional generative modelling might aims at generating pictures of hand-written numbers according to input numbers. A typical discriminative model might tries to classify which number is written in the input picture.

The two classes of modelling cooperates well in Generative Adversarial Networks (Goodfellow et al.,2014)


Random noise
Generator
Generated image
Discriminator
0.9
Image (either real or fake)
Prediction for how likely the image is to be real

A generator (generative model) tries to generate pictures as realistic as possible while a discriminator (discriminative model) tries to distinguish the generated fake pictures from real ones. The adversarial nature of the two models suggests that through iterations of interaction and optimization, the two can both improve.

**Algorithm 1** Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, $k$, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

```
for number of training iterations do
  for $k$ steps do
    • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of $m$ examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:
        $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$
  end for
  • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by descending its stochastic gradient:
      $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$
end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.
```

A naive analogy would be a student trying to create fake signatures on written request for leave, and a TA judging if the request is approved by the professor or not. At the very first, the inexperienced student will randomly draw something, like "Yoda", and it's easy for TA to judge and penalize. Then the student can learn from this failure, and generate an updated version, maybe at least correctly spelling "Jiaming Mao". If one day TA fails to distinguish the signature, he will also learn how to better classify real signatures from fake. Though we never take attendance check, generative and discriminative models can still work better when trained together in this adversarial way.

## References

[1] Goodfellow, Ian J., et al. "Generative adversarial networks." arXiv preprint arXiv:1406.2661 (2014).

[2] Jordan, A. "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes." Advances in neural information processing systems 14.2002 (2002): 841.

[3] David Foster, "Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play" O'Reilly Media (2019)