IFT 6085 - Lecture 12
Generative models

This version of the notes has not yet been thoroughly checked. Please report any bugs to the scribes or instructor.

Scribes
Winter 2019: Stephan Tran & Shawn Tan
Winter 2018: Nicolas Gagné

Instructor: Ioannis Mitliagkas

1 Summary

In this lecture we introduce generative models. We start by contrasting the discriminative vs generative paradigm with an example. We then introduce a generative model – the Gaussian Discriminant Analysis model – and demonstrate that it reduces to its discriminative counterpart: logistic regression. Having done so, we will have learned a lesson: generative models require stronger assumptions than discriminative models. When our assumptions are correct, generative models perform better than discriminative models, but if our assumptions are wrong, they may perform much worse than their more robust discriminative counterpart. Next, we make a further distinction among generative models: those with a prescribed explicit specification and those where the distribution is implicit and induced by a procedure. We conclude this lecture by introducing an example of implicit generative models—generative adversarial networks.

2 Discriminative vs Generative models

In this section, we introduce generative models by contrasting them with discriminative models. We start with an example drawn from Andrew Ng online notes [3].
We consider a classification problem in which we want to learn to distinguish between elephants ($y = 1$) and dogs ($y = 0$), based on some features of an animal. Given a training set, an algorithm like logistic regression tries to find a straight line that separates the elephants from the dogs. In order to classify a new animal, we just check on which side of the boundary it falls, and make our prediction accordingly. This approach corresponds to what is known as a **discriminative** model; a discriminative model tries to directly learn a (possibly stochastic) mapping $p(y|x)$ from the space of input X to the labels $\{0, 1\}$.
Here's a contrasting approach. First, looking at elephants, we build a model of what elephants look like. Similarly, looking at dogs, we build a separate model of what dogs look like. Now, in order to classify a new animal, we match the new animal against the elephant model, and match it against the dog model. We predict according to whether the new animal looks more like the elephants or more like the dogs we have seen in the training set. This approach corresponds to what is known as a **generative** model; a generative model tries to learn $p(x|y)$ and $p(y)$. In our case where y indicates whether an example is a dog (0) or an elephant (1), we have that $p(x|y = 0)$ models the distribution of dogs' features, and $p(x|y = 1)$ models the distribution of elephants' features. After modelling $p(y)$, called the **class priors**, and $p(x|y)$, our algorithm can then use Bayes rule to derive the posterior distribution on y given x:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{p(x|y)p(y)}{\sum_y p(x|y)p(y)}.$$

We will explore the generative cousin of logistic regression in the next section.

Discriminative	Generative
Weaker assumptions	Stronger assumptions
More robust	Better / faster fit when assumptions are correct
More widely used for classification	Can perform badly

Table 1: A summary of properties of discriminative and generative techniques.

3 Gaussian Discriminant Analysis

In *Gaussian Discriminant Analysis*, we assume that $p(x|y)$ is distributed according to a multivariate normal distribution. We recall the definition of the multivariate normal distribution.

Definition 1 (Multivariate normal distribution)**.** *Given a **mean vector** $\mu \in \mathbb{R}^n$ and a **covariance matrix** $\Sigma \in \mathbb{R}^{n\times n}$, where $\Sigma \geq 0$ is symmetric and positive-definite, then the **multivariate normal distribution** $\mathcal{N}(\mu, \Sigma)$ is defined by its density:*

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right).$$

3.1 The Gaussian Discriminant Analysis model

The Gaussian Discriminative Analysis (GDA) model has parameters ϕ, Σ, μ_0 and μ_1 as follows:

$$\begin{aligned} y &\sim \text{Bernouilli}(\phi) \\ p(x|y=0) &\sim \mathcal{N}(\mu_0, \Sigma) \\ p(x|y=1) &\sim \mathcal{N}(\mu_1, \Sigma) \end{aligned}$$

Writing out the distributions, this is:

$$\begin{aligned} p(y) &= \phi^y(1-\phi)^{1-y} \\ p(x|y=0) &= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\right) \\ p(x|y=1) &= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right) \end{aligned}$$

Note that although we have distinct mean vectors μ_0 and μ_1, we have the same covariance matrix Σ. In terms of our dogs (0) and elephants (1) example above, these parameters can be interpreted as:

- ϕ is the proportion of elephants in our population, whereas $1-\phi$ is the proportion of dogs.
- The elephants and dogs features are generated according to a multivariate normal centered at μ_1 and μ_0, respectively. Both multivariate normals have the same variance, so this implies that features for both animals have a similar spread; an assumption that might not quite hold if we haven't normalized the features (for instance, the animal's weight and height).

Remember that we train generative models by building a model of what elephants look like and of what dogs look like. We do so by finding the parameters that maximize the log-likelihood of our data (the observed animals). Given a training set $S := \left((x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\right)$, the log-likelihood of GDA for S is

$$\begin{aligned} \ell(\phi, \mu_0, \mu_1, \Sigma) &= \log \prod_{i=1}^{m} p\left(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma\right) \\ &= \log \prod_{i=1}^{m} p\left(x^{(i)}|y^{(i)}; \mu_0, \mu_1, \Sigma\right) p\left(y^{(i)}; \phi\right) \end{aligned}$$

Maximizing the log-likelihood and we get

$$\begin{aligned}
\phi^{ML} &= \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{\left(y^{(i)}=1\right)} \\
\mu_0^{ML} &= \frac{\sum_{i=1}^{m} \mathbb{1}_{\left(y^{(i)}=0\right)} x^{(i)}}{\sum_{i=1}^{m} \mathbb{1}_{\left(y^{(i)}=0\right)}} \\
\mu_1^{ML} &= \frac{\sum_{i=1}^{m} \mathbb{1}_{\left(y^{(i)}=1\right)} x^{(i)}}{\sum_{i=1}^{m} \mathbb{1}_{\left(y^{(i)}=1\right)}} \\
\Sigma^{ML} &= \frac{1}{m}\sum_{i=1}^{m} \left(x^{(i)} - \mu_{y^{(i)}}\right)\left(x^{(i)} - \mu_{y^{(i)}}\right)^T
\end{aligned}$$

Now that we have captured what an elephant and what a dog should look like, if we are given a new animal x to classify, we can predict according to whether it looks more like an elephant or more like a dog, i.e., we return

$$\arg\max_{y\in\{0,1\}} p\left(y|x;\phi^{ML},\mu_0^{ML},\mu_1^{ML},\Sigma^{ML}\right)$$

which can be computed directly by

$$\arg\max_{y\in\{0,1\}} p\left(x|y;\mu_0^{ML},\mu_1^{ML},\Sigma^{ML}\right) p(y;\phi^{ML}).$$

We next show that a GDA model can be reduced to logistic regression.

3.2 GDA model and logistic regression

We will argue that if $p(x|y)$ is a multivariate gaussian, then $p(y|x)$ necessarily follows a logistic function. More precisely:

Theorem 2. *Given a Gaussian Discriminant Analysis model, the quantity $p(y=1|x;\phi,\mu_0,\mu_1,\Sigma)$, seen as a function of x, can be expressed in the form*

$$p(y=1|x;\phi,\Sigma,\mu_0,\mu_1) = \frac{1}{1+\exp(-\theta^T x + b)},$$

where θ and b are some appropriate functions of Σ, μ_0, μ_1 and ϕ.

Proof.

$$\begin{aligned}
p(y=1|x) &= \frac{p(x|y=1)p(y=1)}{p(x)} && (1)\\
&= \frac{p(x|y=1)p(y=1)}{p(x|y=1)p(y=1)+p(x|y=0)p(y=0)} && (2)\\
&= \frac{1}{1+\frac{p(x|y=0)p(y=0)}{p(x|y=1)p(y=1)}}. && (3)
\end{aligned}$$

Where we get (1) by definition of conditional probability and (2) by the law of total probability. Taking a closer look

at

$$
\begin{aligned}
&\frac{p(x|y=0)p(y=0)}{p(x|y=1)p(y=1)} \\
&= \exp\left(-\frac{(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)}{2} + \frac{(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)}{2}\right)\frac{1-\phi}{\phi} \\
&= \exp\left(-\frac{x^T\Sigma^{-1}x - 2\left(\mu_0^T\Sigma^{-1}x\right) + \mu_0^T\Sigma^{-1}\mu_0}{2} + \frac{x^T\Sigma^{-1}x - 2\left(\mu_1^T\Sigma^{-1}x\right) + \mu_1^T\Sigma^{-1}\mu_1}{2}\right)\frac{1-\phi}{\phi} \\
&= \exp\left(\frac{2\left(\mu_0^T\Sigma^{-1}x\right) - \mu_0^T\Sigma^{-1}\mu_0 - 2\left(\mu_1^T\Sigma^{-1}x\right) + \mu_1^T\Sigma^{-1}\mu_1}{2}\right)\frac{1-\phi}{\phi} \\
&= \exp\left(\frac{2(\mu_0-\mu_1)^T\Sigma^{-1}x - (\mu_0-\mu_1)^T\Sigma^{-1}(\mu_0+\mu_1)}{2}\right)\exp\left(\log\left(\frac{1-\phi}{\phi}\right)\right) \\
&= \exp\left(\left(\underbrace{(\mu_0-\mu_1)\Sigma^{-1}}_{\theta}\right)^T x + \underbrace{\left(-\frac{(\mu_0-\mu_1)^T\Sigma^{-1}(\mu_0+\mu_1)}{2} + \log\left(\frac{1-\phi}{\phi}\right)\right)}_{b}\right) = \exp\left(\theta^T x + b\right).
\end{aligned}
$$

So we can indeed write

$$p(y=1|x;\phi,\Sigma,\mu_0,\mu_1) = \frac{1}{1+\frac{p(x|y=0)p(y=0)}{p(x|y=1)p(y=1)}} = \frac{1}{1+\exp\left(\theta^T x + b\right)},$$

as desired. □

We just argued that if $p(x|y)$ are multivariate normals with shared Σ, then $p(y=1|x;\theta,\Sigma,\mu_0,\mu_1)$ necessarily follows a logistic function. **GDA reduces to logistic regression. The converse, however, is not true;** i.e., $p(y=1|x,\theta,b)$ being a logistic function does not imply $p(x|y)$ are multivariate normals. **This shows that GDA makes stronger modelling assumptions.** When these modelling assumptions are correct, then GDA will find better fits to the data, and is a better model. Specifically, when $p(x|y)$ are indeed multivariate normals with shared Σ, then GDA is *asymptotically efficient*. Informally, this means that in the limit of very large training sets, i.e., for m large enough, there is no algorithm that is strictly better than GDA. In contrast, logistic regression is more *robust* and less sensitive to incorrect modeling assumptions. Indeed, because of the weaker assumptions of the logistic regression models, it could be applied to multiple distributions of data (Gaussian or Poisson for example). Also, when data is non-Gaussian, then in the limit of large datasets, logistic regression will almost always do better than GDA. For this reason, in practice logistic regression is used more often than GDA.
In the next section, we make a further distinction between two types of generative models.

4 Prescribed vs Implicit Generative models

This section and the next are inspired by [2].
Prescribed Generative models are those that provide an explicit specification of the distribution of an observed random variable x; inducing a log-likelihood function $\log q_\theta(x)$ with parameters θ. For instance, the generative model presented above—Gaussian discriminant analysis—is a prescriptive generative model. A difficulty with this type of model is to assume the correct distribution of the input variable for complex problems. To avoid bad assumptions, one could get rid of the task of assuming a model and let the training data generate a distribution that would fit the observed random variable. This is the purpose of the following type of model.
Alternatively, **Implicit Generative models** are those that provide a *procedure* that generates data. More precisely, implicit generative models use a latent variable z and transform it using a deterministic function $G_\theta : \mathbb{R}^m \to \mathbb{R}^d$, where θ is indexing a family of such functions. Given a probability measure q on $\mathbb{R}^m$, G_θ induces a probability measure $\hat{p}(x)$ on $\mathbb{R}^d$:

$$\hat{p}(E) := q\left(G_\theta^{-1}(E)\right), \text{ for } E \text{ a (measurable) subset of } \mathbb{R}^d.$$

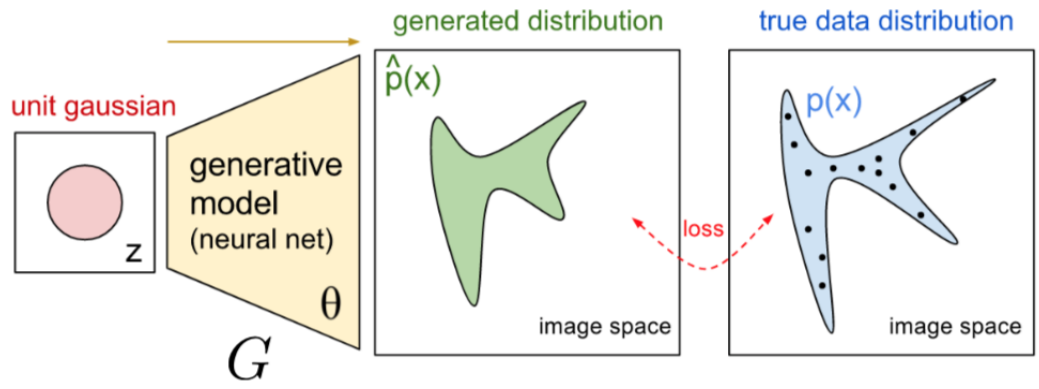

Figure 1: A cartoon of an implicit generative model where q is a unit gaussian and G_θ is a neural network.

The objective would be to find a θ such that the induced $\hat{p}(x)$ is as 'close' as possible to the true data distribution $p(x)$. One of the main challenges is that computing $\hat{p}(x)$ can be highly intractable; for instance, when G_θ is specified by a deep neural network. This difficulty motivates the need for methods that side-step the intractability of computing the likelihood. Also, the loss function of the implicit generative model shown in the figure above still has to be specified. Generative adversarial networks (GANs), among other approaches, provide a solution for these types of problem. We explore GANs in the next section.

5 Generative Adversarial Networks

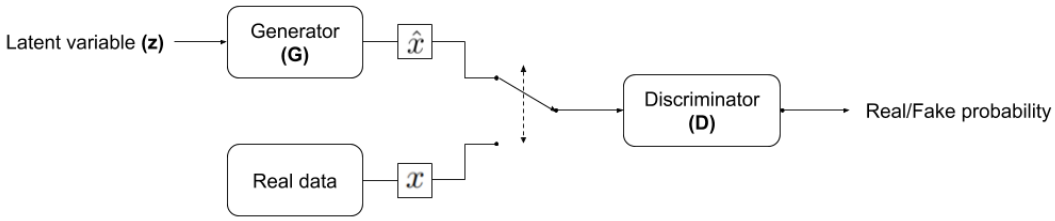

Figure 2: A simplified schematic of a generative adversarial network

We start with our **generator** G_θ introduced above. If we want to train G, we need a way to assess how 'close' the generated $\hat{p}(x)$ is to the true data distribution $p(x)$. In order to do so, we introduce a **discriminator** D whose task is to tell $p(x)$ apart from $\hat{p}(x)$; if a discriminator D can't tell if an instance x came from $p(x)$ or $\hat{p}(x)$, then—according to D—these two distributions are 'close' to each other.

Given an instance x, $D(x) \in [0, 1]$ reflects how strongly D believes x to be a sample from the true distribution $p(x)$. When x is indeed from the true distribution, the loss incurred by D when predicting $D(x)$ is $-\log D(x)$. Conversely, when x comes from the G generated distribution $\hat{p}(x)$, the loss incurred by D when predicting $D(x)$ is $-\log(1 - D(x))$. If we choose to sample from $p(x)$ half of the time and to sample from $\hat{p}(x)$ the other half of the time, then the expected loss of D is

$$\mathcal{L}(D) = -\frac{1}{2}\mathbb{E}_{x\sim p(x)}\left[\log D(x)\right] - \frac{1}{2}\mathbb{E}_{x\sim \hat{p}(x)}\left[\log\left(1 - D(x)\right)\right].$$

So, for a given generator G, the discriminator tries to minimize the above, which is equivalent to the following:

$$\max_D \; \mathbb{E}_{x\sim p(x)}\left[\log D(x)\right] + \mathbb{E}_{x\sim \hat{p}(x)}\left[\log\left(1 - D(x)\right)\right].$$

The generator G, on the other hand, wants to 'preemptively' generate the worst distribution $\hat{p}(x)$ for its adversary D:

$$\min_{\hat{p}(x)} \max_{D} \mathbb{E}_{x\sim p(x)} \left[\log D(x)\right] + \mathbb{E}_{x\sim \hat{p}(x)} \left[\log\left(1 - D(x)\right)\right],$$

by abusing notation, we rewrite it as:

$$\min_{G} \max_{D} \underbrace{\mathbb{E}_{x\sim p} \left[\log D(x)\right] + \mathbb{E}_{z\sim q} \left[\log(1 - D\left(G(z)\right))\right]}_{V(G,D)}.$$

Setups like the one above are called **generative adversarial networks** (GANs). They are often framed as instances of *adversarial* optimization; for instance, interpreting D and G as playing a minimax game with value function $V(G, D)$. Generative adversarial networks can be easily implemented. For instance, the initial implementation for solving the above min-max problem was based on this simple *iterative* algorithm: first fix G, then maximize over D; next, fix D, then maximize over G; repeat until satisfied. So no approximate inference nor estimation of partition function was needed.

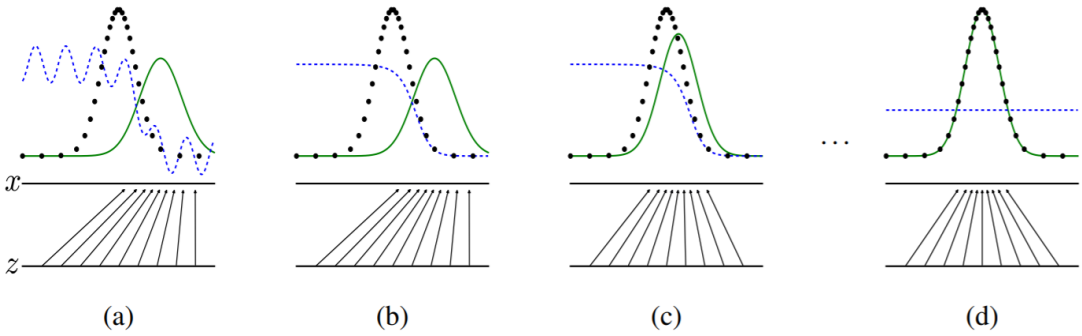

Figure 3: Training of generative adversarial nets [1]

In the figures above, the blue dashed line, the black dotted line and the green solid line illustrate the discriminative distribution, the data generating distribution and the generative distribution respectively. Under each graph, the arrows show how the latent space z is mapped to x through the generator ($x = G(z)$). (a) First, consider the case where p_g is similar to p_{data}. (b) The discriminator is trained to determine if its input is from the p_g or p_{data} distribution. (c) After an update to G, gradient of D has guided $G(z)$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{data}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(x) = \frac{1}{2}$.

5.1 Global optimality of $p_g = p_{data}$

Subsection from [1].

Proposition 3. *For G fixed, the optimal discriminator D is*

$$D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

Proof. The training criterion for the discriminator D, given any generator G, is to maximize the quantity $V(G, D)$.

$$\begin{aligned} V(G, D) &= \mathbb{E}_{x\sim p_{data}(x)} \left[\log D(x)\right] + \mathbb{E}_{z\sim p_z(z)} \left[\log(1 - D\left(G(z)\right))\right] \\ &= \int_x p_{data}(x) \log(D(x)) dx + \int_z p_z(z) \log(1 - D(G(z))) dz \\ &= \int_x p_{data}(x) \log(D(x)) + p_g(x) \log(1 - D(x)) dx \end{aligned}$$

For any $(a, b) \in \mathbb{R}^2 \backslash \{0, 0\}$, the function $y \to a \log(y) + b \log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. In this case, a is $p_{data}(x)$ and b is $p_g(x)$. The discriminator does not need to be defined outside of $Supp(p_{data}) \cup Supp(p_g)$. □

Note that the training objective for D can be interpreted as maximizing the log-likelihood for estimating the conditional probability $P(Y = y|x)$, where Y indicates whether x comes from p_{data} (with y = 1) or from p_g (with y = 0). The minimax game in equation can now be reformulated as:

$$\begin{aligned} C(G) &= \max_D V(G, D) \\ &= \mathbb{E}_{x \sim p_{data}} [\log D_G^*(x)] + \mathbb{E}_{z \sim p_z} [\log(1 - D_G^*(G(z)))] \\ &= \mathbb{E}_{x \sim p_{data}} [\log D_G^*(x)] + \mathbb{E}_{x \sim p_g} [\log(1 - D_G^*(x))] \\ &= \mathbb{E}_{x \sim p_{data}} \left[\log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right] + \mathbb{E}_{x \sim p_g} \left[\log \frac{p_g(x)}{p_{data}(x) + p_g(x)} \right] \end{aligned}$$

Theorem 4. *The global minimum of the virtual training criterion C(G) is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

Proof. For $p_g = p_{data}$, $D_G^*(x) = \frac{1}{2}$, (consider Proposition above). Then,

$$\begin{aligned} C(G) &= \mathbb{E}_{x \sim p_{data}} [\log D_G^*(x)] + \mathbb{E}_{x \sim p_g} [\log(1 - D_G^*(x))] \\ &= \mathbb{E}_{x \sim p_{data}} \left[\log \frac{1}{2} \right] + \mathbb{E}_{x \sim p_g} \left[\log(1 - \frac{1}{2}) \right] \\ &= \log \frac{1}{2} + \log \frac{1}{2} \\ &= -\log 4 \end{aligned}$$

To see that this is the best possible value of $C(G)$, reached only for $p_g = p_{data}$, observe that

$$\mathbb{E}_{x \sim p_{data}} [-\log 2] + \mathbb{E}_{x \sim p_g} [-\log 2] = -\log 4$$

and that by substracting this expression from $C(G) = V(D_G^*, G)$, we obtain:

$$C(G) = -\log(4) + KL(p_{data} || \frac{p_{data} + p_g}{2}) + KL(p_g || \frac{p_{data} + p_g}{2})$$

where KL is the Kullback-Leibler divergence. We recognize in the previous expression the Jensen-Shannon divergence between the models distribution and the data generating process:

$$C(G) = -\log(4) + 2 \cdot JSD(p_{data} || p_g)$$

Since the JensenShannon divergence between two distributions is always non-negative and zero only when they are equal, we have shown that $C^* = -\log(4)$ is the global minimum of $C(G)$and that the only solution is $p_g = p_{data}$, i.e., the generative model perfectly replicating the data generating process. □

GANs can be difficult to train: gradients saturation and mode collapse can be problematic. In the next lecture, we will see how we can mitigate some of those problems by using what is known as the *Wasserstein GAN*.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *arXiv:1406.2661*, 2014.

[2] S. Mohamed and B. Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

[3] A. Ng. CS229 Lecture notes generative learning algorithms. `http://cs229.stanford.edu/notes/cs229-notes2.pdf`. Accessed: 2018-03-05.