Master Notes

Author: Unique Divine

Institute: Applied Technology Solutions, Inc. (ApTSi)

Date: April 12, 2020

Victory won't come to us unless we go to it.

Contents

Chapter 1 Deep Learning & AI

1.1 Important Papers

Goodfellow et al. (2016)

Singh et al. (2015)

1.2 Generative Adversarial Networks

1.2.1 GANs Goodfellow et al. (2014)

Abstract & Introduction

Abstract: Goodfellow et al. propose a new framework for estimating generative models via an adversarial process, in which two models are simultaneously trained:

- G: a generative model that captures the data distribution, and
- D: a discriminative model that estimates the probability that a sample came from the training data rather than G

The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples.

Introduction: In the proposed adversarial nets framework, the generative model is pitted against an adversary: a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution.

- **Counterfeiters Analogy:** The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistiguishable from the genuine articles.

In this article, we explore the special case when the generative model generates samples by passing random noise through a multilayer perceptron, and the discriminative model is also a multilayer perceptron. We refer to this special case as adversarial nets. In this case, we can **train both models using only the highly successful backpropagation and dropout algorithms** [17] and sample from the generative model using only forward propagation.

Adversarial Net Algorithm

The adversarial modeling framework is most straightforward to apply when the models are both multilayer perceptrons. To learn the generator's distribution p_g over data $\boldsymbol{x}$, $p_g(\boldsymbol{x})$, we define a prior on input noise variables, $p_{\boldsymbol{z}}(\boldsymbol{z})$, then represent a mapping to data space as $G(\boldsymbol{z}|\theta_g)$, where G is a differentiable function represented by a multilayer perceptron with parameters θ_g. We also define a second multilayer perceptron $D(\boldsymbol{x}|\theta_d)$ that outputs a single scalar. $D(\boldsymbol{x})$ represents the probability that $\boldsymbol{x}$ came from the data rather than p_g. We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize $\log(1 - D(G(\boldsymbol{z})))$. In other words, D and G play the following two-player minimax game with value function, $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} [\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{z}}(\boldsymbol{z})} [\log (1 - D(G(\boldsymbol{z})))] .$$

Goodfellow et al. present a theoretical analysis of adversarial nets, essentially showing that the training criterion allows one to recover the data generating distribution as G and D are given enough capacity, i.e., in the non-parametric limit.

Figure 1 explains this approach: GANs are trained by simultaneously updating the discriminative distribution, D, so that it discriminates between samples from the data generating distribution, $p_{\boldsymbol{x}}$, and those of the generative distribution p_g (G).

After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate b/w the two distributions, i.e. $D(\boldsymbol{x}) = \frac{1}{2}$.

Algorithm 1: Minibatch SGD training of GAN. The number of steps to apply the discriminator, k, is a hyperparameter. $k = 1$ is the least expensive option.

```
for *number of training iterations* do
    for $k$ *steps* do
        # Apply discriminator
            • 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:
              $$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log \left( D(\boldsymbol{x}^{(i)}) \right) + \log \left( 1 - D(G(\boldsymbol{z}^{(i)})) \right) \right].$$
    end
    # Update generator
        • 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:
          $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left( 1 - D(G(\boldsymbol{z}^{(i)})) \right).$$
end
The gradient-based updates can use any standard gradient-based learning rule.
```

Note:

- $k = 1$ was used in the experiments.
- Momentum was the gradient-based learning rule used in the experiments.

Related Work

The adversarial nets framework does not require a Markov chain for sampling. Because adversarial nets do not require feedback loops during generation, they are better able to leverage piece-wise linear units [19, 9, 10], which improve the performance of backpropagation but have problems with unbounded activation when used in a feedback loop.

1.3 ML Finance Project

example w/ multivariate time series in PyTorch

Q: (cloze) Neural networks can be constructed using the `torch.nn` package.

Q: Import the package for constructing neural networks in PyTorch.

```
import torch.nn as nn
```

Q: (cloze) Seaborn comes with built-in datasets.

Q: Load seaborn's flights dataset.

```
flight_data = sns.load_dataset("flights")
```

Q: Why must time series data be scaled for sequence predictions?

When a network is fit on unscaled data, it is possible for large inputs to slow down the learning and convergence of your network and in some cases prevent the network from effectively learning your problem.

Q: sklearn import for scaling data?

```
from sklearn.preprocessing import MinMaxScaler
```

We know the field is fast moving. If the reader looking for more recent free reading resources, there are some good introductory/tutorial/survey papers on Arxiv; I happen to be compiling a list of them.

One of said review papers Raghu and Schmidt (2020)

(Raghu and Schmidt, 2020, hello)

1.4 Deep Learning for Genomic Risk Scores

> " A central aim of computational genomics is to identify variants (SNPs) in the genome which increase risks for diseases. Current analyses apply linear regression to identify SNPs with large associations, which are collected into a function called a Polygenic Risk Score (PRS) to predict disease for newly genotyped individuals. This project is broadly interested in whether we can improve performance of genomic risk scores using modern machine learning techniques.
>
> A recent study assessed the disease prediction performance of neural networks in comparison to conventional PRSs, but did not find evidence of improvement. This project will explore whether neural networks can improve performance by incorporating gene expression data to the training process. Gene expression is often integrated with SNP data in Transcriptome-Wide Association Studies (TWAS), which bear some resemblance to neural network architectures with SNPs as input nodes, genes as intermediate nodes, and disease status as the output node. Modeling this process as a neural network however will require defining a more unconventional architecture in which a small subset of hidden nodes is anchored to observed values.
>
> This project is designed for students with experience in machine learning topics and preferably with deep learning tools such as tensorflow or pytorch. Students should also be interested in applying machine learning and statistics to genomics applications." - Jie Yuan

Terms to know: Computational genomics, variants, single-nucleotide polymorphism (SNP), genome, Polygenic Risk Score (PRS), Transcriptome-Wide Association Studies (TWAS), gene(s), genomics, genotype, intermediate node (NN), hidden node (NN)

Q: Explain at a high level how PRSs tells us disease risk.

Q: "In what task were neural networks outperformed by conventional PRSs?"

Q: What is meant by "improve performance of genomic risk scores?"

genotype : An individual's collection of genes. Also can refer to the two alleles inherited for a particular gene.

node (NN) : An artificial neural network is an interconnected group of nodes, inspired by a simplification of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of another[1].

hidden node (NN) : A node in a hidden layer.

hidden layer (NN) :

1.4.1 Papers

Polygenic Risk Scores (paper) Wray and Goddard (2010)

Abstract (mining)

recurrence risks : In genetics, the likelihood that a hereditary trait or disorder present in one family member will occur again in other family members[2].

"Evidence for genetic contribution to complex diseases is described by recurrence risks to relatives of diseased individuals."

This is distinguished from recurrence risk for cancer, which is the chance that a cancer that has been treated will recur.

gene : a sequence of DNA that codes for a specific peptide or RNA molecule; the physical and functional unit of heredity.

locus : the position of a gene on a chromosomes

somatic cell : any cell of the body except sperm and egg cells. A non-germline cell. any biological cell forming the body of an organism (except gametes).

sôma (Ancient Greek): body

genome : An organism's complete set of DNA, including all of its genes. Each genome contains all of the information needed to build and maintain that organism. In humans, a copy of the entire genome—more than 3 billion DNA base pairs—is contained in all cells that have a nucleus [3].

"genome-wide association"

allosome : (1) A sex chromosome such as the X and Y human sex chromosomes. (2) An atypical

[1] https://en.wikipedia.org/wiki/Artificial_neural_network

[2] https://www.cancer.gov/publications/dictionaries/genetics-dictionary/def/recurrence-risk

[3] https://ghr.nlm.nih.gov/primer/hgp/genome

chromosome [4].

allo- (Greek): other, differnt

autosome : Any chromosome that is not a sex chromosome. The numbered chromosomes.

auto (Greek): self, one's own, by oneself, of oneself

-some, soma (Greek): body

allele : (genetics) One of a number of alternative forms of the same gene occupying a given position, or locus, on a chromosome.

Borrowed from German Allel, shortened from English allelomorph. Ultimately from the Ancient Greek prefix allēl- from állos ("other").

"their effects and allele frequencies"

allelomorph: another term for allele.

risk loci :

"genome-wide association studies allow a description of the genetics of the same diseases in terms of risk loci..."

haploid : the quality of a cell or organism having a single set of chromosomes.

diploid : the quality of having two sets of chromosomes.

"Sexually reproducing organisms are diploid" (having two sets of chromosomes, one from each parent)

eukaryotes : Organisms whose cells have a nucleus enclosed within a nuclear envelope.

gamete : A mature sexual reproductive cell, as a sperm or egg, that unites with another cell to form a new organism. A haploid cell that fuses with another haploid cell during fertilization in organisms that sexually reproduce. A mature haploid male or female germ cell which is able to unite with another of the opposite sex in sexual reproduction to form a zygote.

gamete (Ancient Greek): to marry

zygote : A eukaryotic cell formed by a fertilization event between two gametes.

zygōtos (Greek): joined. yoked.

monozygotic : Monozygotic (MZ) or identical twins occur when a single egg is fertilized to form one zygote (hence, "monozygotic") which then divides into two separate embryos.

"monozygotic twins"

empirical :

"generate results more consistent with empirical estimates"

genetic variants :

Q: A human cell containing 22 autosomes and a Y chromosome is a sperm.

Neural Networks for Genomic Prediction (paper) Pinto et al. (2019)

[4] https://www.merriam-webster.com/medical/allosome

Transcriptome Wide Association

1.4.2 Meetings

Jie Yuan meeting #2 (Sep 10)

- There's a dot product and its output is passed through some activation function like sigmoid or ReLU. It's still linear in the sense that there's some sort of function that takes in a dot product (which is the definition of a linear operation). You can think of [the betas] as the weights in the neural network because the betas are the coefficients of a linear model.
- Liability is discussed in the multi-locus models paper.
- In the genomics context, this is called the **liability threshold model**. If you google that, you'll find some papers. In a broader machine learning sense, this is called the **probit regression model**. WIkipedia probably has a sufficient article on it.
- The gist of it is that it's a model to map continuous sums (the dot products basically) into binary labels: cases and controls. This can be any sort of logistic regression type thing where you have 0s and 1s. Basically, the **probit model is an alternative to the logistic regression model**.
- The liability is related to what's called the "link function". The link function in the probit model is the normal CDF function. There's a term that goes into the link function, and that term is what we're calling liability.
- The liability is basically the product of the vector of genotypes and the vector of betas. It's the linear part of the generalized linear model.
- Once you have the liability and plug it into the link function. Here, the link function is the probit regression. In a neural network, this would be the linear activation function (sigmoid, ReLU, etc.).
- You know how a sigmoid has a domain that's $(-\infty, \infty)$? It's range is (0,1), so what that function does is map a real value into being a probability. It gives you the probability of being a case. If you look at the normal CDF function, you find that it looks almost the same.

Why is the liability threshold model better than or different from the logistic reg model?

- One reason people use log reg more often is that the betas from log reg are interpretable. The have a meaning in terms of odds ratios. In a log reg model, the effect sizes are the natural log of the odds ratios.
- One disadvantage of log reg compared to liability-threshold is that log reg doesn't have a concept of the underlying distribution of the liability. In log reg, you get the linear $X\beta$ term, but there's no sort of distribution around it. In the liability-thresh model, we say that the liability, itself, is standard normally distributed. By taking the normal CDF, what you're doing is mapping that $X\beta$ value, the liability value, onto this distribution and

asking, “what’s the probability that it’s larger than some threshold?"

- larger than threhsold $\implies$ label as 1, smaller $\implies$ 0.
- Because liability-threshold has that normal distribution, it gives you more concepts to play with such as variance explained, which the logistic reg model doesn’t have.
- liability threshold model in simple terms

  from wikipedia: Liability threshold model

schizophrenia (SZ) example:

- Whether or not someone has SZ comes from a combination of factors in both their genes and environment.
- There are all sorts of variables. Imagine hypothetically that you could collect all of them and produce a score from that (that determines whether or not you have SZ). That score would be the liability.
- The liability is a $\mathcal{N}(0,1)$ distribution that, if higher than the threshold, predicts/indicates the patient has SZ.
- The problem with that is that we don’t actually observe everything. We don’t truly observe every factor that goes into whether you have SZ b/c (1) we can’t measure one’s environment and (2) we can’t know all of the genomic risk factors from GWAS. We can only identify the largest risk factors.

  This means we are identifying only a small fraction of what goes into genetic risk for SZ.
- Essentially, what we have is a small subset of the factors that actually determine your SZ risk (which we’re reasonably confident indicate this relationship). In this example, some subset of the risk factors like the genomic variance which increase your risk for SZ.
- What that means is that inside this standard normal liability distribution/function, what we actually know is a small subset of the factors that go into that. If we score people on that subset of factors, what we get is a smaller distribution, a distribution that has a variance that’s a small fraction of the total liability variance, which we can’t observe. The variance of that small fraction that we observe, that’s the **variance explained**.
-

Chapter 2 Deep Learning & AI

2.1 Important Papers

Goodfellow et al. (2016)

Singh et al. (2015)

2.2 Generative Adversarial Networks

2.2.1 GANs Goodfellow et al. (2014)

Abstract & Introduction

Abstract: Goodfellow et al. propose a new framework for estimating generative models via an adversarial process, in which two models are simultaneously trained:

- G: a generative model that captures the data distribution, and
- D: a discriminative model that estimates the probability that a sample came from the training data rather than G

The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples.

Introduction: In the proposed adversarial nets framework, the generative model is pitted against an adversary: a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution.

- **Counterfeiters Analogy:** The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistiguishable from the genuine articles.

In this article, we explore the special case when the generative model generates samples by passing random noise through a multilayer perceptron, and the discriminative model is also a multilayer perceptron. We refer to this special case as adversarial nets. In this case, we can **train both models using only the highly successful backpropagation and dropout algorithms** [17] and sample from the generative model using only forward propagation.

Adversarial Net Algorithm

The adversarial modeling framework is most straightforward to apply when the models are both multilayer perceptrons. To learn the generator's distribution p_g over data $\boldsymbol{x}$, $p_g(\boldsymbol{x})$, we define a prior on input noise variables, $p_{\boldsymbol{z}}(\boldsymbol{z})$, then represent a mapping to data space as $G(\boldsymbol{z}|\theta_g)$, where G is a differentiable function represented by a multilayer perceptron with parameters θ_g. We also define a second multilayer perceptron $D(\boldsymbol{x}|\theta_d)$ that outputs a single scalar. $D(\boldsymbol{x})$ represents the probability that $\boldsymbol{x}$ came from the data rather than p_g. We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize $\log(1 - D(G(\boldsymbol{z})))$. In other words, D and G play the following two-player minimax game with value function, $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] .$$

Goodfellow et al. present a theoretical analysis of adversarial nets, essentially showing that the training criterion allows one to recover the data generating distribution as G and D are given enough capacity, i.e., in the non-parametric limit.

Figure 1 explains this approach: GANs are trained by simultaneously updating the discriminative distribution, D, so that it discriminates between samples from the data generating distribution, $p_{\boldsymbol{x}}$, and those of the generative distribution p_g (G).

After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate b/w the two distributions, i.e. $D(\boldsymbol{x}) = \frac{1}{2}$.

Algorithm 2: Minibatch SGD training of GAN. The number of steps to apply the discriminator, k, is a hyperparameter. $k = 1$ is the least expensive option.

```
for number of training iterations do
    for $k$ steps do
        # Apply discriminator
            • 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:
                $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log \left( D(\boldsymbol{x}^{(i)}) \right) + \log \left( 1 - D(G(\boldsymbol{z}^{(i)})) \right) \right].$
    end
    # Update generator
        • 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:
            $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left( 1 - D(G(\boldsymbol{z}^{(i)})) \right).$
end
The gradient-based updates can use any standard gradient-based learning rule.
```

Note:

- $k = 1$ was used in the experiments.
- Momentum was the gradient-based learning rule used in the experiments.

Related Work

The adversarial nets framework does not require a Markov chain for sampling. Because adversarial nets do not require feedback loops during generation, they are better able to leverage piece-wise linear units [19, 9, 10], which improve the performance of backpropagation but have problems with unbounded activation when used in a feedback loop.

2.3 ML Finance Project

example w/ multivariate time series in PyTorch

Q: (cloze) Neural networks can be constructed using the `torch.nn` package.

Q: Import the package for constructing neural networks in PyTorch.

```
import torch.nn as nn
```

Q: (cloze) Seaborn comes with built-in datasets.

Q: Load seaborn's flights dataset.

```
flight_data = sns.load_dataset("flights")
```

Q: Why must time series data be scaled for sequence predictions?

When a network is fit on unscaled data, it is possible for large inputs to slow down the learning and convergence of your network and in some cases prevent the network from effectively learning your problem.

Q: sklearn import for scaling data?

```
from sklearn.preprocessing import MinMaxScaler
```

> We know the field is fast moving. If the reader looking for more recent free reading resources, there are some good introductory/tutorial/survey papers on Arxiv; I happen to be compiling a list of them.

One of said review papers Raghu and Schmidt (2020)
(Raghu and Schmidt, 2020, hello)

2.4 Deep Learning for Genomic Risk Scores

> " A central aim of computational genomics is to identify variants (SNPs) in the genome which increase risks for diseases. Current analyses apply linear regression to identify SNPs with large associations, which are collected into a function called a Polygenic Risk Score (PRS) to predict disease for newly genotyped individuals. This project is broadly interested in whether we can improve performance of genomic risk scores using modern machine learning techniques.
>
> A recent study assessed the disease prediction performance of neural networks in comparison to conventional PRSs, but did not find evidence of improvement. This project will explore whether neural networks can improve performance by incorporating gene expression data to the training process. Gene expression is often integrated with SNP data in Transcriptome-Wide Association Studies (TWAS), which bear some resemblance to neural network architectures with SNPs as input nodes, genes as intermediate nodes, and disease status as the output node. Modeling this process as a neural network however will require defining a more unconventional architecture in which a small subset of hidden nodes is anchored to observed values.
>
> This project is designed for students with experience in machine learning topics and preferably with deep learning tools such as tensorflow or pytorch. Students should also be interested in applying machine learning and statistics to genomics applications." - Jie Yuan

Terms to know: Computational genomics, variants, single-nucleotide polymorphism (SNP), genome, Polygenic Risk Score (PRS), Transcriptome-Wide Association Studies (TWAS), gene(s), genomics, genotype, intermediate node (NN), hidden node (NN)

Q: Explain at a high level how PRSs tells us disease risk.

Q: "In what task were neural networks outperformed by conventional PRSs?"

Q: What is meant by "improve performance of genomic risk scores?"

genotype : An individual's collection of genes. Also can refer to the two alleles inherited for a particular gene.

node (NN) : An artificial neural network is an interconnected group of nodes, inspired by a simplification of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of another[1].

hidden node (NN) : A node in a hidden layer.

hidden layer (NN) :

2.4.1 Papers

Polygenic Risk Scores (paper) Wray and Goddard (2010)

Abstract (mining)

recurrence risks : In genetics, the likelihood that a hereditary trait or disorder present in one family member will occur again in other family members[2].

"Evidence for genetic contribution to complex diseases is described by recurrence risks to relatives of diseased individuals."

This is distinguished from recurrence risk for cancer, which is the chance that a cancer that has been treated will recur.

gene : a sequence of DNA that codes for a specific peptide or RNA molecule; the physical and functional unit of heredity.

locus : the position of a gene on a chromosomes

somatic cell : any cell of the body except sperm and egg cells. A non-germline cell. any biological cell forming the body of an organism (except gametes).

sôma (Ancient Greek): body

genome : An organism's complete set of DNA, including all of its genes. Each genome contains all of the information needed to build and maintain that organism. In humans, a copy of the entire genome—more than 3 billion DNA base pairs—is contained in all cells that have a nucleus [3].

"genome-wide association"

allosome : (1) A sex chromosome such as the X and Y human sex chromosomes. (2) An atypical

[1] https://en.wikipedia.org/wiki/Artificial_neural_network

[2] https://www.cancer.gov/publications/dictionaries/genetics-dictionary/def/recurrence-risk

[3] https://ghr.nlm.nih.gov/primer/hgp/genome

chromosome [4].

allo- (Greek): other, differnt

autosome : Any chromosome that is not a sex chromosome. The numbered chromosomes.

auto (Greek): self, one's own, by oneself, of oneself

-some, soma (Greek): body

allele : (genetics) One of a number of alternative forms of the same gene occupying a given position, or locus, on a chromosome.

Borrowed from German Allel, shortened from English allelomorph. Ultimately from the Ancient Greek prefix allēl- from állos ("other").

"their effects and allele frequencies"

allelomorph: another term for allele.

risk loci :

"genome-wide association studies allow a description of the genetics of the same diseases in terms of risk loci..."

haploid : the quality of a cell or organism having a single set of chromosomes.

diploid : the quality of having two sets of chromosomes.

"Sexually reproducing organisms are diploid" (having two sets of chromosomes, one from each parent)

eukaryotes : Organisms whose cells have a nucleus enclosed within a nuclear envelope.

gamete : A mature sexual reproductive cell, as a sperm or egg, that unites with another cell to form a new organism. A haploid cell that fuses with another haploid cell during fertilization in organisms that sexually reproduce. A mature haploid male or female germ cell which is able to unite with another of the opposite sex in sexual reproduction to form a zygote.

gamete (Ancient Greek): to marry

zygote : A eukaryotic cell formed by a fertilization event between two gametes.

zygōtos (Greek): joined. yoked.

monozygotic : Monozygotic (MZ) or identical twins occur when a single egg is fertilized to form one zygote (hence, "monozygotic") which then divides into two separate embryos.

"monozygotic twins"

empirical :

"generate results more consistent with empirical estimates"

genetic variants :

Q: A human cell containing 22 autosomes and a Y chromosome is a sperm.

Neural Networks for Genomic Prediction (paper) Pinto et al. (2019)

[4] https://www.merriam-webster.com/medical/allosome

Transcriptome Wide Association

2.4.2 Meetings

Jie Yuan meeting #2 (Sep 10)

- There's a dot product and its output is passed through some activation function like sigmoid or ReLU. It's still linear in the sense that there's some sort of function that takes in a dot product (which is the definition of a linear operation). You can think of [the betas] as the weights in the neural network because the betas are the coefficients of a linear model.
- Liability is discussed in the multi-locus models paper.
- In the genomics context, this is called the **liability threshold model**. If you google that, you'll find some papers. In a broader machine learning sense, this is called the **probit regression model**. WIkipedia probably has a sufficient article on it.
- The gist of it is that it's a model to map continuous sums (the dot products basically) into binary labels: cases and controls. This can be any sort of logistic regression type thing where you have 0s and 1s. Basically, the **probit model is an alternative to the logistic regression model**.
- The liability is related to what's called the "link function". The link function in the probit model is the normal CDF function. There's a term that goes into the link function, and that term is what we're calling liability.
- The liability is basically the product of the vector of genotypes and the vector of betas. It's the linear part of the generalized linear model.
- Once you have the liability and plug it into the link function. Here, the link function is the probit regression. In a neural network, this would be the linear activation function (sigmoid, ReLU, etc.).
- You know how a sigmoid has a domain that's $(-\infty, \infty)$? It's range is (0,1), so what that function does is map a real value into being a probability. It gives you the probability of being a case. If you look at the normal CDF function, you find that it looks almost the same.

Why is the liability threshold model better than or different from the logistic reg model?

- One reason people use log reg more often is that the betas from log reg are interpretable. The have a meaning in terms of odds ratios. In a log reg model, the effect sizes are the natural log of the odds ratios.
- One disadvantage of log reg compared to liability-threshold is that log reg doesn't have a concept of the underlying distribution of the liability. In log reg, you get the linear $X\beta$ term, but there's no sort of distribution around it. In the liability-thresh model, we say that the liability, itself, is standard normally distributed. By taking the normal CDF, what you're doing is mapping that $X\beta$ value, the liability value, onto this distribution and

asking, “what’s the probability that it’s larger than some threshold?"

- larger than threhsold $\implies$ label as 1, smaller $\implies$ 0.
- Because liability-threshold has that normal distribution, it gives you more concepts to play with such as variance explained, which the logistic reg model doesn’t have.
- liability threshold model in simple terms

  from wikipedia: Liability threshold model

schizophrenia (SZ) example:

- Whether or not someone has SZ comes from a combination of factors in both their genes and environment.
- There are all sorts of variables. Imagine hypothetically that you could collect all of them and produce a score from that (that determines whether or not you have SZ). That score would be the liability.
- The liability is a $\mathcal{N}(0, 1)$ distribution that, if higher than the threshold, predicts/indicates the patient has SZ.
- The problem with that is that we don’t actually observe everything. We don’t truly observe every factor that goes into whether you have SZ b/c (1) we can’t measure one’s environment and (2) we can’t know all of the genomic risk factors from GWAS. We can only identify the largest risk factors.

  This means we are identifying only a small fraction of what goes into genetic risk for SZ.
- Essentially, what we have is a small subset of the factors that actually determine your SZ risk (which we’re reasonably confident indicate this relationship). In this example, some subset of the risk factors like the genomic variance which increase your risk for SZ.
- What that means is that inside this standard normal liability distribution/function, what we actually know is a small subset of the factors that go into that. If we score people on that subset of factors, what we get is a smaller distribution, a distribution that has a variance that’s a small fraction of the total liability variance, which we can’t observe. The variance of that small fraction that we observe, that’s the **variance explained**.
-

Bibliography

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Pinto, C., Gill, M., Heron, E. A., et al. (2019). Can artificial neural networks supplant the polygene risk score for risk prediction of complex disorders given very large sample sizes? *arXiv preprint arXiv:1911.08996*.

Raghu, M. and Schmidt, E. (2020). A survey of deep learning for scientific discovery. *arXiv preprint arXiv:2003.11755*. https://arxiv.org/pdf/2003.11755.pdf.

Singh, K., Sandhu, R. K., and Kumar, D. (2015). Comment volume prediction using neural networks and decision trees. In *IEEE UKSim-AMSS 17th International Conference on Computer Modelling and Simulation, UKSim2015 (UKSim2015)*, Cambridge, United Kingdom.

Wray, N. R. and Goddard, M. E. (2010). Multi-locus models of genetic risk of disease. *Genome Medicine*, 2(2):10.