Machine Learning
Synthesis

Rajesh Ranganath
What did we learn?
Machine Learning

\[ p(y \mid x) \]

[Image of code from Atlantic]
Supervised Learning

Goal: Learn how to predict $y$ from $x$
Goal: Learn how to predict $y$ from $x$
Goal: Learn how to predict $y$ from $x$
Supervised Learning

- \( x \) are feature or covariates

- \( y \) is a real number to be predicted

Observe \( n \) samples

\[
(x_1, y_1), \ldots, (x_n, y_n)
\]
Overfitting vs Underfitting
Regularization

Figure 3.4 Plot of the contours of the unregularized error function (blue) along with the constraint region (3.30) for the quadratic regularizer $q = 2$ on the left and the lasso regularizer $q = 1$ on the right, in which the optimum value for the parameter vector $w$ is denoted by $w^\star$.

The lasso gives a sparse solution in which $w^\star_1 = 0$.

For the remainder of this chapter we shall focus on the quadratic regularizer (3.27) both for its practical importance and its analytical tractability.

3.1.5 Multiple outputs

So far, we have considered the case of a single target variable $t$. In some applications, we may wish to predict $K > 1$ target variables, which we denote collectively by the target vector $t$. This could be done by introducing a different set of basis functions for each component of $t$, leading to multiple, independent regression problems.

However, a more interesting, and more common, approach is to use the same set of basis functions to model all of the components of the target vector so that

$$y(x, w) = W^T \phi(x)$$

(3.31)

where $y$ is a $K$-dimensional column vector, $W$ is an $M \times K$ matrix of parameters, and $\phi(x)$ is an $M$-dimensional column vector with elements $\phi_j(x)$, with $\phi_0(x) = 1$ as before. Suppose we take the conditional distribution of the target vector to be an isotropic Gaussian of the form

$$p(t|x, W, \beta) = N(t|W^T \phi(x), \beta^{-1}I)$$

(3.32)

If we have a set of observations $t_1, \ldots, t_N$, we can combine these into a matrix $T$ of size $N \times K$ such that the $n$th row is given by $t_T n$. Similarly, we can combine the input vectors $x_1, \ldots, x_N$ into a matrix $X$. The log likelihood function is then given by

$$\ln p(T|X, W, \beta) = N \sum_{n=1}^{N} \ln N(t_n|W^T \phi(x_n), \beta^{-1}I) = NK \frac{1}{2} \ln (\beta^2 \pi) - \beta^2 N \sum_{n=1}^{N} ||t_n - W^T \phi(x_n)||^2.$$ (3.33)
Regularization

Figure 3.4 Plot of the contours of the unregularized error function (blue) along with the constraint region (3.30) for the quadratic regularizer $q_2$ on the left and the lasso regularizer $q_1$ on the right, in which the optimum value for the parameter vector $w$ is denoted by $w^\star$.

The lasso gives a sparse solution in which $w^\star_1 = 0$.

For the remainder of this chapter we shall focus on the quadratic regularizer (3.27) both for its practical importance and its analytical tractability.

3.1.5 Multiple outputs

So far, we have considered the case of a single target variable $t$. In some applications, we may wish to predict $K > 1$ target variables, which we denote collectively by the target vector $t$. This could be done by introducing a different set of basis functions for each component of $t$, leading to multiple, independent regression problems.

However, a more interesting, and more common, approach is to use the same set of basis functions to model all of the components of the target vector so that

$$y(x, w) = W^T \phi(x)$$

(3.31)

where $y$ is a $K$-dimensional column vector, $W$ is an $M \times K$ matrix of parameters, and $\phi(x)$ is an $M$-dimensional column vector with elements $\phi_j(x)$, with $\phi_0(x) = 1$ as before. Suppose we take the conditional distribution of the target vector to be an isotropic Gaussian of the form

$$p(t|x, W, \beta) = N(t|W^T \phi(x), \beta^{-1}I)$$

(3.32)

If we have a set of observations $t_1, \ldots, t_N$, we can combine these into a matrix $T$ of size $N \times K$ such that the $n$th row is given by $t_n^T$. Similarly, we can combine the input vectors $x_1, \ldots, x_N$ into a matrix $X$.

The log likelihood function is then given by

$$\ln p(T|X, W, \beta) = N \sum_{n=1}^{N} \ln N (t_n|W^T \phi(x_n), \beta^{-1}I) = NK \ln (\beta^2 \pi) - \beta^2 \sum_{n=1}^{N} \|t_n - W^T \phi(x_n)\|^2.$$ 

(3.33)
Generalized Linear Models

\[ p(y_i \mid x_i) \sim \text{Expfam} \]
Generalized Linear Models

\[ p(y_i | x_i) \sim \text{Expfam} \]

The parameter is the natural parameter. One option

\[ \eta_i = \theta^\top x_i \]
Generalized Linear Models

\[ p(y_i | x_i) \sim \text{Expfam} \]

The parameter is the natural parameter. One option

\[ \eta_i = \theta^\top x_i \]

Learn parameters by maximizing likelihood

\[ \mathcal{L}(\theta) = \sum_i \log(h(y_i)) \exp(\eta_i^\top t(y_i) - a(\eta_i))) \]
Generalized Linear Models

\[ p(y_i | x_i) \sim \text{Expfam} \]

The parameter is the natural parameter. One option

\[ \eta_i = \theta^\top x_i \]

Learn parameters by maximizing likelihood

\[ \mathcal{L}(\theta) = \sum_i \log(h(y_i)) \exp(\eta_i^\top t(y_i) - a(\eta_i))) \]

Parameters can be generalized with link functions
Learning Nonlinear Functions with Trees

[Wikipedia, Decision_tree_learning]
Random forests

- Don’t really overfit with more computational steps
- Embarrassingly parallel training
- Need to randomly find the right input region

Gradient Boosted Trees

- Overfits with more computational steps
- Sequential training
- Uses gradients to focus on right regions
How do we use this?

\[ \nabla_{W_{\ell}} \mathcal{L} = \sum_{i=1}^{n} \nabla_{h_{i,\ell}} \mathcal{L} \nabla_{W} h_{i,\ell} \]

and

\[ \nabla_{b_{\ell}} \mathcal{L} = \sum_{i=1}^{n} \nabla_{h_{i,\ell}} \mathcal{L} \nabla_{b} h_{i,\ell} \]

Derivatives are sums; we can use stochastic optimization
Table 1 shows the results in ImageNet Classification. We first evaluate 18-layer and 34-layer plain networks. The 34-layer plain net is in Fig. 3.4. Implementation. Based on the above plain network, we build a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions.

Right: a residual network with 34 parameter layers (3.6 billion FLOPs). The identity shortcut is a 3x3 convolution (see Eqn. (3)) can be directly used when the input and output have the same dimensions (done by 1x1 convolutions). For both shortcuts (Eqn. (1)) can be directly used when the input and output have the same dimensions (done by 1x1 convolutions). For both shortcuts, right) which turn the input feature map into its counterpart residual version. The identity shortcut (Eqn. (1)) right after each convolution and before activation, following [3]. We use a weight decay of 0.0001 and a momentum of 0.9. We use SGD with a mini-batch size of 256. The learning rate starts from 0.1 and is divided by 10 when the error plateaus, following [3]. We initialize the weights at normal color augmentation in [21]. The image is resized with its shorter side randomly sampled in [256, 480] [21] that consists of 1000 classes. The models are trained on the 1.28 million training images, and evaluated on the 50k validation images. We also obtain a final result on the 100k test images, reported by the test server.
Synthesis: Can’t beat Information Theory

Given data from $p(y, x)$, best predictions

$$p(y|x)$$

If given lots of data, can’t do better. When thinking about methods in practice, ask where does the information come from?

- Semi-Supervised learning?
- Initializing image models in one domain with another?
Joint/Unsupervised Modeling

Is there any difference between $x$ and $y$ with $p(x,y)$?
Joint/Unsupervised Modeling

Is there any difference between $x$ and $y$ with $p(x,y)$?

*All are just variables?*
Neuroscience

[Manning+ 2014]
Abstract

We present a new, fully generative model of optical telescope image sets, along with a variational procedure for inference. Each pixel intensity is treated as a Poisson random variable, with a rate parameter dependent on latent properties of stars and galaxies. Key latent properties themselves random, with scientific prior distributions constructed from large ancillary data sets. We check our approach on synthetic images. We also run it on images from a major sky survey, where it exceeds the performance of the current state-of-the-art method for locating celestial bodies and measuring their colors.

1. Introduction

This paper presents Celeste, a new, fully generative model of astronomical image sets—the first such model to be empirically investigated, to our knowledge. The work we report is an encouraging example of principled statistical inference applied successfully to a science domain underserved by the machine learning community. It is unfortunate that astronomy and cosmology receive comparatively little of our attention: the scientific questions are fundamental, there are petabytes of data available, and we as a data-analysis community have a lot to offer the domain scientists. One goal in reporting this work is to raise the profile of these problems for the machine-learning audience and show that much interesting research remains to be done.

Turn now to the science. Stars and galaxies radiate photons. An astronomical image records photons—each originating from a particular celestial body or from background atmospheric noise—that pass through a telescope's lens during an exposure. Multiple celestial bodies may contribute photons to a single image (e.g. Figure 1), and even to a single pixel of an image. Locating and characterizing the imaged celestial bodies is an inference problem central to astronomy. To date, the algorithms proposed for this inference problem have been primarily heuristic, based on finding bright regions in the images (Lupton et al., 2001; Stoughton et al., 2002).

Generative models are well-suited to this problem—for three reasons. First, to a good approximation, photon counts from celestial objects are independent Poisson processes: each star or galaxy has an intrinsic brightness that
Figure S2: Population structure inferred from the TGP data set using the T\textsc{eras}\textsc{structure} algorithm at three settings for the number of populations $K$. The visualization of the ✓'s in the Figure shows patterns consistent with the major geographical regions. Some of the clusters identify a specific region (e.g. red for Africa) while others represent admixture between regions (e.g. green for Europeans and Central/South Americans). The presence of clusters that are shared between different regions demonstrates the more continuous nature of the structure. The new cluster from $K=7$ to $K=8$ matches structure differentiating between American groups. For $K=9$, the new cluster is unpopulated.

[Peer-reviewed] is the author/funder. All rights reserved. No reuse allowed without permission. This preprint (which was not peer-reviewed) is the author/funder. All rights reserved. No reuse allowed without permission.

http://dx.doi.org/10.1101/013227


[Gopalan+ 2016]
Formal definition

Probabilistic generative models:

- Data: \( x \)
- Hidden Structure (latent variables): \( z \)
- Model: \( p(x, z) = p(z)p(x | z) \) - prior \( p(z) \) and likelihood \( p(x | z) \)
Variational Inference

Posit a family of distributions $q(z; \lambda)$ indexed with parameter $\lambda$
Variational Inference

Find $\lambda$ such that $q$ is close to $p(z|x)$
Variational Inference

Closeness measured by the KL divergence
Synthesis: How are supervised and unsupervised learning related?

Supervised learning

\[ p(y|x) \]

Unsupervised learning

\[ p(x) \]

Both just distribution estimation
Synthesis: How are supervised and unsupervised learning related?

Supervised learning

\[ p_\theta(y|x) \]

Unsupervised learning

\[ p_\theta(x) \]

Both just distribution estimation!
Synthesis: How are supervised and unsupervised learning related?

What if we have

$$p_\theta(x_i | x_{-i})$$

Can sample from $p_\theta(x)$ by

$$x_{t,i} \sim p_\theta(x_i | x_{t-1,-i})$$

For big $t$.

*Joint defined by repeated supervised learning*
Overfitting vs Underfitting

\[ M_{\text{big}} \subset M_{\text{small}} \]

\[ f^* \]
Synthesis: How are supervised and unsupervised learning related?

Overfitting and Underfitting Problem in Both!

- **Supervised learning. Powerful model:**
  - Outputs $y_i$ for training $x_i$
  - Ill defined elsewhere

- **Unsupervised learning. Powerful model:**
  - Outputs $\frac{1}{n}$ for training $x_i$
  - Outputs 0 elsewhere
Variational Autoencoder (VAE)

\[ p(z) = \text{Normal}(0, 1) \]

\[ p(x|z) = \text{Normal}(\mu_\beta(z), \sigma^2_\beta(z)) \]

\( \mu \) and \( \sigma^2 \) are deep networks with parameters \( \beta \).

[Kingma+ 2014; Rezende+ 2014]
Deep generative model with Noise

- Try to model density directly

- $x \sim p_\theta(x) \iff x = f(\theta, \epsilon)$
Adversarial Learning

Objective:

\[ \mathbb{E}_{x \sim p_{\text{data}}} [\log D(x)] + \mathbb{E}_{x = f(\theta, \epsilon)} [\log (1 - D(x))] \]

Goal:

\[
\min_{\theta} \max_{\phi} \mathbb{E}_{x \sim p_{\text{data}}} [\log D(x)] + \mathbb{E}_{x = f(\theta, \epsilon)} [\log (1 - D(x))]
\]

Gradients:

\[
\nabla_\phi \mathcal{L} = \mathbb{E}_{x \sim p_{\text{data}}} [\nabla_\phi \log D(x)] + \mathbb{E}_{x = f(\theta, \epsilon)} [\nabla_\phi \log (1 - D(x))]
\]

\[
\nabla_\theta \mathcal{L} = \mathbb{E}_{x = f(\theta, \epsilon)} [\nabla_x \log (1 - D(x)) \nabla_\theta f(\theta, \epsilon)]
\]
Synthesis: Deep Methods are plug-and-play function approximators

General supervised learning:

\[ y = f(\epsilon, x), \quad \epsilon \sim s \]

General generative modeling

\[ x = f(\epsilon), \quad \epsilon \sim s \]

_Deep methods base f on neural networks_
Synthesis: Can’t escape information theory

- Lot’s of unlabeled data doesn’t inherently improve classification
- More bits of information
- More questions need to be answered
Interventions in Healthcare
Interventions in Healthcare
Question: Should we give medication A or medication B?
Causal Inference

Causal inference seeks to estimate the effect of an intervention

- Which statin to give to a patient with hyperlipidemia?

- Which medication to give to a depressed person?

- Which patients to give hospice care to?
Strong Ignorability

\[ E[y_1] - E[y_0] = E_x[E[y_1 | x]] - E_x[E[y_0 | x]] \]
\[ = E_x[E[y_1 | x, t = 1]] - E_x[E[y_0 | x, t = 0]] \]

Can estimate from observed data!

For fixed covariates \( x \)

- \( E[y_1 | x, t = 1] \) is the average treated outcome on treated
- \( E[y_0 | x, t = 0] \) is the average untreated outcome on untreated
Computational Tool: Regression

Goal: Directly estimate $\mathbb{E}[y \mid x, t]$

Direct averages may require too many samples. Use regression

$$\argmin_f \mathbb{E}[(y - f(x, t))^2] = \mathbb{E}[y \mid x, t]$$

Make $f$ parametric $f_\theta$ and solve

$$\argmin_\theta \mathbb{E}[(y - f_\theta(x, t))^2]$$

Substitute back into ATE

$$ATE = \mathbb{E}_x[\mathbb{E}[y_1 \mid x, t = 1]] - \mathbb{E}_x[\mathbb{E}[y_0 \mid x, t = 0]]$$

$$= \mathbb{E}_x[f_\theta(x, 1)] - \mathbb{E}_x[f_\theta(x, 0)]$$

Why not two separate regressions?
Computational Tool: Propensity Scores

- From strong ignorability $x$ predicts treatment chance
- Treatment chance has enough information to break dependence

Build a treatment chance model with regression

$$p_\phi(t = 1 | x)$$

Use importance sampling to compute expectations
Synthesis: Identification, Then Supervision

- Causal inference needs assumptions
- Those assumptions turn problem into one of finding relationships
- Finding relationships between variables is supervised learning
Synthesis

Supervised learning without confounders is causal inference
Synthesis: Can’t escape information theory

\[ t = f(\epsilon, x), \]

If \( \epsilon \) has little entropy:

- can’t estimate complicated treatment functions
Synthesis: Deep Functions Replace Traditional Estimators

- Deep Propensity Scores: $p(t | x)$
- Deep Outcome Regression: $p(y | x, t)$
Question: How much of medication A should we give every week?
Markov Decision Processes

A Markov Decision Process

- **$S$:** A collection of states $s$
- **$A$:** A collection of actions $a$ to take in each state
- $p(s_{t+1} | a_t, s_t)$: State transition
- $r(r_{t+1} | s, s_{t+1}, a_t)$: Reward distribution
  
  Don’t really need to be random
- Policy Gradients
- Advantage Actor-Critic
- Value Iteration
- Q-learning
Synthesis: MDPs are Causal Inference over Time
Synthesis: Causal Inference (MDP) Easy When Control Over Environment

With massive data under experimental control, lots of stuff becomes easy
Synthesis: Can’t escape information theory

- Need to try an action many times to reduce variance
- Need large number of samples if actions chains are important
Synthesis: Everything is Reinforcement Learning

- Supervised learning: Reward could be classification accuracy
- Generative Modeling: Policy gradients
- Causal Inference: Causal inference over time
- Reinforcement Learning: Is reinforcement learning
Synthesis: If Everything is Reinforcement Learning, Why do Anything Else?
Things to Take Away From the Class

- Supervised Learning
- Unsupervised Learning
- Causal Inference
- Reinforcement Learning
Deep learning can be used anywhere a mapping is needed
Gradients power a lot of machine learning
Assumptions make things possible!
Can’t escape information theory. Ask where is the information coming from!
Questions?