Chapter 7

Model-Based Machine Learning for Transportation

Inon Peled, Filipe Rodrigues and Francisco Câmara Pereira
Department of Management Engineering, Technical University of Denmark (DTU), Lyngby, Denmark

Chapter Outline

1 Introduction **145**
1.1 Background Concepts 146
1.2 Notation 147
2 Case Study 1: Taxi Demand in New York City **147**
2.1 Initial Probabilistic Model: Linear Regression 147
2.2 Key Components of MBML 148
2.3 Inference 150
2.4 Model Improvements 152
3 Case Study 2: Travel Mode Choices **155**
3.1 Improvement: Hierarchical Modeling 157
4 Case Study 3: Freeway Occupancy in San Francisco **160**
4.1 Autoregressive Model 160
4.2 State-Space Model 160
4.3 Linear Dynamical Systems 161
4.4 Common Enhancements to LDS 162
4.5 Nonlinear Variations on LDS 164
5 Case Study 4: Incident Duration Prediction **165**
5.1 Preprocessing 166
5.2 Bag-of-Words Encoding 166
5.3 Latent Dirichlet Allocation 166
6 Summary **169**
6.1 Further Reading 169
References **170**

1 INTRODUCTION

The world around us is rich with **uncertainty**, possibly because of innate non-determinism (Koller and Friedman, 2009). In the transportation domain alone, one finds uncertainty in, e.g., road safety, train delays, bus dwell times, or passengers' choice of travel mode. Models of reality must therefore account for such prevalent uncertainty. Consequently, at the heart of modern Machine

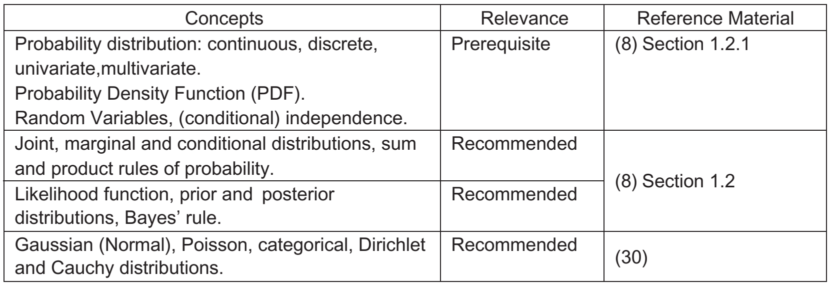
Learning is **probability theory**, which provides a consistent framework for quantifying and manipulating uncertainty.

This chapter describes the **Model-Based approach to Machine Learning (MBML)**, in which uncertainty is represented and manipulated through the coherent use of probability theory. When given data for modeling, the first step in the MBML approach is to define our assumptions about the stochastic process which generated the data. MBML is thus antithetical to traditional Machine Learning methods, where modeling begins with the dilemma of picking an algorithm to learn patterns in the data (Olson et al., 2017), e.g., among the algorithms in Chapters 2 and 3. Furthermore, MBML provides a mathematically grounded framework for measuring how uncertainty changes when new observations become available.

After a short review of preliminary concepts, this chapter introduces MBML through four case studies, which demonstrate how MBML elegantly treats a wide range of common modeling problems. The first case study deals with a **continuous regression problem** and introduces the key components of MBML. It also defines **Bayesian inference**, through which we measure the certainty gain from observed evidence. The second case study deals with a **discrete classification problem**. The third case study handles a **time series problem** through the large class of State-Space Models (SSM). The fourth and last case study introduces **topic modeling** for numerically encoding **textual data** in regression problems. Each case study starts by describing an actual data set, then detaches from specific data details, and proceeds to build useful models for the general problem domain.

1.1 Background Concepts

We assume the reader is familiar with fundamentals of linear algebra, including matrix-vector product (Multiplying matrices and vectors—Math Insight, 2018), as well as with basic concepts and notation of probability theory, as instructed in Fig. 1. We also recommend reading Chapters 2 and 3 for better familiarity with fundamentals of Machine Learning and common terminology.



Concepts	Relevance	Reference Material
Probability distribution: continuous, discrete, univariate,multivariate. Probability Density Function (PDF). Random Variables, (conditional) independence.	Prerequisite	(8) Section 1.2.1
Joint, marginal and conditional distributions, sum and product rules of probability.	Recommended	(8) Section 1.2
Likelihood function, prior and posterior distributions, Bayes' rule.	Recommended	
Gaussian (Normal), Poisson, categorical, Dirichlet and Cauchy distributions.	Recommended	(30)

FIG. 1 Probability theory concepts used in this chapter.

1.2 Notation

This chapter uses the following mathematical notation. Letters which are not bold (e.g., y, D or σ) denote scalars. A lower case bold letter (e.g., $\mathbf{x}$ or $\boldsymbol{\beta}$) denotes a vector, while an upper case bold letter (e.g. $\mathbf{X}$ or $\boldsymbol{\Sigma}$) denotes a matrix. $\mathbf{0}$ and $\mathbf{I}$ denote, respectively, a zero square matrix and an identity matrix, with dimensions determined by context.

2 CASE STUDY 1: TAXI DEMAND IN NEW YORK CITY

As a first case study, we model taxi demand in New York City. Such a model is useful for, e.g., optimizing taxi services, so that taxis can be proactively dispatched to locations where high demand is anticipated. More generally, this model is applicable to various problems of demand modeling, such as public or shared transport, or allocation of resources, such as water, energy, or communication. We use this case study to also introduce some of the key components of MBML.

Our data consist of two data sets: taxi pickups around Wall Street in years 2009–2015 (NYC Open Data, 2018), and corresponding weather information from the National Oceanic and Atmospheric Administration (National Oceanic and Atmospheric Administration | U.S. Department of Commerce, 2018), such as wind speed, pressure, and precipitation. The data are given as a matrix, where each row consists of hourly aggregated pickups and weather information. Each of the $N = 8760$ matrix rows consists of $L = 16$ **data features** (date, hour of day, and weather), followed by the corresponding value of the **target variable** (number of taxi pickups).

Our goal is to model the dependency of the target variable on the data features. We thus partition the data matrix as a submatrix $\mathbf{X}^{N\times L}$ of data features, and a column vector $\mathbf{y}^{N\times 1} = (y_1, \ldots, y_N)^T$ of corresponding pickup counts. We next assume that for all $n = 1\ldots N$, y_n depends on $\mathbf{x}_n$, the n'th row of $\mathbf{X}$, as

$$y_n = f(\mathbf{x}_n) + \varepsilon \tag{1}$$

where f is an unknown function, and ε is error ("noise") in the observations. This case study thus demonstrates a **regression problem**, where we wish to find a function f that models the dependency (Eq. 1) well, as explained also in Chapter 2.

2.1 Initial Probabilistic Model: Linear Regression

Linear Regression is a common starting point in regression problems, and serves as a simple foundation for building powerful models. Our initial model M_1 is a linear regression model, whereby we assume there exists a vector of coefficients $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_L)^T \in \mathbb{R}^L$, so that

$$y_n = \boldsymbol{\beta}^T \mathbf{x}_n + \varepsilon$$

Note that we chose not to include a **zero-intercept ("bias")** coefficient β_0 in our model. If desired, we can nevertheless include β_0 along with a corresponding data feature, which is fixed as 1.

2.1.1 Likelihood Function

Next, we assume that ε is **white noise**, i.e., ε is normally distributed as $\varepsilon \sim \mathcal{N}(\varepsilon | 0, \sigma^2)$ for some unknown $\sigma \in \mathbb{R}$. This too is a common initial assumption. Equivalently, we have modeled $\mathbf{y}$ as a random variable, so that

$$y_n \sim \mathcal{N}\left(y_n | \boldsymbol{\beta}^T \mathbf{x}_n, \sigma^2\right)$$

As such, we assume that all observations are independent samples from a multivariate Gaussian, which has mean $\boldsymbol{\beta}^T \mathbf{X}$ and variance σ^2. For any given value of $\boldsymbol{\beta}$ and σ, the mean and variance are thus known and fixed, and the **likelihood** of obtaining the particular observations $\mathbf{y}$ is

$$p(\mathbf{y} | \mathbf{X}, \boldsymbol{\beta}, \sigma) = \prod_{n=1}^{N} \mathcal{N}\left(y_n | \boldsymbol{\beta}^T \mathbf{x}_n, \sigma^2\right) \tag{2}$$

where $\mathcal{N}\left(y_n | \boldsymbol{\beta}^T \mathbf{x}_n, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\left(y_n - \boldsymbol{\beta}^T \mathbf{x}_n\right)^2}{2\sigma^2}}$ is the Probability Density Function (PDF) of the Normal distribution.

2.1.2 Priors

Let us now assume for simplicity that σ is known in advance in our model, so that we are only interested in learning parameter $\boldsymbol{\beta}$, the linear coefficients. MBML invites us to encode any prior knowledge we have about the model parameters as a **prior** distribution. We thus assign $\boldsymbol{\beta}$ a prior distribution, as

$$\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\beta} | \mathbf{0}, \lambda \mathbf{I})$$

That is, our prior belief is that each linear coefficient is distributed identically and independently ("i.i.d") as a Gaussian with mean zero and variance λ, which is now an additional component of our model. For simplicity, we fix λ, carefully choosing its value: too large variance renders the prior distribution ineffective, which leads to overfitting, whereas too low variance overly restricts the coefficients, which leads to underfitting. This is an instance of **regularization** for prevention of overfitting and underfitting (Ng, 2004).

2.2 Key Components of MBML

Our initial, probabilistic linear regression model M_1 is now complete. As we shall soon explain, our goal now is to obtain the **posterior distribution** of model parameters, namely their distribution after ("a-posteriori") observing the evidence $\mathbf{y}$. MBML facilitates this goal through three key components: (1) generative process, (2) Probabilistic Graphical Model (PGM), and (3) factorized joint probability distribution, all of which we describe next.

2.2.1 Generative Process

We have established our assumptions on the structure and operation of M_1 by reasoning about a possible chain of random events, which yielded the observations $\mathbf{y}$ from the data features $\mathbf{X}$. That is, we imagined a **generative process**, through which the given observations manifested. Establishing a generative process clarifies our initial assumptions about the structure of uncertainty in the problem, and is therefore a first step in designing any MBML model. Fig. 2 summarizes the generative process of M_1 in concise, algorithmic form.

2.2.2 Probabilistic Graphical Model

PGMs describe the structure of probabilistic models intuitively and compactly. A PGM is a graph, where nodes represent random variables, and edges represent probabilistic relationship. In this chapter, we will use only directed PGMs, also known as Bayesian networks, where edge directions represent causality (Jensen, 1996).

Let us describe the **components of PGMs** by reviewing Fig. 3A, which provides a PGM for M_1. Large circles are nodes, annotated with the corresponding variables. Nodes for observed variables are shaded, whereas nodes for unobserved ("latent") variables are unshaded. The plate marked with N stands for N repeated applications, i.e., it describes the relationship between $\mathbf{x}_n$ and y_n for all $n = 1\ldots N$. Small black circles represent prefixed (hyper)parameters, which are sometimes not shown, to reduce visual clutter. Note that a PGM does not specify the underlying probability distributions, hence the same PGM may apply to many different probabilistic models.

```
Given $\mathbf{X}, \sigma, \lambda$:
    1. Draw $\boldsymbol{\beta} \sim N(\mathbf{0}, \lambda\mathbf{I})$.
    2. For each feature vector $\mathbf{x}_n$, $n = 1..N$:
        a. Draw target $y_n \sim \mathcal{N}(\boldsymbol{\beta}^T\mathbf{x}_n, \sigma^2)$.
```

FIG. 2 Generative process for linear regression model M_1.

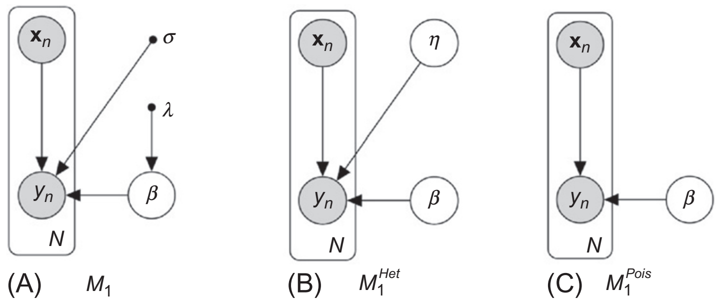


FIG. 3 Probabilistic Graphical Models for case study 1: (A) homoscedastic linear regression, (B) heteroscedastic linear regression, (C) Poisson regression. Hyperparameters are not shown in (B) and (C).

2.2.3 Joint Probability Distribution

Through the sum rule of probability (Bishop, 2008), the **joint probability distribution** of all variables in a probabilistic model can yield the marginal distribution of any subset of these variables. Therefore, the joint probability distribution is the central object for calculating the updated state of uncertainty, given evidence. It is thus beneficial to **factorize** the joint probability distribution as much as possible. For M_1, the joint probability distribution of $\boldsymbol{\beta}$ and $\mathbf{y}$, given all known variables, factorizes as following.

$$\begin{aligned} p(\boldsymbol{\beta}, \mathbf{y} \mid \mathbf{X}, \sigma, \lambda) &= p(\boldsymbol{\beta} \mid \mathbf{X}, \sigma, \lambda) p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\beta}, \sigma, \lambda) \\ &= p(\boldsymbol{\beta} \mid \lambda) p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\beta}, \sigma) \\ &= \underbrace{\mathcal{N}(\boldsymbol{\beta} \mid \mathbf{0}, \lambda \mathbf{I})}_{prior} \underbrace{\prod_{n=1}^{N} \mathcal{N}\left(y_n \mid \boldsymbol{\beta}^T \mathbf{x}_n, \sigma^2\right)}_{likelihood} \end{aligned} \tag{3}$$

The first line in Eq. (3) uses the product rule of probability. The second line follows from the dependency structure in M_1, as seen in the *PGM* in Fig. 3A. The third line follows from the *generative story* we chose for M_1, as in Fig. 2.

Eq. (3) thus demonstrates several aspects common to all MBML models. We see that the three key components of MBML—PGM, generative story, and joint probability distribution—together yield the factorization of a centrally important quantity. This factorization is a **product of conditionally independent** factors, which reflects the relationship between distributions as dictated by Bayes' rule:

$$\text{posterior} \propto \text{prior} \times \text{likelihood} \tag{4}$$

where $\propto$ indicates that both sides are equal up to a multiplicative constant.

To recap, in model M_1 of taxi pickups, the posterior distribution of the latent linear coefficients $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_L)^T$ follows from the prior and likelihood. We have defined the prior on $\boldsymbol{\beta}$ as a Gaussian, and the likelihood of the observations as another Gaussian, which is parameterized on $\boldsymbol{\beta}$. The analytical forms of the prior and likelihood appear in Eq. (3), which provides the *joint* distribution of parameters $\boldsymbol{\beta}$ *together* with observations $\mathbf{y}$. However, we are now interested in the *marginal* posterior distribution of *only* $\boldsymbol{\beta}$, as we already know $\mathbf{y}$. Next, we describe methods for learning the latent parameters of any probabilistic model.

2.3 Inference

To learn the relationship between y and $\mathbf{X}$, we have formulated linear regression model M_1, through which we wish to learn coefficients $\boldsymbol{\beta}$. On one hand, we could use the **Frequentist approach**, whereby we obtain only point estimates for the parameters, e.g., the value of $\boldsymbol{\beta}$ which maximizes some probability of interest. For example, we could compute a Maximum Likelihood Estimate

(MLE), which maximizes Eq. (2), or a Maximum a-posteriori Estimate (MAP), which maximizes Eq. (3). Deterministic algorithms for computing MLE and MAP are described in, e.g., Murphy (2012).

On the other hand, point estimates have some drawbacks, e.g., MLE can lead to overfitting, because MLE is calculated to best fit the observed data. More generally, point estimates do not conserve *uncertainty* about model parameters. In contrast, we can retain a good measure of uncertainty about model parameters, if we infer their posterior *distribution*. E.g., for any value of $\boldsymbol{\beta}$ in M_1, the posterior distribution $p(\boldsymbol{\beta} \mid \mathbf{y})$ tells us how likely it is that the linear coefficients have that value, given the observed taxi pickups $\mathbf{y}$. Hence in MBML, *our goal is to perform* ***inference***: obtain the **posterior distribution** of model parameters.

2.3.1 Bayesian Inference

Bayesian Inference is a systematic way for inferring the posterior distribution of model parameters, given the prior and likelihood distributions, through **Bayes' rule**:

$$\overbrace{p(\boldsymbol{\beta} \mid \mathbf{y})}^{posterior} = \frac{\overbrace{p(\boldsymbol{\beta})}^{prior}\overbrace{p(\mathbf{y} \mid \boldsymbol{\beta})}^{likelihood}}{\underbrace{p(\mathbf{y})}_{evidence}} \tag{5}$$

For any value of $\boldsymbol{\beta}$, the nominator in Eq. (5) is typically easy to compute, because it consists of distributions which we explicitly selected. E.g., for model M_1, Eq. (3) defines this nominator in closed and tractable form. The evidence distribution $p(y)$ in the denominator is a **normalization constant**, equal to the sum (or integral) of the nominator over all possible values of $\boldsymbol{\beta}$. As such, the normalization constant is not necessarily tractable to compute, e.g., if its analytic form cannot be formulated, or if there are too many possible values of $\boldsymbol{\beta}$ for direct summation.

2.3.2 Exact vs Approximate Bayesian Inference

When the normalization constant in Eq. (5) is tractable, we can infer the desired left-hand side precisely through several **exact Bayesian inference** methods, see e.g., Jensen (1996). Because the prior and likelihood in M_1 are multivariate Gaussians, the normalization constant for M_1 is analytically tractable through properties of the Normal distribution, as discussed in Section 2.3 of Bishop (2008).

Generally though, the denominator in Eq. (5) is often intractable, in which case we can only obtain an approximation of the desired left-hand side. One way to perform **approximate Bayesian inference** is to collect **samples** from the distribution of the tractable nominator, and use the samples to approximate quantities related to the posterior distribution (Gilks, 2005). Another way is to search

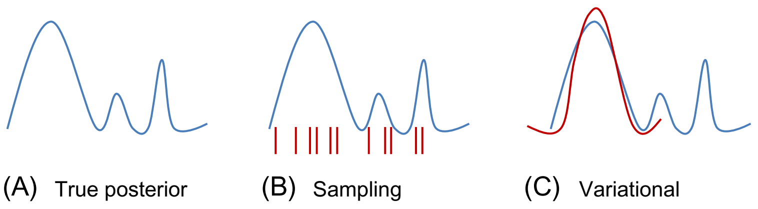


FIG. 4 An intractable posterior distribution (A) can be approximated either by sampling (B), or by optimizing a tractable proxy distribution (C).

for an analytically tractable **proxy distribution** for the true posterior distribution (Jordan et al., 1999). Fig. 4 illustrates both manners of approximate inference.

There exist multiple probabilistic programming tools which support approximate Bayesian inference (Gordon et al., 2014). One such popular tool is **Stan** (2018), and along this chapter, we provide Stan programs for some of the models we build. The reader may notice that each Stan program closely resembles the generative process for the model. Fig. 5 presents a Stan program for a model similar to M_1, except that σ is latent instead of known in advance.

2.4 Model Improvements

In our initial model M_1, the target variable $\mathbf{y}$ corresponds to taxi pickups, and is normally distributed as $\mathbf{y} \sim \mathcal{N}\left(\mathbf{y}|\beta^T \mathbf{X}, \sigma^2\right)$. Let us now present two separate model improvements, by enhancing the model with some domain-specific knowledge about our case study. These improvements also demonstrate the flexibility of the MBML approach, as the corresponding model changes are straightforward to apply.

2.4.1 Heteroscedasticity

Upon further inspection, we detect that the number of taxi pickups varies more wildly at some time intervals than others, e.g., there are nights which exhibit a

```
data { // What is already observed. This will be passed to the program as input.
    int<lower=0> N; // Number of feature vectors.
    int<lower=0> D; // Number of features.
    matrix[N, D] X; // Feature vectors.
    vector[N] y; // Target variables.
}
parameters { // What we wish Stan to infer.
    vector[D] beta; // Linear coefficients for mean of y as multivariate normal.
    real sigma; // Standard deviation for y as multivariate normal.
}
model { // Priors and likelihood.
    for (i in 1:D)
        beta[i] ~ normal(0, 10); // Prior.
    sigma ~ cauchy(0, 1); // Prior.
    y ~ normal(X * beta, sigma); // Likelihood. Note that the mean here is vector[N].
}
```

FIG. 5 Stan program for model M_1, with std. dev. σ as a latent variable rather than prefixed, and $\lambda = 10$. The choice of Cauchy prior is per (Gelman et al., 2008).

far wider range of taxi demand than usual. As such, the time series of pickups is **heteroscedastic**, namely its variance changes over time.

To account for heteroscedasticity, we shall model the noise as input-dependent. We could, e.g., model $\varepsilon_n \sim \mathcal{N}(\varepsilon_n | 0, \boldsymbol{\eta}^T \mathbf{x}_n)$ for each $n = 1 \dots N$, thus replacing the constant variance σ^2 with $\boldsymbol{\eta}^T \mathbf{x}_n$, where $\boldsymbol{\eta}$ is another vector of coefficients to be inferred. However, $\boldsymbol{\eta}^T \mathbf{x}_n$ might be negative for some $\mathbf{x}_n$, whereas variance is never negative. So instead, we model the noise as following:

$$\varepsilon_n \sim \mathcal{N}\left(\varepsilon_n | 0, \exp\left(\boldsymbol{\eta}^T \mathbf{x}_n\right)\right)$$

The exponent ensures that the variance is positive at all times, as $\exp: \mathbb{R} \to (0, +\infty)$. This is an example of using a **link function**, which in this case is the *log link*, because link functions are typically named after the inverse transformation. Equivalently, in the new heteroscedastic model ${M_1}^{Het}$, the target variables are distributed as

$$y_n \sim \mathcal{N}\left(y_n | \boldsymbol{\beta}^T \mathbf{x}_n, \exp\left(\boldsymbol{\eta}^T \mathbf{x}_n\right)\right)$$

Similarly to $\boldsymbol{\beta}$, our prior belief for $\boldsymbol{\eta}$ is a multivariate Gaussian, parameterized by some prefixed hyperparameter τ. The corresponding generative process for ${M_1}^{Het}$ is in Fig. 6.

Fig. 3B provides the PGM for ${M_1}^{Het}$. The corresponding joint probability distribution of $\mathbf{y}, \boldsymbol{\beta}, \boldsymbol{\eta}$ thus factorizes as

$$p(\mathbf{y}, \boldsymbol{\beta}, \boldsymbol{\eta} | \mathbf{X}, \tau, \lambda) = \underbrace{\mathcal{N}(\boldsymbol{\beta} | \mathbf{0}, \lambda \mathbf{I}) N(\boldsymbol{\eta} | \mathbf{0}, \tau \mathbf{I})}_{priors} \underbrace{\prod_{n=1}^{N} \mathcal{N}\left(y_n | \boldsymbol{\beta}^T \mathbf{x}_n, \exp\left(\boldsymbol{\eta}^T \mathbf{x}_n\right)\right)}_{likelihood}$$

Fig. 7 provides a Stan program for ${M_1}^{Het}$.

To compare the **predictive performance** of M_1 and ${M_1}^{Het}$, we first partition the data $\mathbf{X}$ and the observations $\mathbf{y}$ into two parts: a "**train set**" ($\mathbf{X}^{train}, \mathbf{y}^{train}$), and a "**test set**" ($\mathbf{X}^{test}, \mathbf{y}^{test}$). $\mathbf{y}^{train}$ consists of 5781 randomly chosen observations ($\sim$2/3 of $\mathbf{y}$), and $\mathbf{y}^{test}$ consists of the remaining 2979 observations ($\sim$1/3 of $\mathbf{y}$). Data vectors $\mathbf{X}^{train}, \mathbf{X}^{test}$ are the rows in $\mathbf{X}$ which correspond respectively to $\mathbf{y}^{train}$, $\mathbf{y}^{test}$. We then fit each of M_1, ${M_1}^{Het}$ on the train set, and run the trained models on $\mathbf{X}^{test}$ to obtain predictions.

```
Given $\mathbf{X}, \tau, \lambda$:
    1. Draw $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{0}, \lambda \boldsymbol{I})$.
    2. Draw $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \tau \boldsymbol{I})$.
    3. For each feature vector $\mathbf{x}_n$, $n = 1..N$:
            a. Draw target $y_n \sim \mathcal{N}(\boldsymbol{\beta}^T \mathbf{x}_n, \exp(\boldsymbol{\eta}^T \mathbf{x}_n))$.
```

FIG. 6 Generative process for heteroscedastic model ${M_1}^{Het}$.

```
data { ... // Same as for the initial model. }
parameters {
    vector[D] beta;
    vector[D] eta;
}
model {
    beta ~ cauchy(0, 10); // Vectorized, equivalent to beta priors in initial model.
    eta ~ cauchy(0, 10);
    y ~ normal(X * beta, exp(X * eta)); // Heteroscedastic likelihood.
}
```

FIG. 7 Stan program for model ${M_1}^{Het}$ with $\tau = 10$. Differences from initial model (Fig. 5) are highlighted in *bold*.

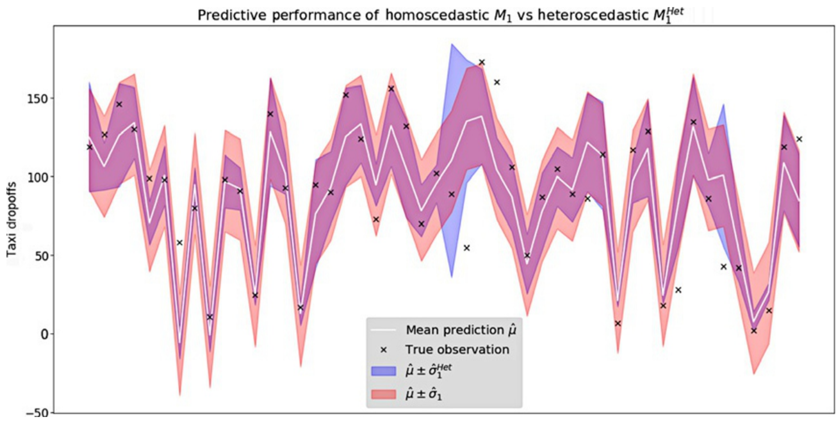


FIG. 8 Predictive performance of homoscedastic M_1 vs heteroscedastic ${M_1}^{Het}$, for 48 randomly chosen observations from the test set. Mean prediction $\hat{\mu}$ is nearly the same for both models, and is thus plotted only once. $\hat{\sigma}_1, \hat{\sigma}_1^{Het}$ are the prediction standard deviations of M_1, ${M_1}^{Het}$, respectively.

Fig. 8 illustrates the predictions of M_1 and ${M_1}^{Het}$ for a randomly selected subset of the test set. For each model, we plot a confidence interval as ± 1 standard deviation around the mean prediction. While the confidence intervals of both models capture most of the true observations, the varying std. dev. of ${M_1}^{Het}$ is often lower than the fixed std. dev. of M_1. Hence in this case study, the heteroscedastic model is often more confident than the homoscedastic model, while maintaining comparable accuracy. Finally, we see that both models can yield negative predictions, and address this issue in the next section.

2.4.2 Count Data

In our initial model M_1, the likelihood $\mathcal{N}\left(y_n | \boldsymbol{\beta}^T \mathbf{x}_n, \sigma^2\right)$ is continuous and can yield negative, noninteger draws. However, each target variable y_n in this case is actually a discrete count of taxi pickups, namely a nonnegative integer. We can thus obtain a better model by modeling y through some discrete distribution for **count data**, such as the Poisson distribution.

The **Poisson distribution** pertains to independent counts of occurrences within successive spatio-temporal intervals of fixed size, where the rate of occurrences is a known constant. We have already assumed that successive counts of taxi pickups are independent. However, a closer look at the data reveals that the mean number of pickups is unstable over time. We therefore choose to model $y_1, \ldots, y_N$ as differently distributed, so that each y_n is Poisson distributed according to the data at time n, namely

$$y_n \sim Poisson\left(\exp\left(\boldsymbol{\beta}^T \mathbf{x}_n\right)\right)$$

where the exponent again ensures that no rate is negative. We have thus obtained a more appropriate model ${M_1}^{Pois}$, which has a generative process as in Fig. 9.

The PGM for ${M_1}^{Pois}$ appears in Fig. 3C and is obtained by omitting the now unused hyperparameter σ from the PGM for M_1. Consequently, the joint probability distribution of $\boldsymbol{\beta}$ and $\mathbf{y}$ in ${M_1}^{Pois}$ factorizes as following.

$$p(\boldsymbol{\beta}, \mathbf{y} | \mathbf{X}, \sigma, \lambda) = \underbrace{\mathcal{N}(\boldsymbol{\beta} | \mathbf{0}, \lambda \mathbf{I})}_{prior} \underbrace{\prod_{n=1}^{N} \text{Poisson}\left(y_n | \exp\left(\boldsymbol{\beta}^T \mathbf{x}_n\right)\right)}_{likelihood} \tag{6}$$

where $Poisson\left(y_n | \exp\left(\boldsymbol{\beta}^T \mathbf{x}_n\right)\right) = \frac{\exp\left(y_n \boldsymbol{\beta}^T \mathbf{x}_n - \exp\left(\boldsymbol{\beta}^T \mathbf{x}_n\right)\right)}{y_n!}$ is the value of the Poisson PDF. We can derive the posterior distribution of ${M_1}^{Pois}$ only through approximate inference, because Eq. (6) yields an intractable normalization constant in Bayes' rule (5). A Stan program for ${M_1}^{Pois}$ appears in Fig. 10.

Finally, let us compare the **predictive performance** of ${M_1}^{Pois}$ and M_1, using the same train and test sets as we defined earlier. This time, we measure performance through some commonly used summary statistics of prediction error (Li, 2017), as defined in Fig. 11. The results appear in Fig. 12, which shows that Poisson regression outperforms linear regression in this case study.

3 CASE STUDY 2: TRAVEL MODE CHOICES

We have concluded the first case study by modeling the target variable as drawn from a discrete distribution. Let us now examine another case study, in which the target variable is discrete, although categorical rather than numeric. Here,

```
Given $\mathbf{X}$, $\lambda$:
    1. Draw $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{0}, \lambda \boldsymbol{I})$.
    2. For each feature vector $\mathbf{x}_n$, $n = 1..N$:
        a. Draw target $y_n \sim Poisson(\exp(\beta^T \mathbf{x}_n))$.
```

FIG. 9 Generative process for ${M_1}^{Pois}$.

```
data { ... // Same as for the initial model. }
parameters { // Unused parameter sigma is omitted.
    vector[D] beta;
}
model {
    beta ~ cauchy(0, 10);
    y ~ poisson_log(X * beta); // Poisson likelihood, using a log link function.
}
```

FIG. 10 Stan program for M_1^{Pois}. Differences from initial model M_1 (Fig. 5) are highlighted in *bold.*

Summary Statistic	Notation	Formula	Units	Comments
Mean absolute error	**MAE**	$\frac{1}{V}\sum_{v=1}^{V}\lvert y_v^{test}-\hat{y}_v\rvert$	Same as observations	Lower is better, 0 is perfect fit.
Rooted mean squared error	**RMSE**	$\sqrt{\frac{1}{V}\sum_{v=1}^{V}(y_v^{test}-\hat{y}_v)^2}$	Same as observations	Lower is better, 0 is perfect fit.
Coefficient of determination	$\boldsymbol{R^2}$	$1-\frac{\sum_{v=1}^{V}(y_v^{test}-\hat{y}_v)^2}{\sum_{v=1}^{V}\left(y_v^{test}-\overline{y^{test}}\right)^2}$	Unitless	Higher is better, 1 is perfect fit.

FIG. 11 Summary statistics for prediction errors, where y_1^{test}, …, y_V^{test} are V test observations, $\hat{y}_1,\ldots,\hat{y}_V$ are the corresponding predictions, and $\overline{y^{test}}=\left(y_1^{test}+\cdots+y_V^{test}\right)/V$ is the mean of the test observations.

Model	MAE	RMSE	R^2
M_1	19.292	25.879	0.645
M_1^{Pois}	**18.128**	**24.711**	**0.676**

FIG. 12 Prediction performance of M_1^{Pois} vs M_1. The best value in each column is highlighted in *bold.*

we wish to model **discrete choice of travel mode**, which takes one of distinct values: bus, train, car, or plane.

The data set is a travel diary, which consists of $N=394$ trips by $G=80$ different individuals. For each trip $n=1\ldots N$, the data features $\mathbf{x}_n$ are properties of the trip—e.g., cost and duration—and possibly also demographic information about the individual g_n who took the trip. The target variable y_n for each trip is the travel mode, which g_n chose among the $K=4$ abovementioned options.

Our goal is to model how each individual chooses the travel mode for a trip, given the trip properties. Such a model has many possible applications, e.g., understanding of human behavior, planning pricing policies, or incentivizing the usage of "green" transportation. Because the target variable in this model takes one of a finite number of categories ("classes"), this case study is an example of a **classification problem**.

As in the previous case study, our initial model M_2 is a **Generalized Linear Model (GLM)**, i.e., at the core of M_2 are linear combinations of the data features. We begin by numbering the travel modes ("classes") arbitrarily as $1 \ldots K$, and assign for each class $k = 1 \ldots K$ a parameter vector $\boldsymbol{\beta}_k$ with as many elements as there are data features. Our goal is thus to infer the posterior distribution of model parameters $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K$.

We finish building M_2 by modeling each y_n as a draw from a **categorical distribution (*Cat*)**. That is, each class $k = 1 \ldots K$ has some probability $p_n{}^{(k)}$ of being drawn as the n'th observation. To model these probabilities, we use **logistic regression**, as discussed in Chapter 1, so that for all $k = 1 \ldots K$:

$$p_n^{(k)} = \left(\text{Softmax}\left(\boldsymbol{\beta}_1^T \mathbf{x}_n, \ldots, \boldsymbol{\beta}_K^T \mathbf{x}_n\right)\right)_k \triangleq \frac{\exp\left(\boldsymbol{\beta}_k^T \mathbf{x}_n\right)}{\exp\left(\boldsymbol{\beta}_1^T \mathbf{x}_n\right) + \cdots + \exp\left(\boldsymbol{\beta}_K^T \mathbf{x}_n\right)}$$

Hence for model M_2, the generative process is as in Fig. 13A, where λ is a prefixed hyperparameter. The corresponding PGM for M_2 appears in Fig. 13B.

The joint probability distribution for M_2 thus factorizes as

$$p(\mathbf{y}, \boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K \mid \mathbf{X}, \lambda) = \underbrace{\left(\prod_{k=1}^{K} \mathcal{N}(\boldsymbol{\beta}_k \mid \mathbf{0}, \lambda \mathbf{I})\right)}_{priors} \underbrace{\prod_{n=1}^{N} Cat\left(y_n \mid \text{Softmax}\left(\boldsymbol{\beta}_1^T \mathbf{x}_n, \ldots, \boldsymbol{\beta}_K^T \mathbf{x}_n\right)\right)}_{likelihood} \tag{7}$$

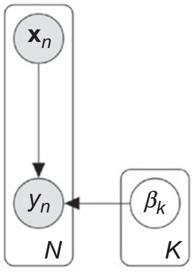
For any value of $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K$, the posterior probability in M_2 can be inferred from Eq. (7) only approximately, because the normalization constant in Bayes' rule (5) is intractable. Fig. 14 shows a Stan program for M_2.

3.1 Improvement: Hierarchical Modeling

Our initial model M_2 has one set of parameters $\boldsymbol{\beta}_k$ for each class (travel mode) $k = 1 \ldots K$. In other words, M_2 is strongly "pooled", as all observations for the

```
Given $\mathbf{X}, K, \lambda$:
    1. For each class $k = 1..K$:
        a. Draw coefficients $\boldsymbol{\beta}_k \sim \mathcal{N}(\boldsymbol{\beta}_k | \mathbf{0}, \lambda\mathbf{I})$.
    2. For each $n = 1..N$:
        a. Draw class $y_n \sim$
           $Cat(y_n | \text{Softmax}(\boldsymbol{\beta}_1^T \mathbf{x}_n, \ldots, \boldsymbol{\beta}_K^T \mathbf{x}_n))$.
```

(A)



(B)

FIG. 13 Probabilistic Graphical Model (A) and generative process (B) for classification model M_2.

```
data {
    int N, D, K; // Number of observations, features, and classes, respectively.
    matrix[N, D] X; // Input features.
    int y[N]; // Output classes.
}
parameters {
    matrix[K, D] beta; // Feature coefficients for each class.
}
model {
    for (k in 1:K) {
        for (d in 1:D) {
            beta[k, d] ~ normal(0, 10); // Prior.
        }
    }
    for (i in 1:N) {
        y[i] ~ categorical(softmax(beta * X[i, :]')); // Likelihood.
    }
}
```

FIG. 14 Stan program for classification model M_2.

same class share the same parameters. We will now revise this modeling assumption, and improve the model by constructing a **hierarchy of parameters** in two successive steps.

First, we note that for any class k, $\boldsymbol{\beta}_k$ is shared among all G individuals, as also seen in the PGM in Fig. 13B. However, different individuals could surely have very different travel mode preferences. Therefore, as a first step, we change the model to have one parameter vector $\boldsymbol{\beta}_k^{(g)}$ for each class k *and each individual* $g = 1\ldots G$. This has the same effect as grouping the trips of each individual, and assigning each group its own separate parameters. The resulting, intermediate model M_2^{Hier1} appears in Fig. 15A.

However, this change alone will likely lead to terrible overfitting in our case, because the data set has many more features than observations per group (i.e., per individual). We counter this by also modeling a restriction, whereby for

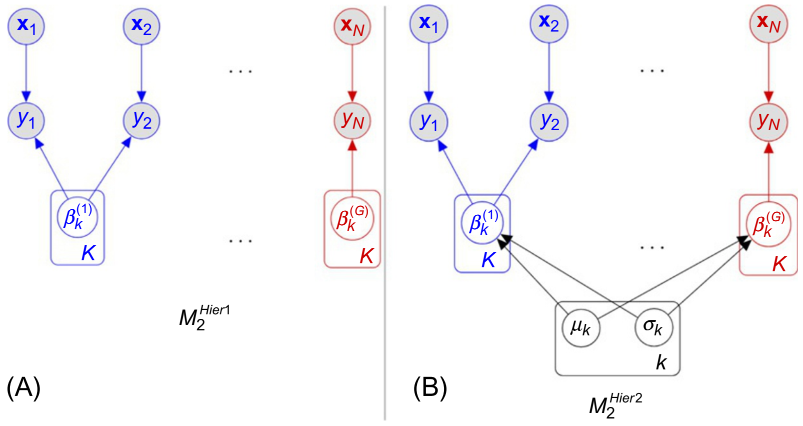


FIG. 15 Probabilistic Graphical Models for case study 2: (A) intermediate, one-level hierarchical model M_2^{Hier1}, (B) final, two-level hierarchical model M_2^{Hier2}. Different colors indicate different groups.

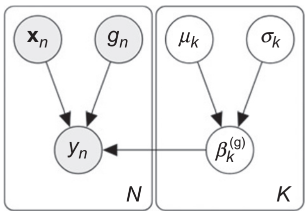
each class k, all parameters $\boldsymbol{\beta}_k^{(1)}, \ldots, \boldsymbol{\beta}_k^{(G)}$ are drawn from the same distribution. In this manner, our prior belief is that travel mode preferences can differ between individuals, but are nevertheless drawn from a common distribution. Our new and improved model M_2^{Hier2} thus comprises of a two-level hierarchy of parameters, as seen in the PGM in Fig. 15B.

For each class $k=1\ldots K$, we let the common distribution of parameters $\boldsymbol{\beta}_k^{(1)}$, …, $\boldsymbol{\beta}_k^{(G)}$ in M_2^{Hier2} be a multivariate Gaussian with mean $\boldsymbol{\mu}_k$ and positive variance $\exp(\sigma_k)$, so that $\boldsymbol{\mu}_k$ and σ_k are additional latent parameters. Fig. 16 gives the generative process for M_2^{Hier2}, while Fig. 17 provides a corresponding Stan program.

From here on, we stop specifying how the joint probability distribution factorizes for each model. As before, this factorization can be derived directly from the PGM and generative process for the model.

```
Given $\mathbf{X}, K, g_1, \ldots, g_N, \lambda, \tau$:
1. For each class $k = 1..K$:
   a. Draw global mean $\boldsymbol{\mu}_k \sim \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{0}, \lambda\mathbf{I})$.
   b. Draw global std. dev. $\sigma_k \sim \mathcal{N}(\sigma_k|0, \tau)$.
   c. For each group $g = 1..G$:
      i. Draw coefficients $\boldsymbol{\beta}_k^{(g)} \sim \mathcal{N}(\boldsymbol{\beta}_k^{(g)}|\boldsymbol{\mu}_k, \exp(\sigma_k)\mathbf{I})$.
2. For each $n = 1..N$:
   a. Draw class $y_n \sim Cat\left(y_n|\mathrm{Softmax}\left(\left(\boldsymbol{\beta}_1^{(g_n)}\right)^T \mathbf{x}_n, \ldots, \left(\boldsymbol{\beta}_K^{(g_n)}\right)^T \mathbf{x}_n\right)\right)$
```

(A)



(B)

FIG. 16 Generative process (A) for hierarchical model M_2^{Hier2}, and corresponding PGM (B), which is equivalent to Fig. 15B.

```
data {
    int N, D, K;   matrix[N, D] X;   int y[N]; // As in model M_2.
    real<lower=0> lambda, tau; // Non-negative hyper-parameters.
    int G; // Number of groups.
    int g[N]; // Group label for each x_n, y_n.
}
parameters {
    matrix[K, G] beta; // Feature coefficients for each group.
    vector[K] mu; // For shared mean prior per class.
    vector[K] sigma; // For shared std. dev. prior per class.
}
model {
    for (k in 1:K) { // Priors.
        mu[k] ~ normal(0, lambda);
        sigma[k] ~ normal(0, tau);
        beta[k] ~ normal(mu[k], exp(sigma[k])); // 2nd dimension G is vectorized.
    }
    for (n in 1:N) { // Likelihood.
        y[n] ~ categorical(softmax(beta[:, g[n]] * X[n, :]'));
    }
}
```

FIG. 17 Stan program for hierarchical model M_2^{Hier2}. Note the difference in the second dimension of `beta` vs the Stan program for the completely pooled model M_2 (Fig. 14).

Hierarchical modeling is thus a compromise between two extremes. One extreme is too much pooling, such as assigning the same parameters to all groups (Fig. 13B), while the other extreme is too little pooling, such as assigning separate parameters for each group (Fig. 15A). When modeling in general, one chooses the degree of pooling based on the data and the assumed priors.

4 CASE STUDY 3: FREEWAY OCCUPANCY IN SAN FRANCISCO

Our third case study deals with a **temporal data set**: lane occupancy rates in a freeway in San Francisco, as measured by a sensor on the road. The measurements are aggregated into consecutive 10-min intervals, so that the resulting time series $\mathbf{y}=y_1, \ldots, y_T$ consists of $T=1008$ consecutive real values ("lags"), each between 0 and 1.

Our goal is to model the freeway occupancy rates. Some applications of such a model include: prediction of future occupancy rates for better traffic routing, real-time detection of extraordinary queuing, and better understanding of driver behavior. To this end, we next build and incrementally improve a model, as well as introduce the highly versatile families of SSM and Linear Dynamical Systems (LDS).

4.1 Autoregressive Model

Our first model M_3 is $AR(k)$, namely an **autoregressive** model of some prefixed order k. In M_3, each lag y_t is a linear combination of the k preceding lags, plus white noise. Thus in the PGM for M_3, the nodes are $y_1, \ldots, y_T$, and the edges are from each y_n to each of $y_{n+1}, \ldots, y_{n+k}$. The generative process for M_3 appears in Fig. 18; note the special treatment of the first k observations, for which some previous lags are unknown.

4.2 State-Space Model

Noise in M_3 may build up over time, because each observation depends on k previous observations, themselves noisy. We thus move to a more robust,

```
Given $\mathbf{y}, k, \lambda, \sigma, \mu_0$ :
  1. Draw transition coefficients $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_k) \sim \mathcal{N}(\boldsymbol{\beta}|\mathbf{0}, \lambda\mathbf{I})$.
  2. Draw first observation $y_1 \sim \mathcal{N}(y_1|\mu_0, \sigma^2)$.
  3. For $i = 2..k$:
      a. Draw $i$'th observation $y_i \sim \mathcal{N}(y_i|\beta_1 y_{i-1} + \beta_2 y_{i-2} + \cdots + \beta_{i-1} y_1, \sigma^2)$.
  4. For $t = (k+1)..T$:
      a. Draw observation $y_t \sim \mathcal{N}(y_t|\beta_1 y_{t-1} + \beta_2 y_{t-2} + \cdots + \beta_k y_{t-k}, \sigma^2)$.
```

FIG. 18 Generative process for $AR(k)$ model. Steps 2 and 3 handle the first k observations, for which some previous lags are unknown. Step 4 handles the subsequent observations, for which all k previous lags are known.

SSM M_3^{SSM}, in which the observations are generated ("emitted") by a stochastic state. That is, M_3^{SSM} assumes that occupancy rates evolve over time through some latent, nondeterministic process, which is visible only through its observed measurements $y_1, \ldots, y_T$.

As such, M_3^{SSM} has a set of **hidden state** variables $\mathbf{h}_1, \ldots, \mathbf{h}_T$: vectors of dimension D, which we are free to choose. Each state yields the next state per some **transition distribution**. Each observation y_t is then a draw from a **measurement distribution**, which depends on the corresponding hidden state $\mathbf{h}_t$. Fig. 19A shows the basic structure of a PGM for M_3^{SSM}. Special treatment is required only for the first hidden state $\mathbf{h}_1$, for which no preceding hidden state is defined.

4.3 Linear Dynamical Systems

It remains to describe the form of the transition and measurement distributions. For computational tractability, we assume these distributions are linear Gaussians, so that

$$\mathbf{h}_t \sim \mathcal{N}(\mathbf{h}_t | \mathbf{B}\mathbf{h}_{t-1}, \mathbf{R}) \tag{8}$$

$$y_t \sim \mathcal{N}\left(y_t | \mathbf{c}^T \mathbf{h}_t, \sigma^2\right) \tag{9}$$

where $\mathbf{B}^{D\times D}$, $\mathbf{R}^{D\times D}$, $\mathbf{c}^{D\times 1}$, σ are latent model parameters to be inferred. The corresponding PGM appears in Fig. 19B. This model, which we denote as M_3^{LDS}, is called an **LDS**. It is also widely known as **Linear Kalman Filter** in control theory literature (Grewal, 2011).

For any choice of priors for the latent parameters $\mathbf{B}$, $\mathbf{R}$, $\mathbf{c}$, σ, we can obtain the generative process for M_3^{LDS}, by combining the PGM in Fig. 19B with distributions (8) and (9). By imposing **structural restrictions** on parameter matrices $\mathbf{B}$, $\mathbf{R}$ in M_3^{LDS}, many popular time series can be modeled (including $AR(k)$), and inference becomes less computationally demanding (Nasrabadi, 2007).

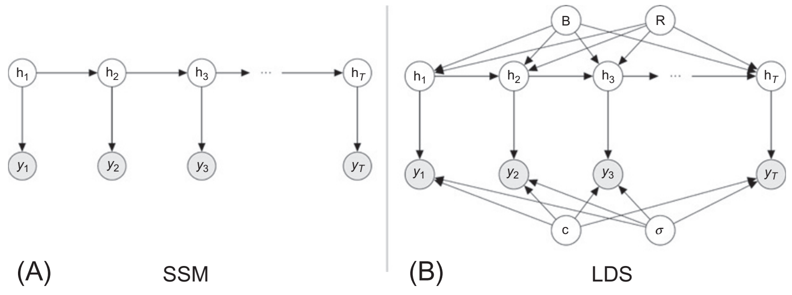


FIG. 19 PGMs for (A) general State-Space Model (SSM), and (B) Linear Dynamical System (LDS), a subclass of SSM.

4.4 Common Enhancements to LDS

Let us now show that LDS models are easily adaptable to various common usage scenarios. Some of the PGMs for the enhanced models may appear very similar to the PGM of M_3^{LDS} in Fig. 19B, but the underlying probabilistic models are significantly different nonetheless.

4.4.1 Filling Gaps

Suppose that the time series has some missing lags in-between 1 and T. We can perform **imputation** for any missing value y_t, simply by modeling y_t as a latent (i.e., unobserved) measurement from state $\mathbf{h}_t$. Similarly, to generate **predictive distributions** for $r \geq 1$ lags immediately after T, we simply extend M_3^{LDS} to include latent states $\mathbf{h}_{T+1}, \ldots, \mathbf{h}_{T+r}$, along with corresponding latent measurements $y_{T+1}, \ldots, y_{T+r}$. Fig. 20 illustrates both changes to the PGM of M_3^{LDS}.

Following is an example of data imputation for this case study of lane occupancy rates. First, we randomly select 144 observations in $\mathbf{y}$, then partition the selection randomly into a train set $\mathbf{y}^{known}$ of size 29,and a test set $\mathbf{y}^{missing}$ of size 115. As nearly 80% of observations are rendered missing, this is a relatively difficult imputation setting. We then fit M_3^{LDS} on $\mathbf{y}^{known}$, and run the trained model on $\mathbf{y}^{missing}$. Fig. 21 illustrates the prediction quality, and shows that M_3^{LDS} fills in the missing data rather accurately.

4.4.2 External Data

To take advantage of any additional data $\mathbf{x}_1, \ldots, \mathbf{x}_T$, e.g., weather or seasonality information, we can augment M_3^{LDS} simply as

$$\mathbf{h}_t \sim \mathcal{N}(\mathbf{h}_t | \mathbf{B}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t, \mathbf{R})$$

where $\mathbf{W}$ is an additional matrix of latent coefficients to be inferred. Fig. 22A shows one time step of the PGM for the augmented model.

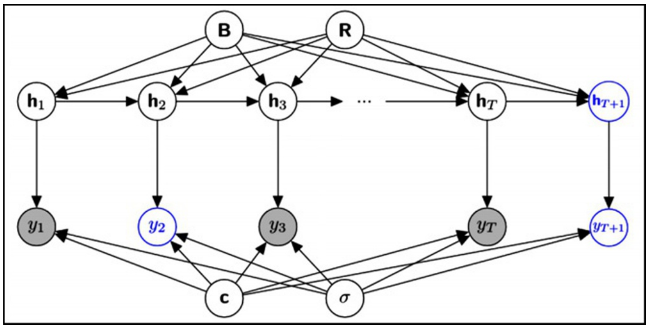


FIG. 20 PGM for an LDS model with imputation at time step 2, and forecasting of time step $T+1$.

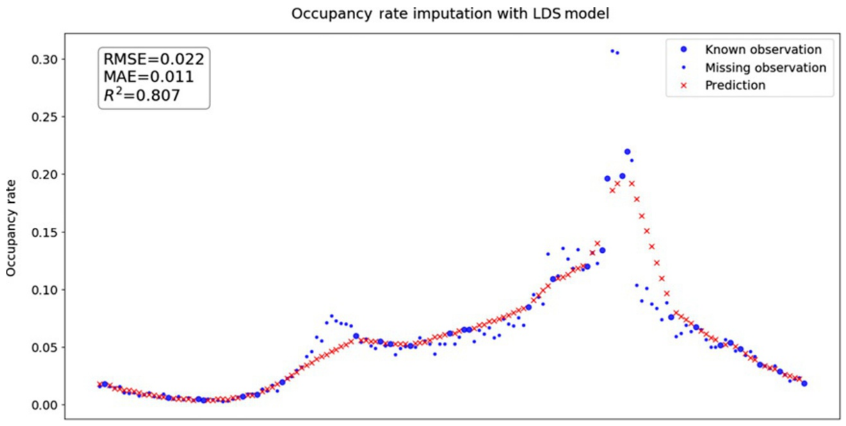


FIG. 21 Occupancy rate imputation with M_3^{LDS} for 144 randomly selected observations, 80% of which are treated as missing.

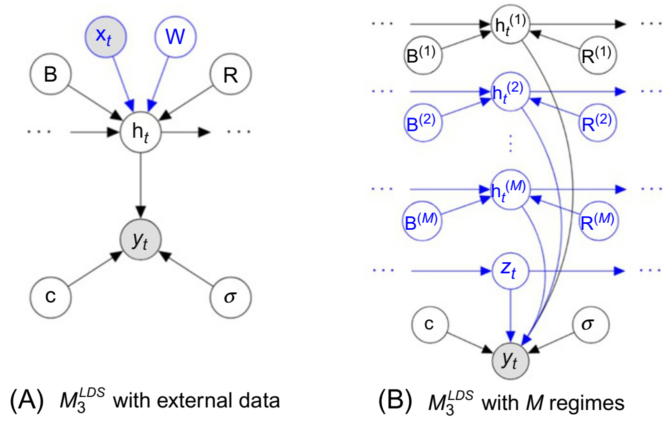


FIG. 22 One time step t in the PGM for an enhanced LDS model, with changes to M_3^{LDS} highlighted in color: (A) external data, (B) M regimes with separate transition distributions.

4.4.3 Regimes

Suppose now that occupancy rates behave quite differently under different **regimes**, such as congested traffic, free flow, road incidents, or harsh weather. We can model such regimes with a **Switching LDS**, the PGM of which appears in Fig. 22B, as following.

First, we prefix the number of regimes M, and assign to each regime $m=1\ldots M$ a dedicated chain of latent states $\mathbf{h}_1^{(m)}, \ldots, \mathbf{h}_T^{(m)}$. Then, at time step $t=1\ldots T$, we let categorical variable z_t choose which of regimes $1\ldots M$ applies. Finally, we model y_t as a measurement from $\mathbf{h}_t^{(z_t)}$: the hidden state corresponding to the regime which z_t chooses. We can also assign (perhaps hierarchically) matrices $\mathbf{B}^{(m)}$, $\mathbf{R}^{(m)}$ for the transition distribution of chain m, because different regimes can evolve differently over time.

4.5 NonLinear Variations on LDS

We conclude the discussion of LDS by introducing two commonly used variations, wherein the transition and measurement distributions incorporate **nonlinear functions**. One popular variation is the **Extended Kalman Filter (EKF)**, which replaces the linear combinations in Eqs. (8), (9) with freely chosen, **differentiable functions** f and g, namely

$$\mathbf{h}_t \sim \mathcal{N}(\mathbf{h}_t | f(\mathbf{h}_{t-1}), \mathbf{R})$$

$$y_t \sim \mathcal{N}\left(y_t | \, g(\mathbf{h}_t), \sigma^2\right)$$

EKF is at the foundation of a very wide range of applications, including many navigation systems (Wan, 2006).

Another popular variation on LDS is the **Hidden Markov Model (HMM)** (Schuster-Böckler and Bateman, 2007), which uses discrete, categorical distributions for the transitions and measurements. Without loss of generality, let us assume the same set of possible values $1\ldots K$ for both hidden states and measurements.

In HMM, the transition distribution is specified as a matrix $\mathbf{B} \in \mathbb{R}^{K\times K}$. For all $i, j = 1\ldots K$, $\mathbf{b}_{i,j}$ is the probability to obtain state value j, given that the preceding state had value i. Matrix $\mathbf{C} \in \mathbb{R}^{K\times K}$ similarly specifies the measurement distribution. Hence in HMM, matrices **B** and **C** are latent parameters, which we intend to infer. Fig. 23 presents the corresponding PGM.

For any hidden state value, the corresponding rows in **B** and **C** cover *all possible* transitions and measurements from that state, respectively. Therefore, each row in **B** and in **C** is a vector of k probabilities, which sum to 1. We can draw such normalized probability vectors from *Dir*, the continuous **Dirichlet distribution** (Bishop, 2008; Section 2.2.1), for which we use hyperparameters $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^k$, as in the generative process in Fig. 24.

Finally, we remark that it is also common to use hybrid HMM-LDS models, which combine, e.g., discrete hidden states with continuously distributed measurements (Rabiner, 1990).

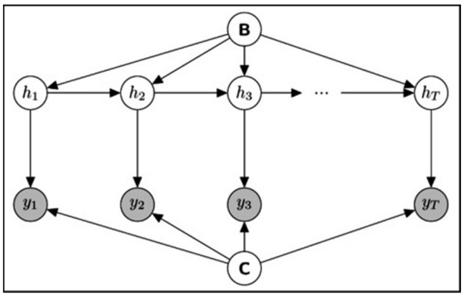


FIG. 23 PGM for Hidden Markov Model (HMM).

```
Given $\mathbf{y}, K, \boldsymbol{\alpha} \in \mathbb{R}^K, \boldsymbol{\beta} \in \mathbb{R}^K$:
    1. For each $k = 1..K$:
        a. Draw transition probabilities $\mathbf{b}_k \sim Dir(\mathbf{b}_k|\boldsymbol{\alpha})$.
        b. Draw measurement probabilities $\mathbf{c}_k \sim Dir(\mathbf{c}_k|\boldsymbol{\beta})$.
    2. For each time step $t = 1..T$:
        a. Draw hidden state $\mathbf{h}_t \sim Cat(h_t|\mathbf{b}_{\mathbf{h}_{t-1}})$.
        b. Draw observed measurement $y_t \sim Cat(y_t|\mathbf{c}_{\mathbf{h}_t})$.
```

FIG. 24 Generative process for Hidden Markov Model. $\mathbf{b}_k$ and $\mathbf{c}_k$ denote the $\boldsymbol{k}$'th row in **B** and **C**, respectively.

5 CASE STUDY 4: INCIDENT DURATION PREDICTION

Textual data is often an additional source of useful information for statistical models, e.g., websites and social media can provide prediction models with highly pertinent context. However, textual data is nonnumeric, often unstructured, and poses challenges of processing natural language.

This case study is based on Pereira et al. (2013) and gives a methodological example of combining textual and numeric data in a prediction model. The data set consists of police reports of $D = 10{,}000$ incidents in years 2010–2011 in several expressways. For each report, we have **numeric features**—such as incident time, number of affected lanes, and duration until clearance—and a **textual document**, which describes the incident chronology in free-form text.

Our goal is to build a model for predicting incident duration. Such a model is also useful for studying the impact of incidents on traffic. We begin by performing regression on only the numeric features, as described in (Pereira et al., 2013). Fig. 25A shows that the predictions of this initial model M_4^{NoText} are far from ideal.

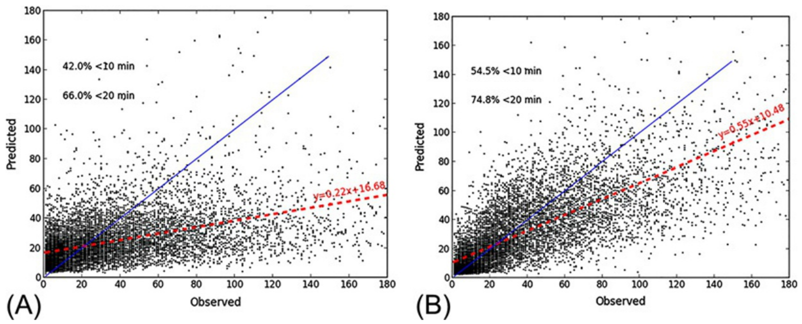


FIG. 25 Prediction of incident duration in minutes, either (A) without or (B) with text-based features. The blue line indicates ideal performance, while the red dashed lines correspond to linear fits on the observations. The plot for each model also shows the percentage of prediction errors which are smaller than 10 or 20 min (Pereira et al., 2013).

For the rest of this case study, we describe how to **encode** each textual document compactly as a numeric vector. We can then concatenate the text-based encodings with the corresponding numeric features, to obtain unified data vectors for all the documents.

Finally, we fit a regression model $M_4^{WithText}$ on the unified data vectors, and obtain significantly better predictive performance than M_4^{NoText}. As Fig. 25B shows for $M_4^{WithText}$, the predictions are much closer to ideal, and there are significantly less observations for which the prediction error is greater than 10 (or 20) min.

5.1 Preprocessing

As a first common step in dealing with textual information, we preprocess each document into a simplified textual form, which is easier to encode. For instance, we reduce each word to its stem (e.g., each of "injury," "injuries," and "injured" becomes "injur"), and remove common function words (e.g., "the," "is," and "which"). We then collect all unique words in the processed documents into an arbitrarily ordered list $C=(C_1,C_2,\ldots)$, which we call "the **dictionary**". In this case study, the length $|C|$ of the dictionary is in the thousands.

5.2 Bag-of-Words Encoding

Any preprocessed document can now be encoded numerically as a **Bag of Words (BoW)**: a vector where the j'th element is the frequency of dictionary word C_j in the document. It may thus be tempting to encode each of the processed documents as BoW, and feed the BoW vectors along with the other numeric features into a prediction model. However, doing so could greatly hinder learning, because the dimension $|C|$ of a BoW vector is typically large, as in this case study. We next overcome this problem through Topic Modeling, which yields for each document a numeric encoding much shorter than BoW, as illustrated in Fig. 26.

5.3 Latent Dirichlet Allocation

Going over the incident reports, we notice that they seem to revolve around several repeated topics, such as: incident severity, casualties, oil spill, road blockage, and emergency services. Each textual report thus pertains to a mixture of topics of different proportions, although it is uncertain what the topics are and how they mix for each document. **Topic Modeling** generalizes this idea of topics over documents. For this case study, we introduce a relatively simple topic model called **Latent Dirichlet Allocation (LDA)**.

In LDA, a **topic** as a concept, which is formulated from the words in the dictionary C. Namely, a topic is an unobserved vector of weights, where the j'th entry corresponds to word C_j. For example, a topic $\mathbf{s}_1$ which pertains to

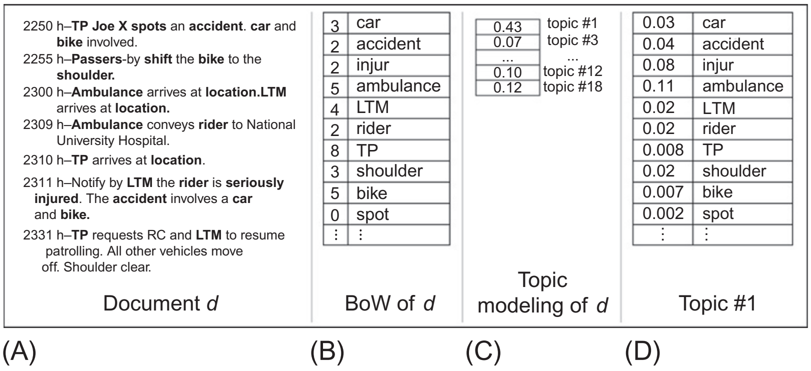

2250 h–TP Joe X spots an accident. car and bike involved.
2255 h–Passers-by shift the bike to the shoulder.
2300 h–Ambulance arrives at location.LTM arrives at location.
2309 h–Ambulance conveys rider to National University Hospital.
2310 h–TP arrives at location.
2311 h–Notify by LTM the rider is seriously injured. The accident involves a car and bike.
2331 h–TP requests RC and LTM to resume patrolling. All other vehicles move off. Shoulder clear.
Document d
3 car
2 accident
2 injur
5 ambulance
4 LTM
2 rider
8 TP
3 shoulder
5 bike
0 spot
BoW of d
0.43 topic #1
0.07 topic #3
0.10 topic #12
0.12 topic #18
Topic modeling of d
0.03 car
0.04 accident
0.08 injur
0.11 ambulance
0.02 LTM
0.02 rider
0.008 TP
0.02 shoulder
0.007 bike
0.002 spot
Topic #1
(A)
(B)
(C)
(D)

FIG. 26 (A) Example incident report, before preprocessing. (B) Bag-of-Words encoding of the report. (C) Much shorter encoding of the report through topic modeling. (D) Example topic, which pertains mainly to serious injuries (Pereira et al., 2013).

the concept of a severe accident could attribute high weights to each of the words "injur," "ambulance," and "emergency," whereas a topic $\mathbf{s}_2$ pertaining to a large oil spill could attribute a high weight to "oil."

Also in LDA, we assume that each **document** pertains to an unobserved mixture of the topics themselves. For instance, a police report about a fatal oil spill may pertain to a topic mixture, in which each of topics $\mathbf{s}_1$ and $\mathbf{s}_2$ is weighted as 40%. Finally in LDA, we assume that the mixture of topics per document yields a probability distribution, from which every **word** in the document is drawn.

Before rigorously formulating all the above-mentioned components of LDA, we note that although LDA might at first seem to be an overly simplistic probabilistic model, it is nevertheless powerful and effective. Indeed, for this case study, LDA yields numeric encodings which greatly improve regression results, as Fig. 25 shows. Furthermore, LDA admits completely **unsupervised learning**, as none of the documents is actually labeled with any topics.

5.3.1 Formal Definition

For LDA, we need only choose and fix the number of topics K. To reduce dimensionality, K should be chosen as much smaller than the length $|C|$ of the dictionary, so for this case study of incidents, we choose $K=25$. We denote the latent topics as normalized vectors $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_K$, each of length $|C|$.

Next, for each document $d=1\ldots D$, we define the topic mixture as a linear combination of $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_K$, and let latent vector $\boldsymbol{\theta}_d$ denote the corresponding linear coefficients.

Finally, we let N_d denote the number of words in document d, and let $w_{d,n}$ denote the n'th word in d. Interchangeably, $w_{d,n}$ is a number in $1\ldots|C|$, indicating the index of the corresponding word in the dictionary. Word $w_{n,d}$ is then generated in two steps. First, a topic $z_{d,n}$ is chosen per the topic mixture $\boldsymbol{\theta}_d$ of the document d. Then, $w_{n,d}$ is drawn per the word proportions in the chosen topic $z_{d,\,n}$.

The resulting probabilistic model is M_{LDA}, in which the latent parameters to be inferred are $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_K$ for the topics, and $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_D$ for the documents. The corresponding PGM appears in Fig. 27, and the generative process is in Fig. 28. A Stan program for M_{LDA} is available in Section 17.5 of (Stan, 2018).

In LDA, the priors are Dirichlet distributions, hence the model name. For any document d, $\boldsymbol{\alpha} \in \mathbb{R}^K$ parameterizes the prior on topic mixture $\boldsymbol{\theta}_d$, while for any topic k, $\boldsymbol{\beta} \in \mathbb{R}^{|C|}$ parameterizes the prior on word mixture $\boldsymbol{\phi}_k$. It follows from the dimension of $\boldsymbol{\alpha}$ that for any document d, $\boldsymbol{\theta}_d$ is of relatively low dimension $K \ll |C|$.

This concludes the definition of M_{LDA}. The joint distribution of M_{LDA} has no analytical solution, hence LDA is carried out through approximate Bayesian inference. For example, we could use Variational Inference (VI) (Wainwright and Jordan, 2008), which yields the approximate parameters for the posterior distributions. The posterior distributions of $\boldsymbol{\theta}_d$ and $\boldsymbol{\phi}_k$ are also Dirichlet, parameterized on some other vectors from $\mathbb{R}^K$ and $\mathbb{R}^{|C|}$, respectively.

In particular, for any document $d = 1 \ldots D$, we can obtain from VI an approximation of parameter $\boldsymbol{\gamma}_d$ for the posterior Dirichlet distribution of $\boldsymbol{\theta}_d$. $\boldsymbol{\gamma}_d$ is of low dimension $K \ll |C|$—much shorter than BoW encoding—and expresses a

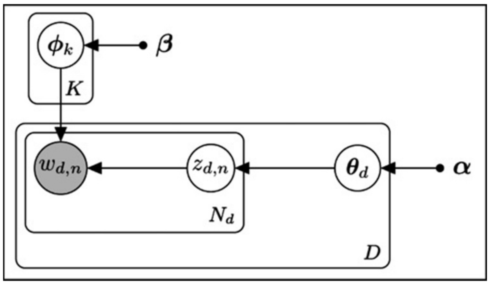


FIG. 27 PGM for M_{LDA}.

```
Given $\boldsymbol{\alpha} \in \mathbb{R}^K$, $\boldsymbol{\beta} \in \mathbb{R}^{|C|}$, and documents:
    1. For each topic $k = 1..K$:
        a. Draw word proportions $\boldsymbol{\phi}_k \sim Dir(\boldsymbol{\phi}_k|\boldsymbol{\beta})$.
    2. For each document $d = 1..D$:
        a. Draw topic proportions $\boldsymbol{\theta}_d \sim Dir(\boldsymbol{\theta}_d|\boldsymbol{\alpha})$.
        b. For each word $n = 1..N_d$:
            i. Draw topic $z_{d,n} \sim Cat(z_{d,n}|\boldsymbol{\theta}_d)$.
            ii. Draw word $w_{d,n} \sim Cat(w_{d,n}|\boldsymbol{\phi}_{z_{d,n}})$.
```

FIG. 28 Generative process for M_{LDA}.

posterior belief about topic proportions in d. Hence for regression purposes, we can use γ_d as the desired, short numeric encoding of d.

6 SUMMARY

Through several case studies, we have described the MBML, which provides a systematic framework for approaching any modeling problem. The first step in MBML is to formulate our assumptions about the *uncertainty* structure of the problem as a stochastic model, which we express through a PGM and a generative process. MBML thus contrasts with traditional Machine Learning approaches, which require us to first select the learning algorithm among a multitude of options.

The following step in MBML is to infer the posterior distribution of unknown parameters in the model, given observed evidence. Subsequently, it is often straightforward to apply model improvements, as well as incorporate new evidence, if any.

In MBML, the posterior distribution is learnt through Bayesian Inference, either exactly or approximately. A popular tool for Bayesian Inference is Stan, and we have provided examples of Stan programs for the various models we built in the case studies. These models apply to a wide range of common modeling problems, including regression, classification, time series, and textual data.

We have limited this chapter to the core concepts of Model-Based Machine Learning, and concentrated on demonstrating the ease and applicability of this approach. However, MBML reaches far beyond this constrained review, and offers a radically different alternative to traditional Machine Learning techniques. It thus comes as no surprise that MBML is currently a very active subject of research and software development.

6.1 Further Reading

This chapter is based on notes of course "Model-Based Machine Learning" by co-authors F. Pereira and F. Rodrigues (42186 Model-based machine learning, 2018), which encompasses many additional aspects of MBML. Some parts of this chapter are also adapted from the PhD thesis of F. Rodrigues (Rodrigues, 2016).

We encourage the enthusiastic reader to extend their knowledge into additionally important MBML concepts, as following. Conjugate priors and the exponential family of distributions play important roles in Bayesian Inference (Duda and Hart, 1973; Bernardo and Smith, 2009). There exist many methods for approximate Bayesian Inference, such as Markov Chain Monte Carlo (Gilks, 2005), or VI (Jordan et al., 1999; Wainwright and Jordan, 2008). There are also methods for exact Bayesian Inference, such as Belief Propagation (Pearl, 2014). PGMs other than directed Bayesian networks are available, such as unordered graphs and factor graphs (Kschischang et al., 2001). Finally, there is ongoing research into exciting combinations of probabilistic models with Deep Learning (Wang and Yeung, 2016).

REFERENCES

42186 Model-based machine learning. [Online]. 2018. Available: http://kurser.dtu.dk/course/42186. Accessed 8 June 2018.

Bernardo, J., Smith, A., 2009. Bayesian Theory. vol. 405. John Wiley & Sons, New York City, USA.

Bishop, C.M., 2008. Pattern Recognition and Machine Learning. Springer-Verlag, New York.

Duda, R., Hart, P., 1973. Pattern Classification and Scene Analysis. vol. 3. Wiley, New York City, USA.

Gelman, A., Jakulin, A., Pittau, M.G., Su, Y.-S., et al., 2008. A weakly informative default prior distribution for logistic and other regression models. Ann. Appl. Stat. 2 (4), 1360–1383.

Gilks, W., 2005. Markov Chain Monte Carlo. Wiley Online Library.

Gordon, A.D., Henzinger, T.A., Nori, A.V., Rajamani, S.K., 2014. Probabilistic programming. In: Proceedings of the on Future of Software Engineering. Hyderabad, India, pp. 167–181.

Grewal, M.S., 2011. Kalman filtering. In: International Encyclopedia of Statistical Science. Springer, pp. 705–708.

Jensen, F.V., 1996. An Introduction to Bayesian Networks. vol. 210. UCL Press, London.

Jordan, M.I., Ghahramani, Z., Jaakkola, T., Saul, L., 1999. An introduction to variational methods for graphical models. Mach. Learn. 37 (2), 183–233.

Koller, D., Friedman, N., 2009. Probabilistic Graphical Models: Principles and Techniques. MIT Press, Cambridge, MA, USA.

Kschischang, F., Frey, B., Loeliger, H., 2001. Factor graphs and the sum-product algorithm. IEEE Trans. Inf. Theory 47 (2), 498–519.

Li, J., 2017. Assessing the accuracy of predictive models for numerical data: Not r nor r2, why not? Then what? PLoS ONE 12 (8), e0183250.

Multiplying matrices and vectors—Math Insight. [Online]. 2018. Available: https://mathinsight.org/matrix_vector_multiplication. Accessed 8 June 2018.

Murphy, K.P., 2012. Machine Learning: A Probabilistic Perspective. MIT Press, Cambridge, MA, USA.

Nasrabadi, N.M., 2007. Pattern recognition and machine learning. J. Electron. Imaging 16 (4), 49901.

National Oceanic and Atmospheric Administration | U.S. Department of Commerce. [Online]. 2018. Available: http://www.noaa.gov/. Accessed 1 April 2018.

Ng, A., 2004. Feature selection, L1 vs. L2 regularization, and rotational invariance.In: Proceedings of the International Conference on Machine Learning. p. 78.

NYC Open Data. [Online]. 2018. Available: https://opendata.cityofnewyork.us/. Accessed 1 April 2018.

R.S. Olson, W. La Cava, Z. Mustahsan, A. Varik, and J.H. Moore, "Data-driven advice for applying machine learning to bioinformatics problems," arXiv Prepr. arXiv1708.05070, 2017.

Pearl, J., 2014. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco, California.

Pereira, F., Rodrigues, F., Ben-Akiva, M., 2013. Text analysis in incident duration prediction. Transp. Res. Part C Emerg. Technol. 37, 177–192.

Rabiner, L.R., 1990. A tutorial on hidden Markov models and selected applications in speech recognition. In: Readings in Speech Recognition. Morgan Kaufmann Publishers Inc., San Mateo, California, pp. 267–296

Rodrigues, F., 2016. Probabilistic Models for Learning from Crowdsourced Data. University of Coimbra, Coimbra, Portugal.

Schuster-Böckler, B., Bateman, A., 2007. An introduction to hidden Markov models. Curr. Protoc. Bioinform. 18 (1), A–3A.

Stan. [Online]. 2018. Available: http://mc-stan.org/. Accessed 8 June 2018.

Wainwright, M., Jordan, M., 2008. Graphical models, exponential families, and variational inference. Found. Trends Mach. Learn. 1 (1–2), 1–305.

Wan, E., 2006. Sigma-point filters: An overview with applications to integrated navigation and vision assisted control. In: Nonlinear Statistical Signal Processing Workshop, 2006 IEEE, pp. 201–202.

Wang, H., Yeung, D.Y., 2016. Towards Bayesian deep learning: a framework and some existing methods. IEEE Trans. Knowl. Data Eng. 28 (12), 3395–3408.

Woolf, P., Burge, C., Keating, A., Yaffe, M., 2004. Statistics and Probability Primer for Computational Biologists. Massachusetts Institute of Technology, Cambridge, MA, USA.