Regression models

Filipe Rodrigues

Francisco Pereira

DTU Management Engineering
Department of Management Engineering

Outline

- Case study: Modeling taxi demand in NYC
- Linear regression
- Poisson regression
- Heteroscedastic regression
- Non-linear models

Learning objectives

At the end of this lecture, you should be able to:

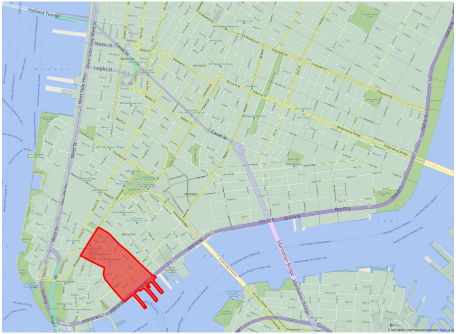
- Explain what (Bayesian) linear regression is and its underlying assumptions
- Modify the standard linear regression assumptions (e.g. Poisson likelihood or other) according to the data available
- Explain what an heterescedastic model is, and how to model data-dependent observation noise
- Explain the basic ideas behind neural networks and what they allow you to do
- Relate different ways of modelling the dependency of a continuous random variable on other variables and justify their suitability for a problem, e.g.:
    - Linear-Gaussian relationship
    - Poisson with conditional rate
    - Gaussian with data-dependent noise
    - Combinations of linear models with (Bayesian) neural networks
- Implement the modelling techniques above in STAN/Pyro

Modeling taxi demand in New York City

- (Almost) all taxi trips in NYC from 2009 to mid 2016
- Original files have one trip per line
  - Pick-up location and time
  - Drop-off location and time
  - Other variables such as trip price and number of passengers
- Weather data from the National Oceanic and Atmospheric Administration
- Research question: **model taxi pickups across the city**
- Useful to optimize taxi service
  - Similar to many other demand problems (shared modes, public transport, energy, water, goods, communication...)

Modeling taxi demand in New York City (cont'd)

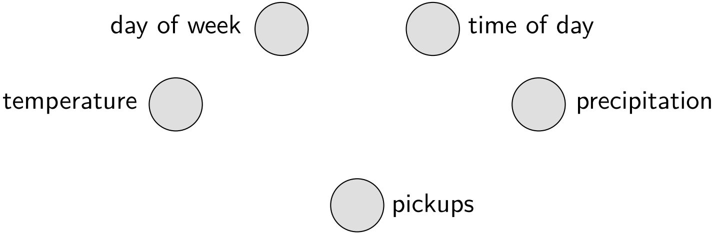
- Preprocessed data
  - Grouped data by census tract in 1 hour intervals
  - Extended grouped taxi data with relevant weather information
- Case study: **Wall Street**



- What we know: day of the week, time of the day, temperature, precipitation, etc.
- Target variable: number of taxi pickups

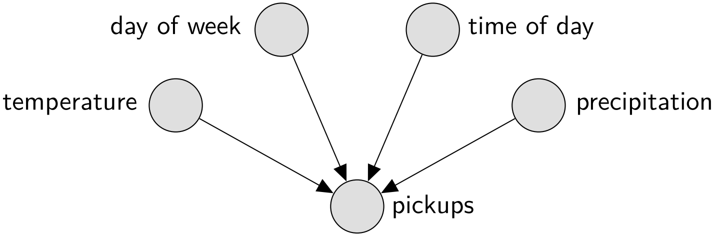
Modeling taxi demand in New York City (cont'd)

- Let's start thinking about the graphical model...



Modeling taxi demand in New York City (cont'd)

- Let's start thinking about the graphical model...



- What distribution should we assign to the pickups variable?
- How should we model the dependency of the pickups on the other variables?
- Do we need to assign distributions to these other variables (i.e. temperature, day of week, time of day, etc.)?
- This puts us right into the **regression** framework!

Regression

- Regression - predict response variable y from a collection of D predictor variables $x_1, x_2, \ldots, x_d, \ldots, x_D$

  y - target, response or dependent variable
  x_d - feature(s), covariate(s), explanatory or independent variable(s)

- The dependent variable y is a function of all the predictor variables: $y = f(x_1, x_2, \ldots, x_d, \ldots, x_D)$

- A few examples:
  - travel time prediction
  - predicting demand for autonomous vehicles
  - temperature/rainfall forecast
  - estimation of audience to a concert
  - prediction of future values of a share or a commodity (e.g. petrol)
  - prediction of house prices, number of voters in a state, births in a year
  - and, of course, predicting taxi demand!

Linear regression

- The dependent variable y is a function of all the predictor variables

$$y = f(x_1, x_2, \dots, x_d, \dots, x_D)$$

- Ok, but what function?
- Simplest approach is to assume a **linear relationship**

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_D x_D = \beta_0 + \sum_{d=1}^{D} \beta_d x_d$$

  β_0 is the *intercept* (or bias) and $\{\beta_1, \beta_2, \dots, \beta_D\}$ are the *coefficients* (or weights)

- We can write this more compactly using **vector notation**

$$y = \beta_0 + \boldsymbol{\beta}^{\mathsf{T}} \mathbf{x}$$

  where $\boldsymbol{\beta} = (\beta_1, \beta_2, \dots, \beta_D)^{\mathsf{T}}$ and $\mathbf{x} = (x_1, x_2, \dots, x_D)^{\mathsf{T}}$

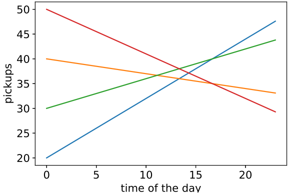
Note

The intercept β_0 can be seen as a coefficient for a special covariate x_0 that is always equal to 1. Thus, it is sometimes omitted.

Linear regression

- Linear assumption can seem naive...

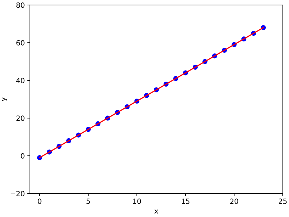
$$y = f(\mathbf{x}) = \boldsymbol{\beta}^\mathsf{T}\mathbf{x}$$



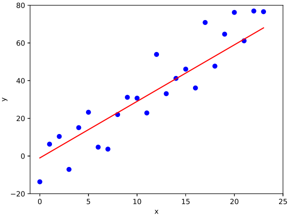
- But, the features **x** can be extremely **flexible**!
  - Any characteristic of the data
  - Indicator functions and 1-of-K encodings (e.g. $x_1 = \mathbb{I}[\text{weekend} = \text{True}]$)
  - Transformations of the original features (e.g. $x_2 = \log x_1$)
  - Basis expansion (e.g. $x_2 = x_1^2$ and $x_3 = x_1^3$) - polynomial fitting!
  - Interactions between features (e.g. $x_3 = x_1 x_2$)
- Key aspects of linear regression
  - Simplicity
  - Flexibility
- One of the most important and widely used methods in statistics and machine learning!

Linear regression

- In practice observations are **noisy**



(a) dream world



(b) tough reality

- Add error term ϵ to account for observation noise

$$y = \boldsymbol{\beta}^{\mathsf{T}}\mathbf{x} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

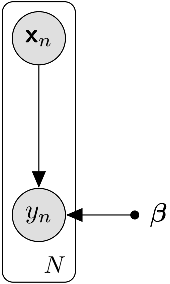
- We can equivalently write

$$\mathcal{N}(y|\boldsymbol{\beta}^{\mathsf{T}}\mathbf{x}, \sigma^2)$$

Linear regression as a graphical model

- We have a dataset $\mathcal{D}$ consisting of N observations of the targets y_n which depend on their corresponding explanatory variables $\mathbf{x}_n$

$$\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$$



- Generative process
  1. For $n = 1 \ldots N$ do:
     a. Draw target $y_n \sim \mathcal{N}(y_n | \beta^{\mathsf{T}} \mathbf{x}_n, \sigma^2)$

- Joint probability distribution factorizes as

$$p(\mathbf{y}|\mathbf{X}, \beta, \sigma) = \underbrace{\prod_{n=1}^{N} \mathcal{N}(y_n | \beta^{\mathsf{T}} \mathbf{x}_n, \sigma^2)}_{\text{likelihood}}$$

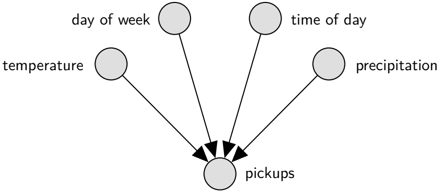
where $\mathbf{y} = \{y_n\}_{n=1}^{N}$, $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^{N}$, β are the model parameters and σ is fixed.

Note

We don't care about modeling $p(\mathbf{X})$. This is called a **conditional model** and contrasts with fully generative models.

Going back to our taxi demand case study...

- Let's revise the **modeling assumptions** that we made

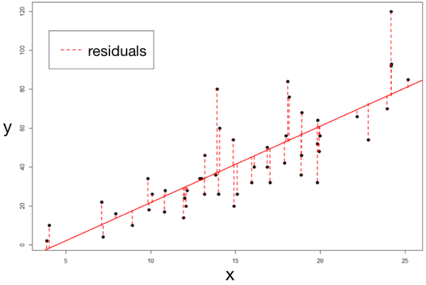


- What distribution should we assign to the pickups variable?
    - Gaussian
- How should we model the dependency of the pickups on the other variables?
    - Mean of the Gaussian distribution for the pickups is a linear function of the other variables
- Do we need to assign distributions to these other variables (i.e. temperature, day of week, time of day, etc.)?
    - In this case, no. They are always observed and we are only interested in modeling the behavior of the pickups variable

Model estimation (or fitting)

- **Goal:** given a dataset $\mathcal{D}$ find the coefficients β that best predict y given $\mathbf{x}$
- A reasonable approach is to minimize the sum of squared errors (*residuals*) between each fitted response $f(\mathbf{x}_n) = \boldsymbol{\beta}^\mathsf{T}\mathbf{x}$ and the true response y_n

$$\hat{\beta} = \arg\min_{\boldsymbol{\beta}} \sum_{n=1}^{N} \left(y_n - \boldsymbol{\beta}^\mathsf{T}\mathbf{x}_n\right)^2$$



- Has a nice analytical solution (the famous *normal equation*)

$$\hat{\beta} = \left(\mathbf{X}^\mathsf{T}\mathbf{X}\right)^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$$

Model estimation: an alternative view

- Alternatively, we can find the coefficients β that maximize the joint probability
  - In practice, for both numerical and computational reasons, we consider the logarithm of the joint probability instead
  - Recall that in this case, the joint probability distribution is just a product of N likelihood terms

$$\hat{\beta} = \arg\max_{\boldsymbol{\beta}} \log \prod_{n=1}^{N} \mathcal{N}(y_n|\boldsymbol{\beta}^\mathsf{T}\mathbf{x}_n, \sigma^2) = \arg\max_{\boldsymbol{\beta}} \sum_{n=1}^{N} \log \mathcal{N}(y_n|\boldsymbol{\beta}^\mathsf{T}\mathbf{x}_n, \sigma^2)$$

- As it turns out, this is **equivalent** to minimizing the sum of squared errors!

Don't believe it?

Replace $\mathcal{N}(y_n|\boldsymbol{\beta}^\mathsf{T}\mathbf{x}_n, \sigma^2)$ in the expression above by the definition of the Gaussian, take the derivative w.r.t. β, set it to zero and solve for β.

- This is called **maximum likelihood estimation (MLE)**!
- It allows to find a **point estimate** for the parameters in a probabilistic model

Adding priors

- We have been assuming the coefficients $\boldsymbol{\beta}$ to be deterministic values, but...
  - What if we have some prior knowledge on the values of $\boldsymbol{\beta}$?
  - What if we wish $\hat{\boldsymbol{\beta}}$ not to be too large (i.e. prevent overfitting)?
- Model $\boldsymbol{\beta}$ as a random variable and assign it a (prior) distribution!

$$p(\boldsymbol{\beta}) = \mathcal{N}(\boldsymbol{\beta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

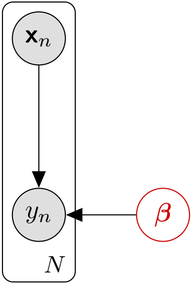
- **Prior distribution encodes our prior knowledge** about the values of the coefficients
- A typical choice is a Gaussian with zero mean and a diagonal covariance matrix with λ in the diagonal elements

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

- This encourages the values of $\boldsymbol{\beta}$ to be centered around zero with more or less variance depending on λ. But be careful:
  - Too large λ may lead **overfitting** (neutralizes the benefit of the prior)
  - Too small λ may lead **underfitting** (constrains the model too much)

Bayesian linear regression model

- Updated graphical model



- Updated generative process

  1. Draw coefficients $\beta \sim \mathcal{N}(\beta|\mathbf{0}, \lambda\mathbf{I})$
  2. For each feature vector $\mathbf{x}_n$
     a. Draw target $y_n \sim \mathcal{N}(y_n|\beta^\mathsf{T}\mathbf{x}_n, \sigma^2)$

- Joint probability distribution now factorizes as

$$p(\mathbf{y}, \beta|\mathbf{X}, \sigma, \lambda) = p(\beta|\lambda) \prod_{n=1}^{N} p(y_n|\beta, \mathbf{x}_n, \sigma)$$

$$= \underbrace{\mathcal{N}(\beta|\mathbf{0}, \lambda\mathbf{I})}_{\text{prior}} \times \underbrace{\prod_{n=1}^{N} \mathcal{N}(y_n|\beta^\mathsf{T}\mathbf{x}_n, \sigma^2)}_{\text{likelihood}}$$

Inference

- **Goal:** compute **posterior** distribution on β
- Following Bayes' theorem

$$\underbrace{p(\boldsymbol{\beta}|\mathbf{y}, \mathbf{X}, \sigma, \lambda)}_{\text{posterior}} \propto \underbrace{\mathcal{N}(\boldsymbol{\beta}|\mathbf{0}, \lambda\mathbf{I})}_{\text{prior}} \times \underbrace{\prod_{n=1}^{N} \mathcal{N}(y_n|\boldsymbol{\beta}^{\mathsf{T}}\mathbf{x}_n, \sigma^2)}_{\text{likelihood}}$$

- We can find an analytical solution to this - exact inference is possible!
- We will cover inference methods later in the course...
- For now, STAN/Pyro will take care of it for us :-)

Model estimation: MAP

- Alternatively to computing the posterior distribution on β, we can find a point estimate by maximizing the (log) joint probability of the model

$$\hat{\beta} = \arg\max_{\boldsymbol{\beta}} \log \left(\mathcal{N}(\beta|\mathbf{0}, \lambda\mathbf{I}) \prod_{n=1}^{N} \mathcal{N}(y_n|\beta^{\mathsf{T}}\mathbf{x}_n, \sigma^2) \right)$$

$$= \arg\max_{\boldsymbol{\beta}} \left(\log \mathcal{N}(\beta|\mathbf{0}, \lambda\mathbf{I}) + \sum_{n=1}^{N} \log \mathcal{N}(y_n|\beta^{\mathsf{T}}\mathbf{x}_n, \sigma^2) \right)$$

- This is called **maximum-a-posteriori (MAP)** estimation

Note

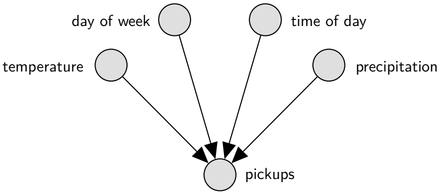
This is just like the MLE estimator plus a new term: $\log \mathcal{N}(\beta|\mathbf{0}, \lambda\mathbf{I})$. It penalizes the coefficients β for getting too large (overfitting). This is called **regularization**. See the derivation in Appendix.

Playtime!

- Ancestral sampling from linear regression model
  - See “05 - Regression models - Part 1.ipynb" notebook
  - Expected duration: 15 minutes
- Linear regression model of taxi pickups in NYC
  - See “05 - Regression models - Part 2.ipynb" notebook
  - Do section 2.1
  - Expected duration: 1 hour

Going back to our taxi demand case study...

- Let's revise (again) the **modeling assumptions** that we made



- What distribution should we assign to the pickups variable?
    - Gaussian
- But is this really the most appropriate distribution in this case?
    - Number of pickups is a count: $y_n \in \mathbb{N}$
    - A common distribution for modelling count data is the **Poisson**

Poisson regression

- "Poisson distribution expresses the probability of a given number of events occurring in a fixed interval of time and/or space if these events occur with a **known constant rate** and independently of the time since the last event"
- Sounds appropriate to model taxi pickups, but the rate is both **unknown** and **non-constant**...
    - As we did for the Gaussian, we can make the rate of the Poisson linearly dependent on the features $\mathbf{x}$

$$y_n \sim \text{Poisson}(y_n | \boldsymbol{\beta}^\mathsf{T} \mathbf{x}_n)$$

    - But this allows for negative rates: $\boldsymbol{\beta}^\mathsf{T} \mathbf{x}_n \in (-\infty, \infty)$
    - Use exponential transformation to ensure non-negativity! $e^{\boldsymbol{\beta}^\mathsf{T} \mathbf{x}_n} \in (0, \infty)$

$$y_n \sim \text{Poisson}(y_n | e^{\boldsymbol{\beta}^\mathsf{T} \mathbf{x}_n})$$

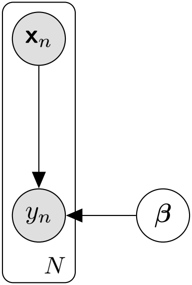
    - This is called a **link function**. In this case, a *log* link function

Note

Link functions get their names from the inverse of the transformation.

Bayesian poisson regression model

- Graphical model looks the same as before



- Updated generative process
  1. Draw coefficients $\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\beta}|\mathbf{0}, \lambda\mathbf{I})$
  2. For each feature vector $\mathbf{x}_n$
     a. Draw target $y_n \sim \text{Poisson}(y_n|e^{\boldsymbol{\beta}^\mathsf{T}\mathbf{x}_n})$

- Joint probability distribution becomes

$$p(\mathbf{y}, \boldsymbol{\beta}|\mathbf{X}, \lambda) = p(\boldsymbol{\beta}|\lambda) \prod_{n=1}^{N} p(y_n|\boldsymbol{\beta}, \mathbf{x}_n)$$

$$= \underbrace{\mathcal{N}(\boldsymbol{\beta}|\mathbf{0}, \lambda\mathbf{I})}_{\text{prior}} \times \underbrace{\prod_{n=1}^{N} \text{Poisson}(y_n|e^{\boldsymbol{\beta}^\mathsf{T}\mathbf{x}_n})}_{\text{likelihood}}$$

Inference

- **Goal:** compute **posterior** distribution on β
- Following Bayes' theorem

$$\underbrace{p(\boldsymbol{\beta}|\mathbf{y},\mathbf{X},\lambda)}_{\text{posterior}} \propto \underbrace{\mathcal{N}(\boldsymbol{\beta}|\mathbf{0},\lambda\mathbf{I})}_{\text{prior}} \times \underbrace{\prod_{n=1}^{N} \text{Poisson}(y_n|e^{\boldsymbol{\beta}^\mathsf{T}\mathbf{x}_n})}_{\text{likelihood}}$$

- Exact inference is no longer tractable
- Must resort to **approximate inference** methods
- Not a problem for STAN/Pyro :-)

Playtime!

- Poisson regression model of taxi pickups in NYC
- See “05 - Regression models - Part 2.ipynb" notebook
- Do part 2.2
- Expected duration: 30 minutes

Going back to the modelling assumptions...

- Suppose that Gaussian was indeed the most appropriate distribution for the target variable y

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

- What if our observations of y had **non-constant noise**?
- Consider the problem of modelling traffic speed data from probe vehicles
- The assumption of constant observation noise σ^2 might be too strong!

Figure: Traffic speeds in a road segment in Nørreport

Heteroscedastic regression

- We can relax the constant observation noise assumption by making the variance (linearly) dependent on a set of arbitrary features $\mathbf{u}$ (e.g. time of the day)

$$y_n \sim \mathcal{N}(y_n | \boldsymbol{\beta}^\mathsf{T} \mathbf{x}_n, e^{\boldsymbol{\eta}^\mathsf{T} \mathbf{u}_n})$$

  where $\boldsymbol{\eta}$ is a new set of coefficients to parameterize the relation between the features $\mathbf{u}$ and the observation noise

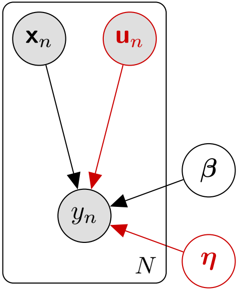
- As with the Poisson, we use a log link function to ensure non-negative variances

$$e^{\boldsymbol{\eta}^\mathsf{T} \mathbf{u}_n} \in (0, \infty)$$

- This allows to account for things like **time-varying observation noise** and produce better **uncertainty estimates** for the predictions!
  - Useful to know how reliable the predictions are

Bayesian heteroscedastic regression model

- Updated graphical model



- Updated generative process

  1. Draw coefficients $\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\beta}|\mathbf{0}, \lambda\mathbf{I})$
  2. Draw coefficients $\boldsymbol{\eta} \sim \mathcal{N}(\boldsymbol{\eta}|\mathbf{0}, \tau\mathbf{I})$
  3. For the n^{th} observation

     a. Draw target $y_n \sim \mathcal{N}(y_n|\boldsymbol{\beta}^{\mathsf{T}}\mathbf{x}_n, e^{\boldsymbol{\eta}^{\mathsf{T}}\mathbf{u}_n})$

- Joint probability distribution becomes

$$p(\mathbf{y}, \boldsymbol{\beta}, \boldsymbol{\eta}|\mathbf{X}, \mathbf{Z}, \lambda, \tau) = \underbrace{p(\boldsymbol{\beta}|\lambda)\, p(\boldsymbol{\eta}|\tau)}_{\text{priors}} \underbrace{\prod_{n=1}^{N} p(y_n|\boldsymbol{\beta}, \boldsymbol{\eta}, \mathbf{x}_n, \mathbf{u}_n)}_{\text{likelihood}}$$

Playtime!

- Heteroscedastic model of taxi pickups in NYC
- See "05 - Regression models - Part 2.ipynb" notebook
- Do part 2.3

Beyond linearity...

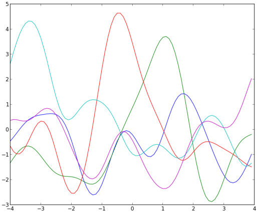
- So far we have been assuming a linear relationship between y_n and $\mathbf{x}_n$, such that

$$y_n = f(\mathbf{x}_n) + \epsilon$$

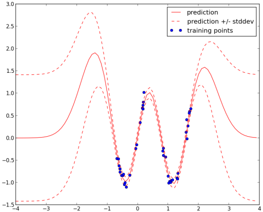
  where $f(\mathbf{x}_n) = \boldsymbol{\beta}^\mathsf{T} \mathbf{x}_n$
- As previously explained this is far more powerful than it looks at first sight!
- However, in some cases, it might still not be enough... But what can we do?
- **Gaussian processes** (GPs) allow us to model **non-linear** relationships!
  - **Non-parametric** models
  - Provide a probability distribution over functions
  - Place Gaussian process prior on the function f: $f \sim \mathcal{GP}$
  - GP prior specifies characteristics of the function, like stationarity, smoothness, periodicity, etc.
  - We will talk about GPs later on in the course! :-)
  - "Gaussian Processes for Machine Learning" book is a great resource[1]
  - Also, check out the STAN manual if you're interested

[1] http://www.gaussianprocess.org/gpml/

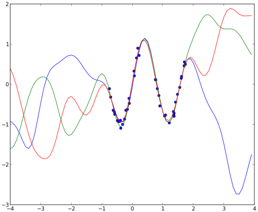
Beyond linearity...



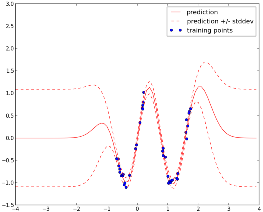
(a) samples from the GP prior



(b) predictive posterior



(c) samples from the GP posterior



(d) pred. post. after hyper-param. optimization

Beyond linearity...

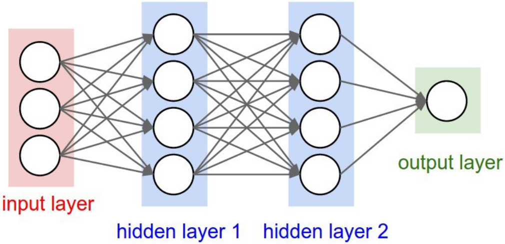
- So far we have been assuming a linear relationship between y_n and $\mathbf{x}_n$, such that

$$y_n = f(\mathbf{x}_n) + \epsilon$$

  where $f(\mathbf{x}_n) = \boldsymbol{\beta}^{\mathsf{T}} \mathbf{x}_n$
- We can assume $f(\mathbf{x}_n)$ to be a complex **deep neural network (DNN)**!
  - In fact, we can parametrize any exponential family distribution using a DNN - check out, e.g.: *Deep Exponential Families*[2]
  - Intersection between Deep Learning and Bayesian methods is currently a very popular research topic!
- We can **combine PGMs with DNNs**!
  - Variables in a PGM can be (part of) the input to a DNN
  - Output variables of a DNN can be part of a PGM

[2] Paper: `https://arxiv.org/abs/1411.2581`

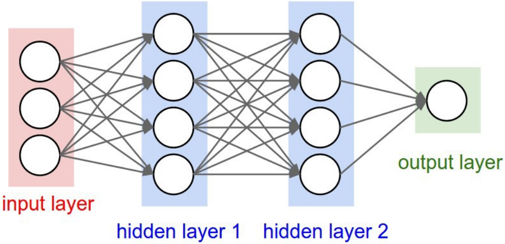
Crash course on (fully-connected/dense) Neural Networks



- Looks familiar doesn't it? :-)
  You can think of a neural networks as a PGM with some special characteristics!
- Each node is called a **neuron** and it computes a non-linear function of its inputs
- There is a weight w associated with each connection between neurons
- We **stack multiple layers** to obtain increasingly complex functions of the inputs
- More complex architectures exist, but are out of the scope of this course[3]

[3] Check "Deep Learning" course: `http://kurser.dtu.dk/course/02456`

Crash course on (fully-connected/dense) Neural Networks



- A neuron $h_i^{(l)}$ in layer l computes a weighted sum of its inputs from the previous layer $l-1$, and passes the result through a non-linearity (e.g. sigmoid, tanh,...)

$$h_i^{(l)} = \tanh\left(b_i^{(l)} + \sum_j w_{i,j}\, h_j^{(l-1)}\right)$$

where $b_i^{(l)}$ is a bias parameter and $w_{i,j}$ is the weight of the connection between $h_i^{(l)}$ and $h_j^{(l-1)}$

- More compactly using vector notation: $\mathbf{h}^{(l)} = \tanh(\mathbf{b}^{(l)} + \mathbf{W}^{(l)}\, \mathbf{h}^{(l-1)})$

Combining PGMs with neural networks

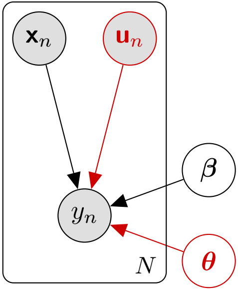
- A very simple practical example:

$$y_n = f_{\text{linear}}(\mathbf{x}_n) + f_{\text{nnet}}(\mathbf{u}_n) + \epsilon$$

where $f_{\text{linear}}(\mathbf{x}_n) = \beta^{\mathsf{T}}\mathbf{x}_n$ and $f_{\text{nnet}}(\mathbf{u}_n)$ is a DNN with parameters denoted by $\boldsymbol{\theta}$ (i.e. $\boldsymbol{\theta}$ includes all bias parameters and weights of the DNN)

- Updated generative process

1. Draw linear coefficients $\beta \sim \mathcal{N}(\beta|\mathbf{0}, \lambda\mathbf{I})$
2. Draw DNN parameters $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \tau\mathbf{I})$
3. For the n^{th} observation
   a. Draw target $y_n \sim \mathcal{N}(y_n|f_{\text{linear}}(\mathbf{x}_n) + f_{\text{nnet}}(\mathbf{u}_n), \sigma^2)$



Playtime!

- Combining PGMs with neural networks
- See “05 - Neural Network + Linear PGM - STAN.ipynb" notebook

Appendix: Gaussian prior and regularization

derivation

- We have the following model (Bayesian linear regression with Gaussian prior):

$$p(\mathbf{y}, \beta | \mathbf{X}, \sigma, \lambda) = \mathcal{N}(\beta | \mathbf{0}, \lambda \mathbf{I}) \times \prod_{n=1}^{N} \mathcal{N}(y_n | \boldsymbol{\beta}^{\mathsf{T}} \mathbf{x}_n, \sigma^2)$$

- We want to prove that the Gaussian prior component makes the MAP solution equivalent to the L2 regularization of the least squares solution. I.e.:

$$\arg\max_{\boldsymbol{\beta}} \log \left(\mathcal{N}(\beta | \mathbf{0}, \lambda \mathbf{I}) \prod_{n=1}^{N} \mathcal{N}(y_n | \boldsymbol{\beta}^{\mathsf{T}} \mathbf{x}_n, \sigma^2) \right) = \arg\min_{\boldsymbol{\beta}} \sum_{n=1}^{N} (y_n - \boldsymbol{\beta}^{\mathsf{T}} \mathbf{x}_n)^2 + \gamma \boldsymbol{\beta}^{\mathsf{T}} \boldsymbol{\beta}$$

- For all $\lambda > 0$, and $\gamma = \frac{\sigma^2}{\lambda}$.

Appendix: Gaussian prior and regularization

derivation

- For simpler notation, let's assume $\hat{y}_n = \boldsymbol{\beta}^\mathsf{T}\mathbf{x}_n$
- Let's start by calculating the $\log\, p(\mathbf{y}, \boldsymbol{\beta}|\mathbf{X}, \sigma, \lambda)$

$$\log\, p(\mathbf{y}, \boldsymbol{\beta}|\mathbf{X}, \sigma, \lambda) = log\Big(\mathcal{N}(\boldsymbol{\beta}|\mathbf{0}, \lambda\mathbf{I}) \times \textstyle\prod_{n=1}^{N} \mathcal{N}(y_n|\boldsymbol{\beta}^\mathsf{T}\mathbf{x}_n, \sigma^2)\Big)$$

$$= \log \frac{1}{(2\pi|\lambda\mathbf{I}|)^{\frac{1}{2}}} e^{\left(-\frac{1}{2}(\boldsymbol{\beta}^\mathsf{T}(\lambda\mathbf{I})^{-1}\boldsymbol{\beta})\right)} + \textstyle\sum_{n=1}^{N} \log \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} e^{(-\frac{1}{2\sigma^2}(\hat{y}_n - y_n)^2)}$$

$$= \underbrace{\log \frac{1}{(2\pi|\lambda\mathbf{I}|)^{\frac{1}{2}}}}_{const.} - \frac{1}{2}(\boldsymbol{\beta}^\mathsf{T}(\lambda\mathbf{I})^{-1}\boldsymbol{\beta}) + \cancel{N\log 1} - \underbrace{\frac{N}{2}\log(2\pi\sigma^2)}_{const.} - \textstyle\sum_{n=1}^{N} \frac{(\hat{y}_n - y_n)^2}{2\sigma^2}$$

$$= -\frac{1}{2}(\boldsymbol{\beta}^\mathsf{T}(\lambda\mathbf{I})^{-1}\boldsymbol{\beta}) - \textstyle\sum_{n=1}^{N} \frac{(\hat{y}_n - y_n)^2}{2\sigma^2} + const.$$

Appendix: Gaussian prior and regularization

derivation

- So, we have:
  $\log\ p(\mathbf{y}, \boldsymbol{\beta} | \mathbf{X}, \sigma, \lambda) = -\frac{1}{2}(\frac{\boldsymbol{\beta}^\mathsf{T}\boldsymbol{\beta}}{\lambda}) - \frac{1}{2\sigma^2}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2 + const.$
- ...and we want to calculate $\arg\max_{\boldsymbol{\beta}} \log\ p(\mathbf{y}, \boldsymbol{\beta} | \mathbf{X}, \sigma, \lambda)$

$$\arg\max_{\boldsymbol{\beta}} \log\ p(\mathbf{y}, \boldsymbol{\beta} | \mathbf{X}, \sigma, \lambda) = \arg\min_{\boldsymbol{\beta}} - \log\ p(\mathbf{y}, \boldsymbol{\beta} | \mathbf{X}, \sigma, \lambda)$$

$$= \arg\min_{\boldsymbol{\beta}} - \log\ p(\mathbf{y}, \boldsymbol{\beta} | \mathbf{X}, \sigma, \lambda) = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}(\frac{\boldsymbol{\beta}^\mathsf{T}\boldsymbol{\beta}}{\lambda}) + \frac{1}{2\sigma^2}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2$$
$$= \arg\min_{\boldsymbol{\beta}} \frac{1}{2\sigma^2}\left(\frac{\sigma^2}{\lambda}(\boldsymbol{\beta}^\mathsf{T}\boldsymbol{\beta}) + \sum_{n=1}^{N}(\hat{y}_n - y_n)^2\right)$$

- If we equate $\gamma = \frac{\sigma^2}{\lambda}$, and rewrite $\boldsymbol{\beta}^\mathsf{T}\mathbf{x}_n = \hat{y}_n$, we get the L2 regularized least squares formulation, as desired[4]:

$$\arg\min_{\boldsymbol{\beta}} \sum_{n=1}^{N}(y_n - \boldsymbol{\beta}^\mathsf{T}\mathbf{x}_n)^2 + \gamma\boldsymbol{\beta}^\mathsf{T}\boldsymbol{\beta}$$

Q.E.D.

[4] Note that this γ effectively corresponds to the usual λ parameter on the regularization literature, which is **not** the same as the λ in our prior, which is also common in literature!