Review

Reminders/Comments

- Thought questions due today
- If you are struggling with python or programming, keep practicing and try some tutorials; learning to program takes lots of practice and trial-and-error
- Review session: feel free to ask questions you did not ask before
    - someone else likely has the same question

Preface

- Our goal was to learn functions to provide good predictions
- Example, given information about a patient (x = (age, weight, height, previous conditions, etc.)) we want to predict
  - y = whether has cancer or not (classification)
  - y = the length of time until their next doctor's visit (regression)
- Our goal is to learn a function f so that f(x) returns an accurate value for the target y
- We used probabilities to reflect uncertainty in our predictions
  - due to the fact that there is variability in the targets y

Topics so far

- Basics of probabilities, including PMFs (discrete values) and PDFs (continuous values)

- Basics of parameter estimation: MAP and ML

- Generalized linear models
  - linear regression
  - logistic regression

- Generative and discriminative classifiers
  - naive Bayes (generative model)
  - logistic regression (discriminative model)

Exercise question

- Have joint distribution over many variables, p(**x**)
  - where **x** is a vector
- When has it been useful to know about independencies between the variables in **x**?
  - recall the definition that p(x1,x2) = p(x1)p(x2) means they are independent
- Why do we care about p(y | x)?

ML and MAP

- MAP: formalize problem using the posterior, select model

$$M_{MAP} = \arg\max_{M \in \mathcal{M}} \{p(M|\mathcal{D})\}$$

- For discrete space over M: p(M l D) is the PMF
- For continuous space over M: p(M l D) is the PDF
- ML: formalize problem using the likelihood, select model

$$M_{ML} = \arg\max_{M \in \mathcal{M}} \{p(\mathcal{D}|M)\} \,.$$

$$M_{\mathrm{MAP}} = \arg \max_{M \in \mathcal{M}} P(\mathcal{D}|M)p(M)$$

Exercise questions

- How might we pick p(x | M)?
- For p(M | D), is M a random variable? Is D a random variable?
- For w in a constrained space, say w in [-10,10], what is the relationship between MAP and ML?
- We typically assume iid data. What happens if we no longer assume iid data?

Thought exercise

- Why do we use maximum likelihood? Why not just write down a loss function, that wants to make the difference between f(x) and y smaller?
  - Sometimes we do in fact do that, but its good to know about both
  - How do we pick the loss function? Maximum likelihood tells us what the loss function should be
  - The idea of maximizing the likelihood of the model is a natural criteria, and allows a clearer connection to other more probabilistic approaches in machine learning
  - A made-up loss might have preferences that are not obvious; for maximum likelihood, we are being more explicit that the chosen model is the one that most likely generated the data
  - In some cases, we do change the loss function and it no longer explicitly corresponds to maximum likelihood; empirical risk minimization tries to take a more distribution-free approach

Different models

- Have considered the following algorithms
  - Linear regression
  - Logistic regression
  - Naive Bayes
  - SVMs (you will not be tested on this one)
- We train these models (functions) on a training set (batch of data with pairs (xi, yi))
  - How do we use these models for prediction on new data?
  - What are the properties of the trained models?

How to use linear regression

- Learn f(x) = <x, w> using linear regression solution

- For a new data point x, giving information about a patient, we predict y = how long it will be until their next visit

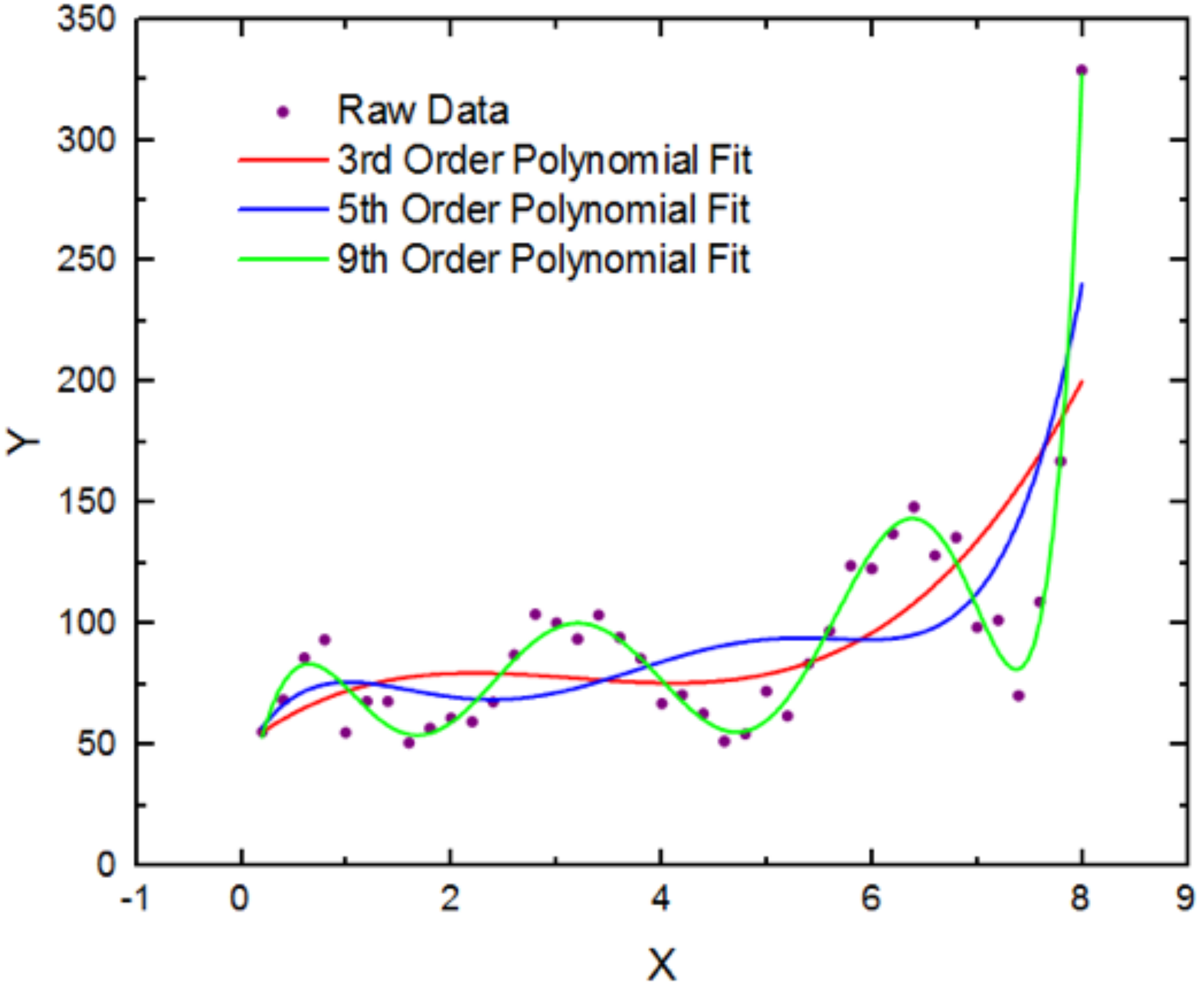
- How do you do so?

One way to move from linear to nonlinear functions

- We have focused on linear functions, <x, w>
- We have discussed how we can move to nonlinear functions by first transforming x, to get phi(x), and then learning a linear weighting on representation phi(x)
  - e.g., polynomial transformation (polynomial representation)
  - e.g., similarity transformation to prototypes (kernel representation)
- We looked at polynomial transformations for linear regression
- Can we do this for logistic regression as well?

Refresher on polynomial transformations

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ x^4 \\ x^5 \\ x^6 \\ x^7 \\ x^8 \\ x^9 \end{bmatrix}$$



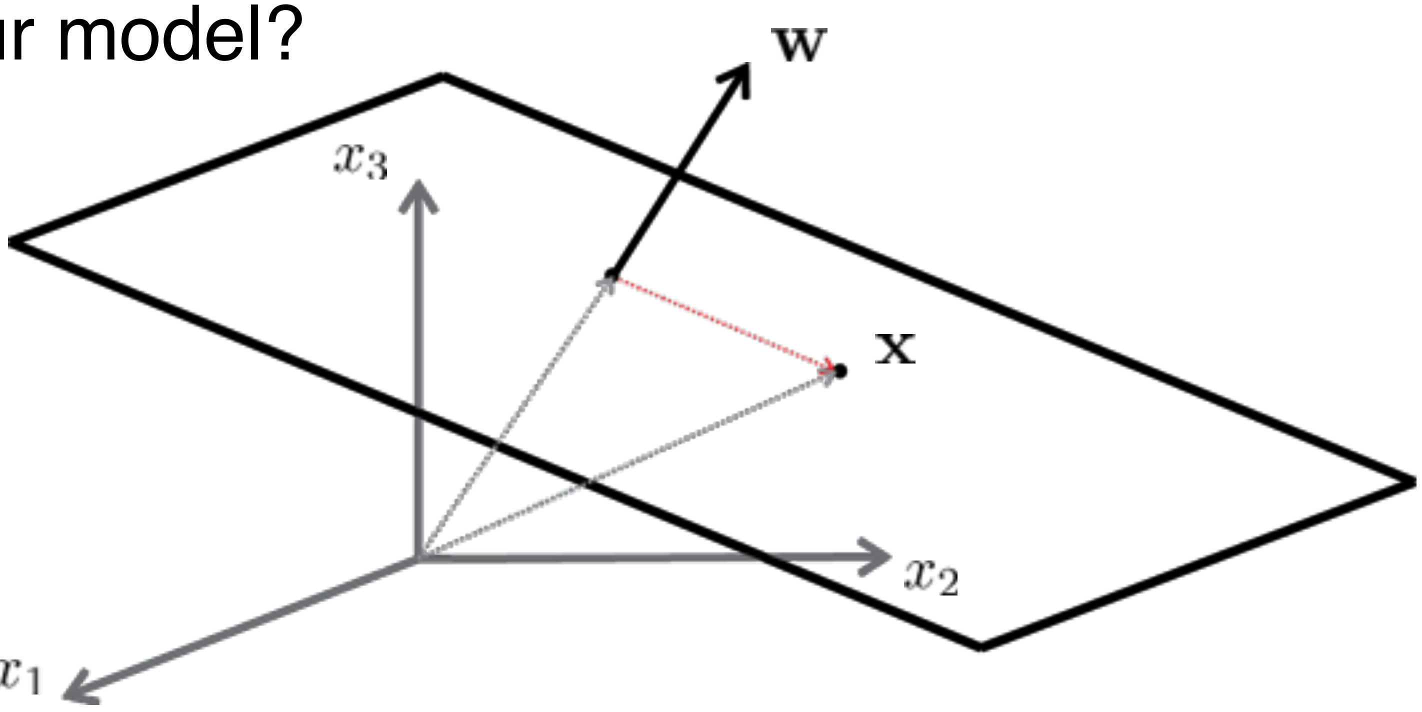
$$w_0 + w_1x^1 + w_2x^2 + \ldots + w_9x^9$$

Logistic regression

- Learn f(x) = sigmoid(< phi(x), w >) to approximate p(y = 1 l x)
  - learn hyperplane to separate points
  - <phi(x), w> > 0 means we classify point x as y = 1
  - <phi(x), w> < 0 means we classify point x as y = 0

- Now imagine we get a new instance, x and we want to predict if y = 0 or y = 1

- How do we use our model?



Overfitting

- Can we have overfitting issues with logistic regression, if we use a high-order polynomial representation?

- If so, can we use regularization?
  - Recall that for linear regression, we added regularization by considering MAP. We assumed p(y l x) was Gaussian, and then assumed a prior on w that corresponded to a regularizer
  - For logistic regression, we assumed p(y l x) is Bernoulli, and computed the maximum likelihood solution for a dataset D

How do we use naive Bayes?

- Recall that in binary classification we learned p(y=1|x)
  - This means p(y = 0 | x) = 1 - p(y=1 | x)
  - We pick y = 1 if p(y = 1 | x) is bigger (i.e., > 0.5)
  - We pick y = 0 if p(y=0 | x) is bigger (i.e., p(y=1|x) < 0.5)
- In naive Bayes, we use Bayes rule that states

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

- To use naive Bayes, we pick (why?)
  - y = 1 if p(x | y=1) p(y=1) is bigger
  - y = 0 if p(x | y=0) p(y=0) is bigger

Key assumption in naive Bayes

- Learning p(x l y) can be difficult, since features x can be complex
  - but really we do not need accurate p(x l y), we just need it to sufficiently separate instances into two clusters, for y = 0, y =1
  - if p(x l y = 0) is inaccurate for x, but it still correctly says p(x l y = 0)p(y=0) > p(x l y=1) p(y=1), then can still get accurate classification performance
- Aggressive assumption: assume features are independent, given class they are in p(x l y) = p(x1 l y) p(x2 l y) … p(xd l y)
- Learn (simpler) univariate distributions p(xj l y)
  - e.g., learn Gaussian for each xj, y = c, with parameters mu_{j,c} and sigma_{j,c}

Some questions

- What if we want to assume that our features are Poisson distributed? How does this change our approach?
- Can we use different distributions for different features?
- What if we have four classes, e.g., x is patient information and y = blood type (out of four blood types). For a new patient, how do we predict their blood type based on information about them?
- Why would we use naive Bayes instead of logistic regression?

Practical issues

- Collinear features making the closed form solution unstable for linear regression
    - can also make gradient descent updates behave more poorly
- Regularization to improve stability and avoid overfitting
- For huge datasets stochastic gradient descent more viable
- Regularization with l1 to enables feature selection; used a more specialized optimization technique to improve convergence to optimal solution with zeroed entries in w

Bias and variance

- What does it mean for an estimator to be unbiased?
  - We assume we have a true parameter/model theta*
  - We learn a model based on a given dataset, theta(D)
  - An unbiased estimator satisfies E[theta(D)] = theta*
- What does it mean for an estimator to be consistent?
  - In the limit, with more and more data, theta(D_n) —> theta*
  - This is asymptotically unbiased
- Why do we care?
  - want to understand properties of our estimators

Exercise: is the sample average unbiased?

$$\theta(D) = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\theta(D) = \frac{1}{n}\sum_{i=1}^{n} X_i$$

$$\begin{aligned}
\mathbb{E}[\theta(D)] &= \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} X_i\right] \\
&= \frac{1}{n}\mathbb{E}\left[X_1 + X_2 + \ldots X_n\right] \\
&= \frac{1}{n}\left(\mathbb{E}[X_1] + \mathbb{E}[X_2] + \ldots \mathbb{E}[X_n]\right) \\
&= \mu \quad \text{i.e., } \theta^*
\end{aligned}$$

Exercise: is the sample variance unbiased?

$$v(D) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \theta(D))^2$$

$$\mathbb{E}[v(D)] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(X_i - \theta(D))^2\right]$$

Bias-variance summary

$$
\begin{aligned}
\mathrm{MSE}(\mathbf{w}, \boldsymbol{\omega}) &= \mathbb{E}[||\mathbf{w} - \boldsymbol{\omega}||_2^2] \\
&= \mathbb{E}[\sum_{j=1}^{d} (\mathbf{w}_j - \boldsymbol{\omega}_j)^2] \\
&= \sum_{j=1}^{d} \mathbb{E}[(\mathbf{w}_j - \boldsymbol{\omega}_j)^2] \\
&= \sum_{j=1}^{d} \mathrm{Bias}(\mathbf{w}_j, \boldsymbol{\omega}_j)^2 + \mathrm{Var}(\mathbf{w}_j)
\end{aligned}
$$

$$\mathrm{Bias}(\mathbf{w}_j, \boldsymbol{\omega}_j) = \mathbb{E}[\mathbf{w}_j] - \boldsymbol{\omega}_j$$

l2 regularization trades off bias and variance

Optimization

- Once a problem is formalized, need to find the solution to that problem
- How can we find a solution?
- Is there a general strategy to always find a solution?
- What strategies could you use to find a solution?

Stationary points

- Our goal is to obtain stationary points, where gradient = 0
    - Local maximum
    - Local minimum
    - Saddle point
- Can find solution by taking gradient, setting to 0 and solving for desired parameters w
    - When do we know a closed form solution exists?
    - If does not exist, use iterative methods
- Then can determine what type of stationary point
    - What if you know the function is convex?
    - What if you know the function is concave?

Iterative methods

- Step in the direction of the gradient to reach a stationary point
- If maximizing, step in the direction of the gradient (gradient ascent)
- If minimizing, step in the negative direction of the gradient (gradient descent)
- How do we know this will reach a stationary point?
  - Example: minimize quadratic function
  - Example: maximize quadratic function

Efficient iterative methods

- Goal: reduce the number of iterations required to reach a stationary point

- First-order gradient descent uses information about function to direct the search (i.e., gradient information)

- Can reduce the number of iterations of gradient descent by more effectively choosing the step-size e.g., with line search

- Can reduce the number of iterations by using second-order methods rather than first-order methods

- Can better balance computation and number of iterations using stochastic gradient descent

Quiz example question

Given dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i = (x_{i1}, x_{i2})$, the objective is to find parameters of the function

$$f(\mathbf{x}) = \log(w_1 x_1 + w_2 x_2)$$

such that the weighted sum of squared errors between the target and the prediction is minimized, with some of the samples having higher importance than other samples according to importance weights $c_1, \ldots, c_n > 0$.

Provide an algorithm for obtaining the parameter vector $\mathbf{w} = (w_1, w_2)$. If there is a closed form solution, provide the closed-form solution; otherwise, provide an iterative, first-order gradient descent update.