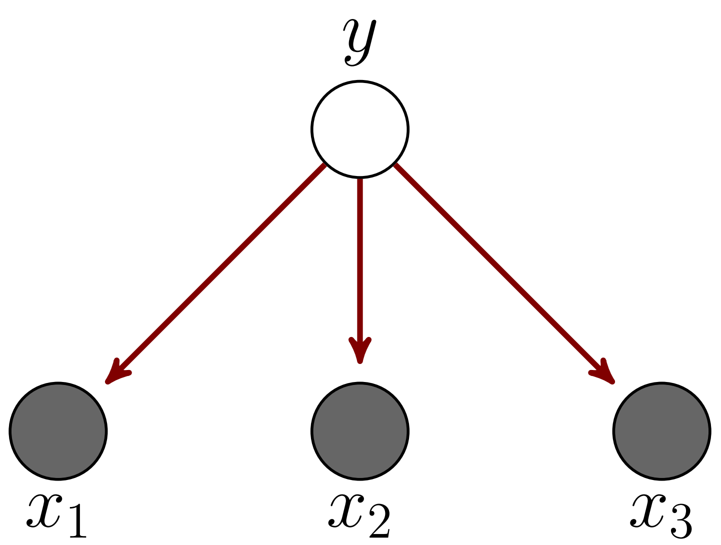
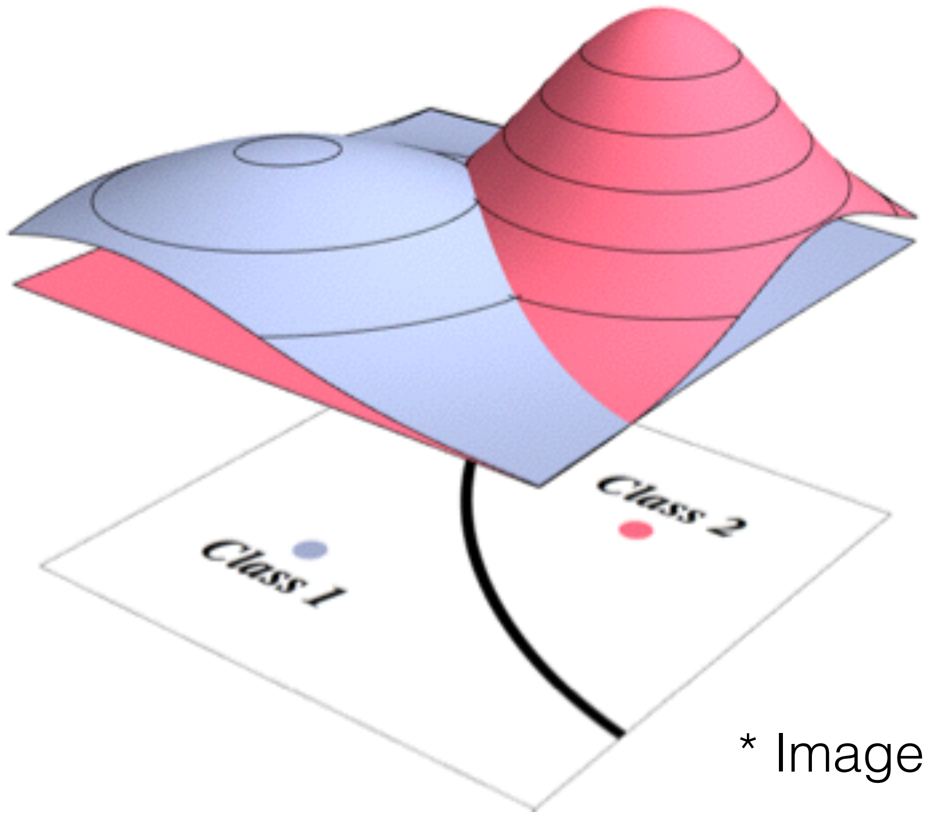
COMPUTER SCIENCE
INDIANA UNIVERSITY
School of Informatics and Computing
Bloomington



Logistic regression and Naive Bayes



* Image from Yaroslav Bulatov

Reminders/Comments

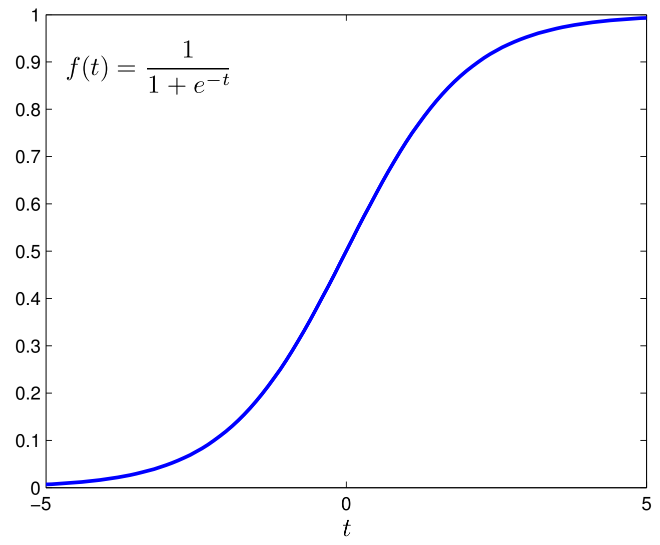
- Assignment 2 due next week
- Clarifications for notes and assignment
  - The **number of epochs** is the number of times we iterate through the dataset in stochastic gradient descent
  - Matching pursuit algorithm: simple idea of computing correlation to residual errors. Provided a paper with nice pseudocode, if confused

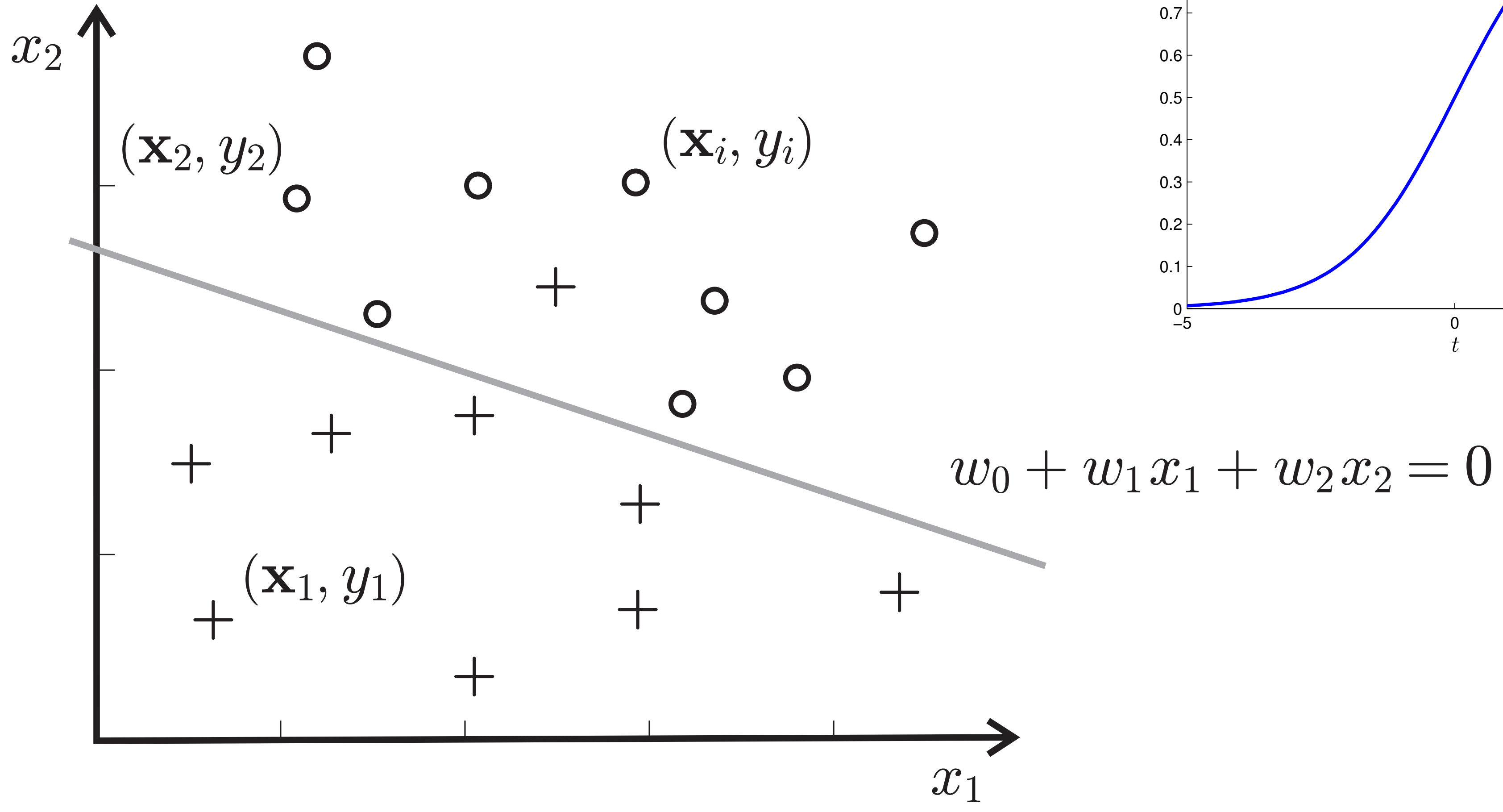
Thought question

- “Linear regression is very specific in the sense that Y is some kind of special function of X, now if we just throw too many independent variables into the mix, which may not be very smart, it may be difficult to find hidden patterns as the algorithm only look for linear, additive patterns among them. How do we avoid adding unnecessary variables in our algorithm? I think for linear regression to work we already should have a basic understanding of the data and thus decide our variables carefully, but is this the only way.”
    - This is largely the purpose of regularization
    - In particular, l1 regularization subselects features

Logistic regression is a linear classifier

- Hyperplane $\mathbf{w}^\top \mathbf{x} = 0$ separates the two classes
  - P(y=1 | x, w) > 0.5 only when $\mathbf{w}^\top \mathbf{x} \geq 0.$
  - P(y=0 | x, w) > 0.5 only when P(y=1 | x, w) < 0.5, which happens when $\mathbf{w}^\top \mathbf{x} < 0$





Logistic regression versus linear regression

- Why might one be better than the other? They both use linear approach.

Discriminative versus generative

- So far, have learned p(y l x) directly
- Could learn other probability models, and using probability rules to obtain p(y l x)
- In generative learning,
  - learn p(x l y) p(y) (which gives the joint p(x, y) = p(xly) p(y))
  - compute p(x l y) p(y), which is proportional to p(y l x)

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

- Question: how do we use p(xly) p(y) for prediction?

How to use generative model?

- Our decision rule for using these probabilities is the same as with logistic regression: pick class with the highest probability
- Assume you've learned p(x l y) and p(y) from a dataset
- Compute

$$
\begin{aligned}
f(\mathbf{x}) &= \arg\max_{y \in \mathcal{Y}} p(y|\mathbf{x}) \\
&= \arg\max_{y \in \mathcal{Y}} p(\mathbf{x}|y)p(y)/p(\mathbf{x}) \\
&= \arg\max_{y \in \mathcal{Y}} p(\mathbf{x}|y)p(y)
\end{aligned}
$$

Pros and Cons

- Discriminative
  - focus learning on the value we care about: p(y | x)
  - can be much simpler, particularly if features x complex, making p(x | y) difficult to model without strong assumptions
- Generative
  - can be easier for expert to encode prior beliefs, e.g., for classifying trees in evergreen or deciduous, structure/distribution over the features (height, location) can be more clearly specified by p(x | y), whereas p(y | x) does not allow this information to be encoded
  - can sample from the generative model
- naive Bayes can perform better in the small sample setting
  - see “On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes”, Ng and Jordan, 2002

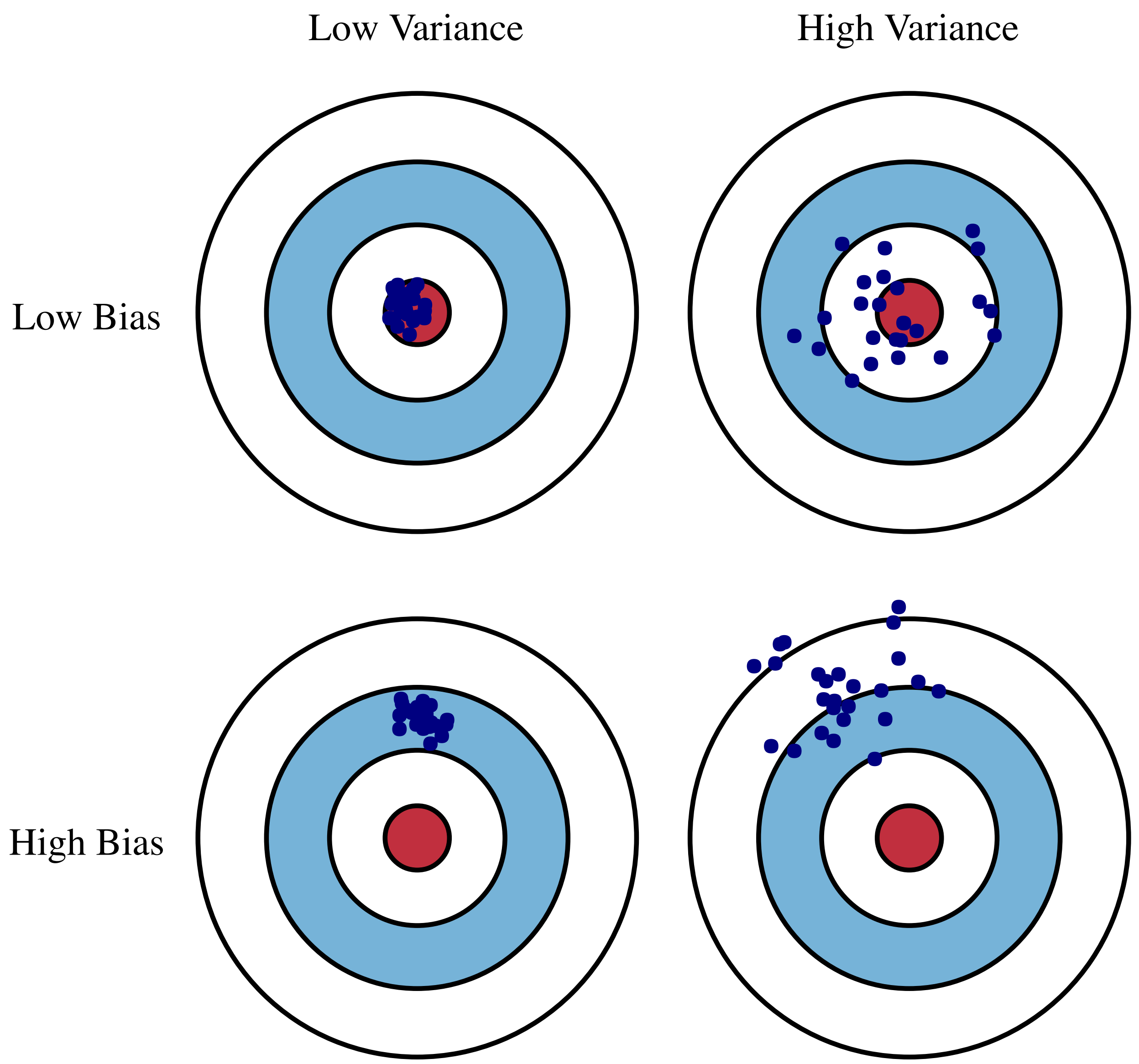
February 27

- Assignment 2 due this week
- Comment about optimization
  - can use first-order gradient descent
  - I have taught you about second-order methods, but you will not be tested on them, nor need to use them in your assignments
- Standard error is the sample standard deviation, divided by square-root of number of samples. For A2 the number of samples is the number of runs
  - each run gives an estimate of expected error for a learned weight vector on a training subsample
  - we want to know the variance in error across these learned w

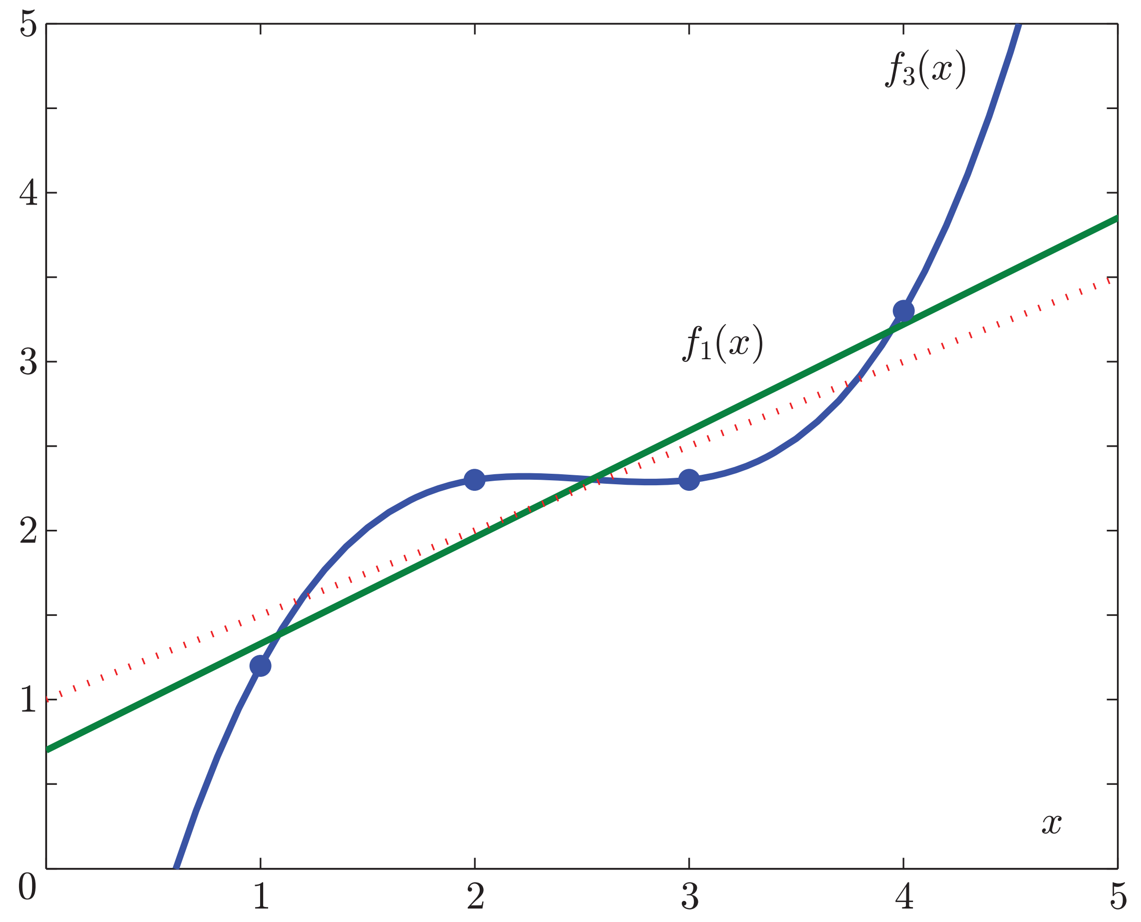
Though questions

- About overfitting:
  - Why would one want high bias, low variance?
  - Shouldn't providing a more accurate function of a higher degree of polynomial (that goes through more points) be better for future predictions? The example you gave doesn't make sense to me as it would appear that the model (a linear function) of the prior data was just a line of best fit, but none of the data points where on the line of the actual model.
  - Is overfitting due to a mismatch between train and test data? i.e., when training distribution is different from test

High-bias, low-variance



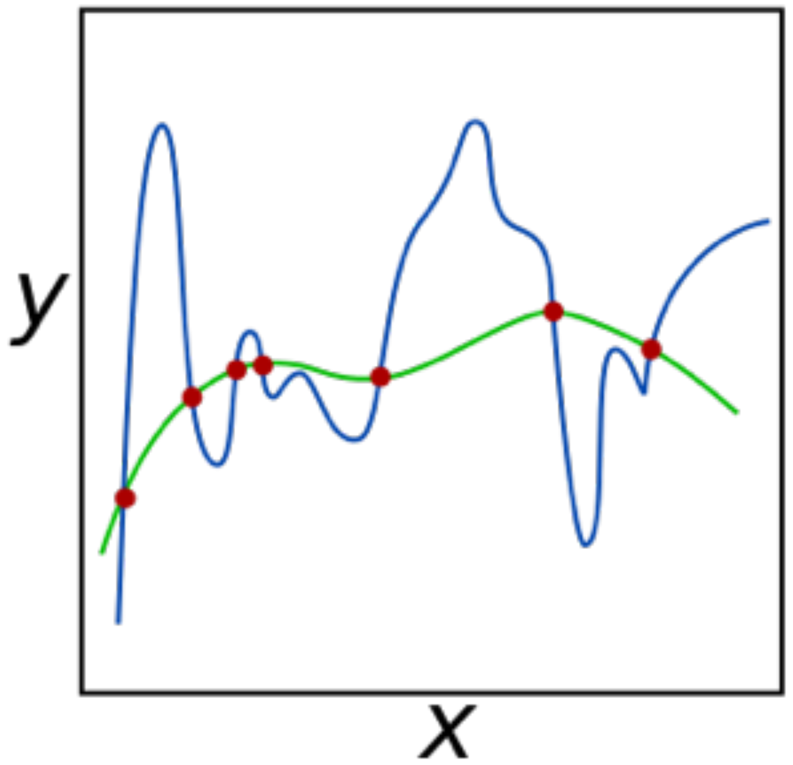
High-degree polynomial



- Red line is the true model
- Should the polynomial be better for future predictions?
- Why are none of the points on the true line (red line)?

Mismatch between train and test

- We do not usually refer to overfitting because of a mismatch between training and test data
- Usually assume train and test data is identically distributed
- Rather, if we completely fit to any subsample of data, even if it is representative, we could get overfitting
  - e.g., imagine memorizing training points (xi, yi), and just reporting yi if you see xi, and otherwise reporting 0



Why does regularization prevent overfitting?

- Imagine expand up features into high-dimensional polynomial
- Can fit points exactly by using these degrees of freedom to add and subtract different polynomial terms
    - e.g., yhat = 10*x1 - 10*x2, yhat not large but coefficients might be
- Discussed that X^T X might not be full rank, or it might be nearly singular
    - i.e., have singular values close to zero
- When those singular values are inverted, it produces really large values, so coefficient likely to be high magnitude
- llwll_2 prefers values near zero, adds lambda to singular values

How can we determine the order of the polynomial?

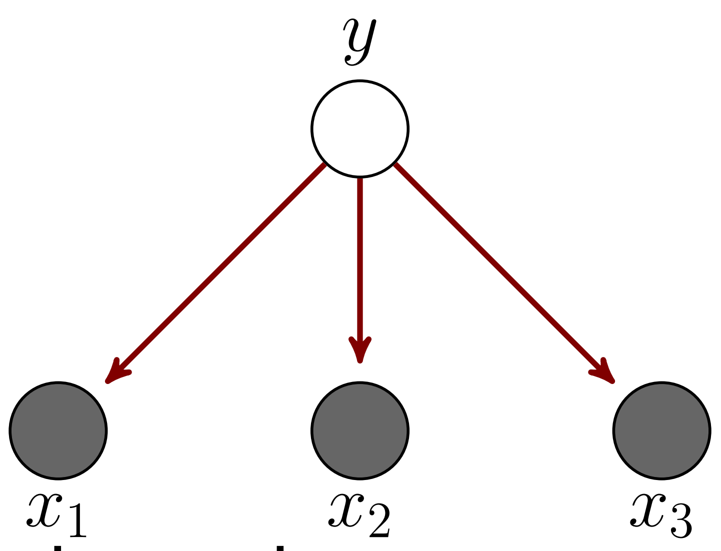
- If too high-order (e.g., 9th degree polynomial), might overfit
  - includes terms like x^9
- If too low-order (e.g., 2nd degree polynomial), might underfit
  - includes terms like x^2
- Do we need an expert to decide?
- Alternative: over-parametrize (high-degree), and use l1 regularization to sub-select features

Thought question

- Is it possible to provide a prediction and an estimate of how confident we are in that prediction?
  - “Let's say the data you are modeling is inherently high-variance and it may not be appropriate to give exact estimates for new data. Is it possible to generate a y-hat that is something of a bar, e.g. a line, linear or non-linear, but that defines a distribution range around it implying that data will likely fall within some N(mu, sigma^2) of the line?”
- When we learn p(y I x) with mu = <x, w>, we can also learn sigma to model how much y varies for a given x

Naive Bayes

$$p(\mathbf{x}|y) = \prod_{i=1}^{d} p(x_i|y).$$



- How do we realistically learn p(x l y)?
- One option is to make a (strong) conditional independence assumption: the features are independent given the label
- Example: given a patient does not have the disease (y=0), attributes about patient are uncorrelated (e.g., age & smokes)
  - even within a class, age and smokes could be correlated
- Surprisingly, despite the fact that this seems unrealistic, in practice this can work well
  - one hypothesis is we are running these algorithms on “easy” data
  - another is that dependencies skew the distribution equally across all classes, so no one class gets an increased probability

Types of features

- Before, we had to choose the distribution over y given x
  - e.g. p(y l x) is Gaussian for continuous y
  - e.g. p(y l x) is Poisson for y in {1, 2, 3, …}
  - e.g. p(y l x) is Bernoulli for y in {0,1}
- Parameters to p(y l x) had E[y l x] = f(xw)
- Now we need to choose the distribution over x given y; how do we pick p(x l y)? What are the parameters?

How do we pick distributions?

- For p(x), picked Gaussian, Bernoulli, Poisson, Gamma
    - depending on the values x could take, or looking at a plot
- How do we pick conditional distributions, like p(y l x)?
- General answer: based on the properties of y. Once the given features x are fixed, are are just learning a distribution over y

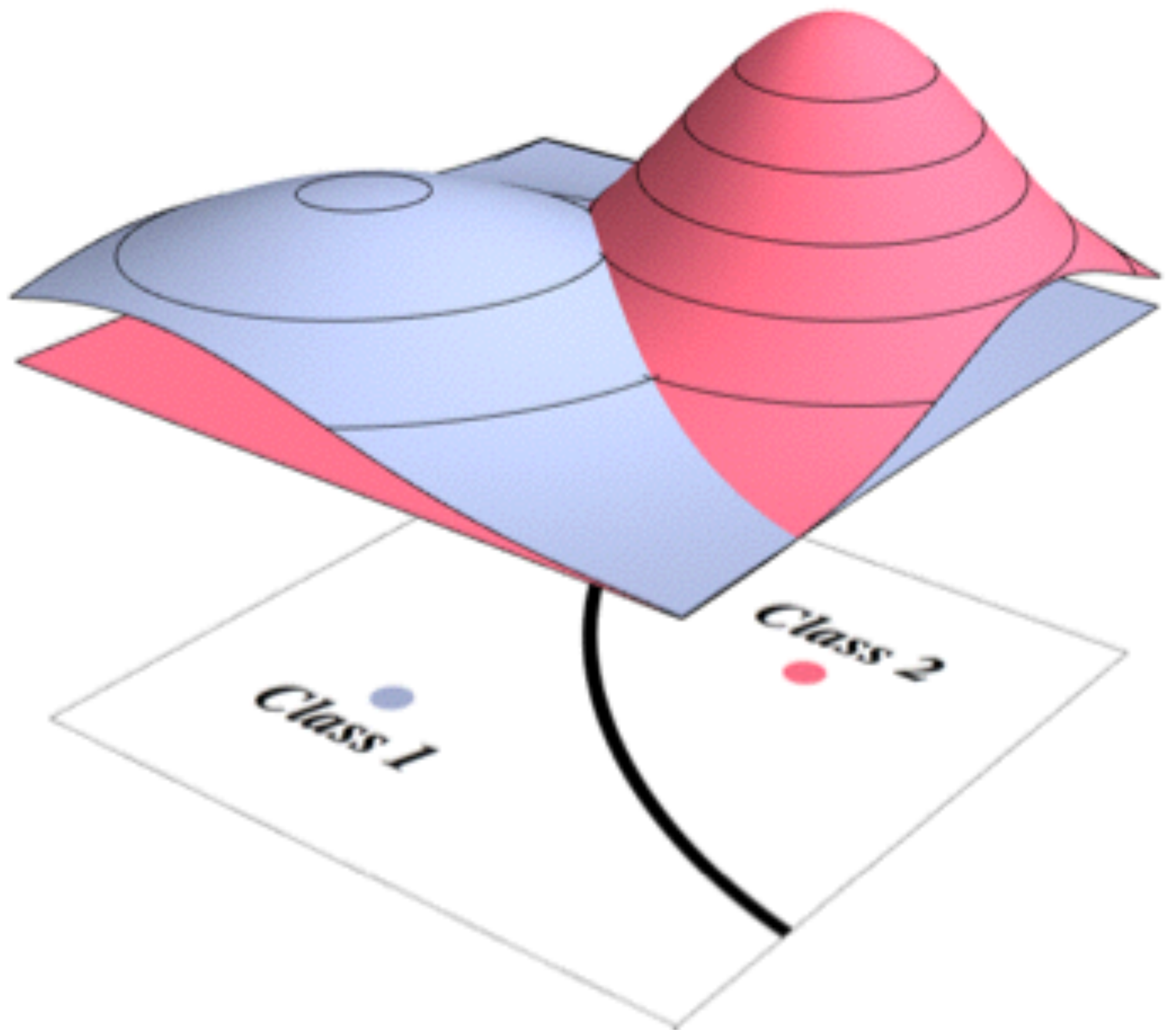
Exercise: binary features

- For binary features x in {0,1}, binary y in {0,1}, what is p(x l y)?
- How do we learn p(x l y)?
- How do we learn p(y)?

Exercise: continuous features

- For continuous features x, binary y in {0,1}, what could we choose for p(x l y)?
- What are the parameters and how do we learn them?

Continuous naive Bayes



* Image from Yaroslav Bulatov

Exercise

- Imagine someone gives you p(x l y) and p(y)
- They give you a test instance, with features x
  - e.g., x is an image, of pixel values (values between 0 to 255)
- Your goal is to predict the output y
  - e.g., if the image contains a cat or not
- How do you use p(x l y) and p(y) to make prediction?

Whiteboard

- Multiclass classification
  - Naive Bayes

March 1, 2017

- Deadline change for Assignment 2 to March 2, 11:59 p.m.
  - remove #import plotfcns from your file, as plotfcns not given to you
  - pseudocode for all questions in notes, except for MPRegression
- Thank you for the feedback
  - concerns about pace of course
  - more real-world examples
- Update to reading assignment
  - no longer have to read about factorization approaches
  - added a bit of reading on running experiments

Difficulty of course/assignments

- Assignments are meant to challenge you
- Quiz/Final will have simpler questions that can be answered in the allotted time
    - Final designed to take 1 hour, you are given 2 hours
- You will be allowed a 4 page cheat sheet
- Lectures are supposed to complement assignments
    - lectures are not really there for assignments, assignments are for you
- Additional reading materials, such as Bishop book or Barber book

A few clarifications

- Polynomial regression is not a type of GLM (generalized linear model)
    - The term Generalized means the generalization to the distribution
    - Logistic regression and Linear regression are examples of GLMs, with Bernoulli and Gaussian exponential family distributions respectively
- OLS and maximum likelihood with a Gaussian for p(y I x) is equivalent
- Bias unit (add an additional feature to feature vector that is always one) versus bias from regularizer, these are different
    - the term bias unit should really be called intercept unit

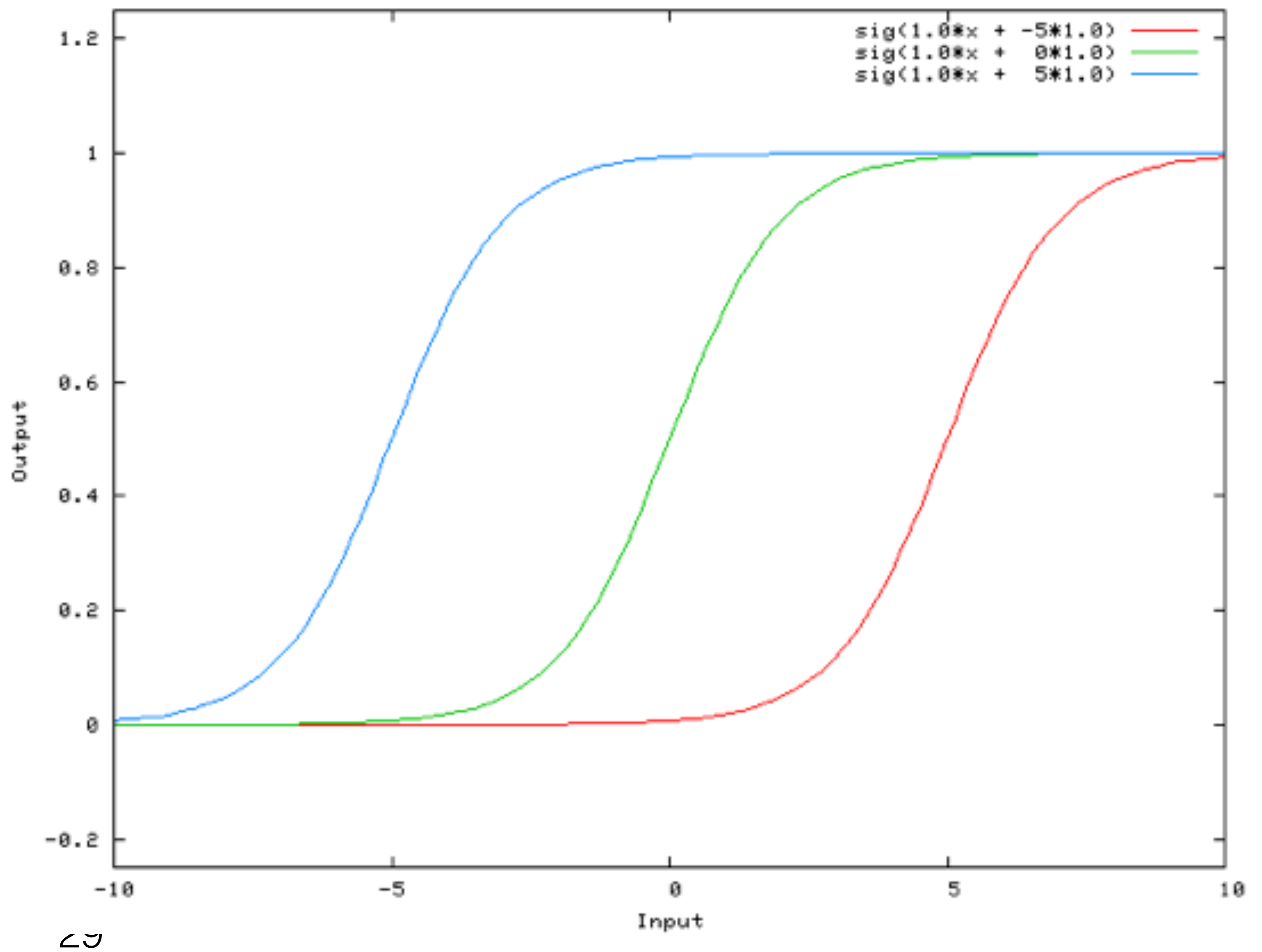
Though question

- In the linear classifiers section, it mentions that x_0 = 1 is used. Why is it better to use this bias as opposed to using an activation function which takes outputs of 0 into account?
- Additional question: what happens when we add an intercept unit (bias unit) to the features for logistic regression?

Adding a column of ones to X for logistic regression (and GLMs)

- This provides the same outcome as for linear regression

- g(E[y l x]) = x w —-> bias unit in x with coefficient w0 shifts the function left or right (where g = inverse of sigmoid)



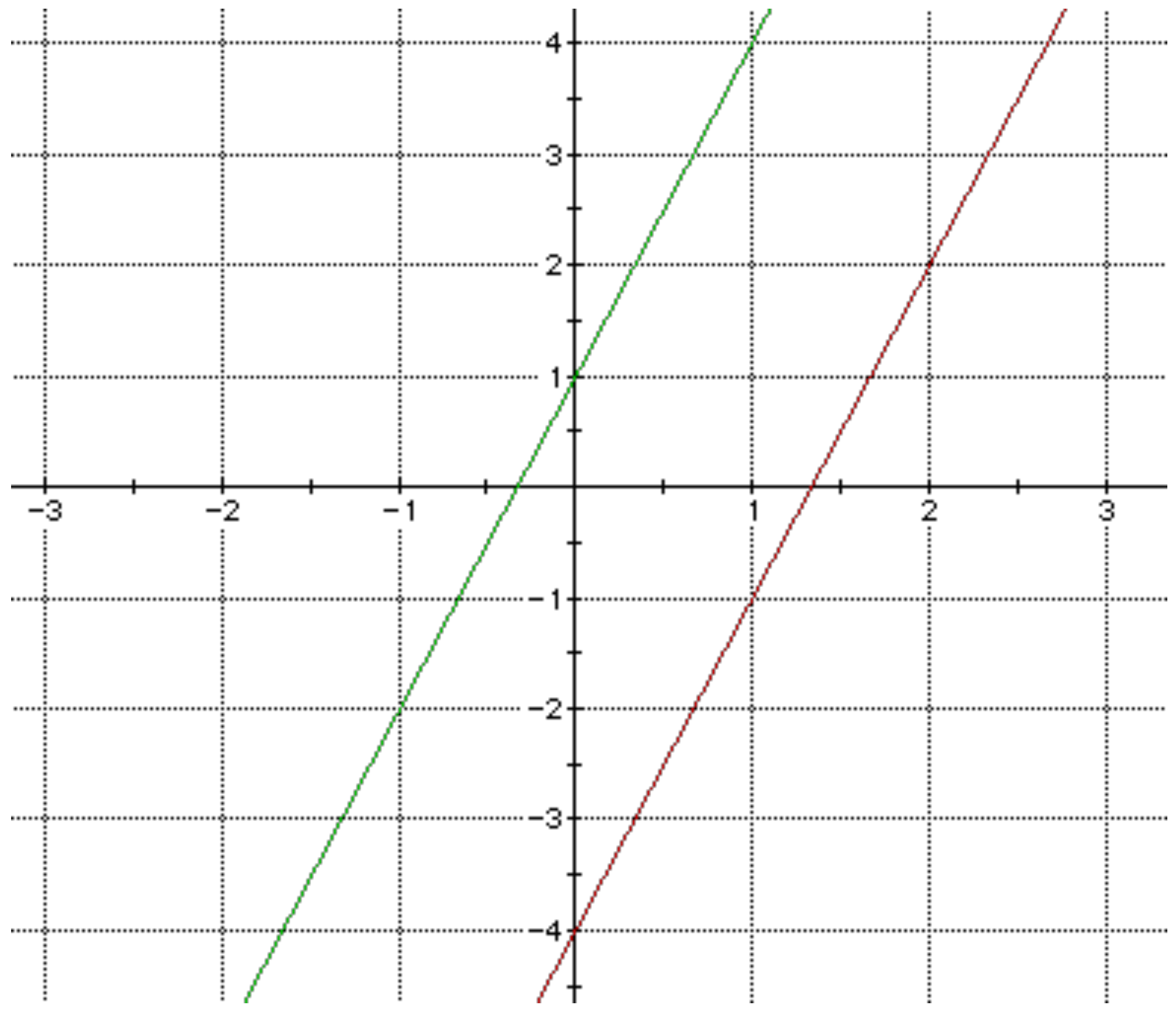


*Figure from http://stackoverflow.com/questions/2480650/role-of-bias-in-neural-networks

Feedback quiz

- Difference between logistic regression and linear regression
  - assume different distribution on y, given x: logistic uses Bernoulli for p(y lx) and linear uses Gaussian for p(ylx)
- Question about datasets
  - targets {-3.0, 2.2, -5.3, -1.0, 4.3} —> linear regression
  - targets {High, Low, High, High, Low} —> logistic regression, naive Bayes
  - targets {1, 2, 3, 2, 1} —> naive Bayes, since y always in these three categories (or multinomial logistic regression)

Whiteboard

- Complete naive Bayes

Practical considerations: incorporating new data

- How do we incorporate new data into naive Bayes?
- Learned mu_{j,c} and sigma_{j,c} for each feature j, each class c
- Keep a running average of mu_{j,c} and sigma_{j,c}, with number of samples n_{j,c} for feature j class c
- For a new sample (x, y), update parameters mu_{j,y} and sigma_{j,y} using:
  - mu_{j,y} = mu_{j,y} + (x_j - mu_{j,y})/ n_{j,c}
  - sndmoment_{j,y} = sndmoment_{j,y} + (x_j^2 - sndmoment_{j,y})/ n_{j,c}
  - sigma_{j,c} = sqrt(sndmoment_{j,y} - mu_{j,y}^2)

Practical considerations: incorporating new data

- Imagine you have a model (say weights w) and now want to incorporate new data
- How can you do this with regression approaches?
- Option 1: add data to your batch and recompute entire solution
- Option 2: start from previous w (warm start), add data to your batch and do a few gradient descent steps
- Option 3: use stochastic gradient descent update and step in the direction of the gradient for those samples
  - for a constant stream of data, will only ever use one sample and then throw it away

Stochastic gradient descent

Algorithm 2: Stochastic Gradient Descent$(\mathrm{Err}, \mathbf{X}, \mathbf{y})$

```
1: $\mathbf{w} \leftarrow$ random vector in $\mathbb{R}^d$
2: for $i = 1, \dots$ number of epochs do
3:     for $t = 1, \dots, n$ do
4:         // For many settings, we need step-size $\eta_t$ to decrease with time
5:         // For example, a common choice is $\eta_t = \eta_0 t^{-1}$ or
6:         // $\eta_t = \eta_0 t^{-1/2}$ for some initial $\eta_0$, such as $\eta_0 = 1.0$.
7:         $\mathbf{w} \leftarrow \mathbf{w} - \eta_t \nabla \mathrm{Err}_t(\mathbf{w}) = \mathbf{w} - \eta_t(\mathbf{x}_t^\top \mathbf{w} - y_t)\mathbf{x}_t$
8: return $\mathbf{w}$
```

For logistic regression

$$\nabla \mathrm{Err}_t(\mathbf{w}) = \mathbf{x}_t(\sigma(\mathbf{x}_t^\top \mathbf{w}) - y_t)$$

Stochastic gradient descent

- How do we decide when to stop? How do we select step-size?
- See Bottou's tricks for SGD: http://cilvr.cs.nyu.edu/diglib/lsml/bottou-sgd-tricks-2012.pdf
    - this is not a polished document, but useful anyway
- Convergence proof if step-sizes decreasing to zero, at a reasonable rate (that is not too quickly)
- In practice, additional approaches to improve convergence
    - monitor convergence by checking full training error periodically
    - many variance-reduced approaches now (SVRG)

Batch GD versus SGD

- What are some advantages of batch gradient descent?
  - less noisy gradient
  - more clear strategies for selecting stepsize
- What are some advantages of SGD?
  - more computationally efficient: does not waste an entire epoch to provide an update to the weights
  - convenient for updating current solution with new data
- Does batch always give better solutions than SGD?