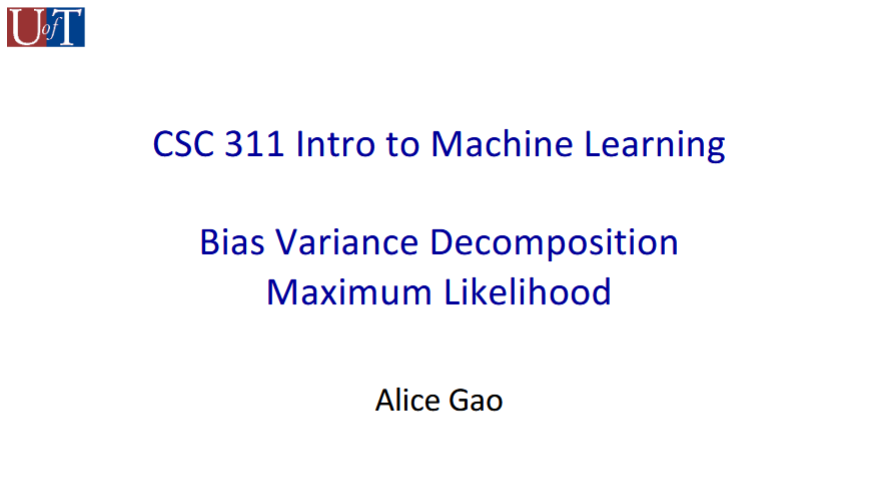
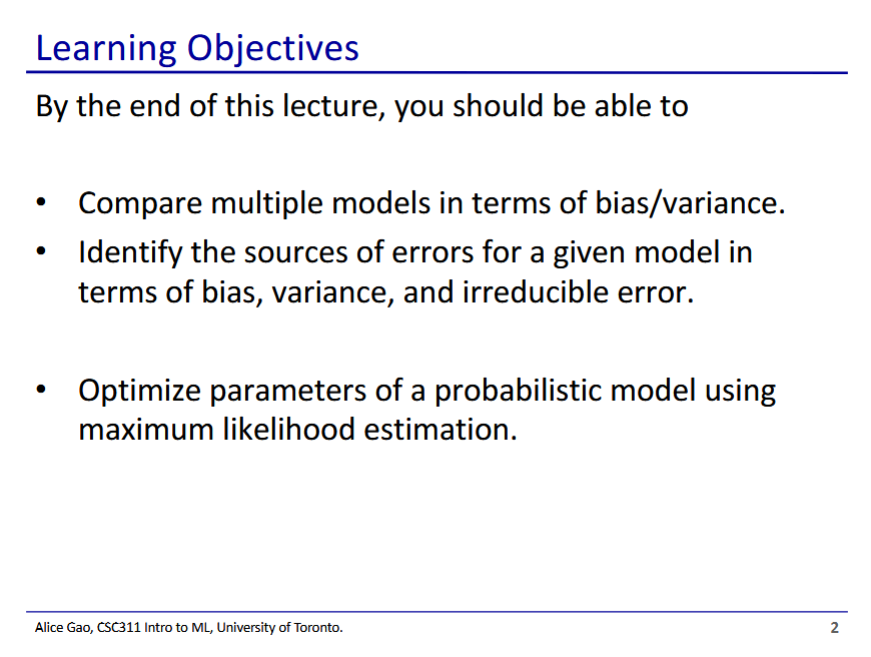
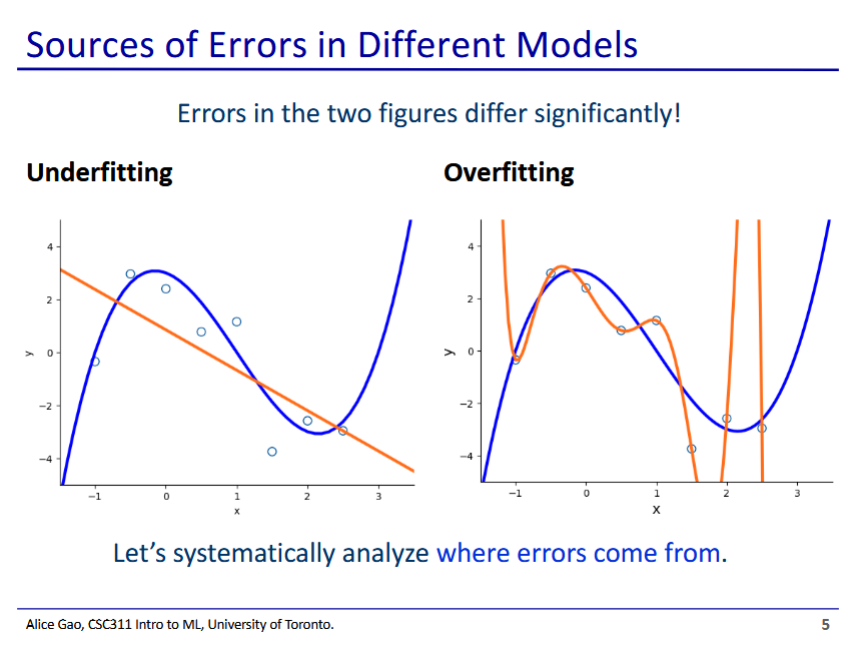
| **Admin stuff**   * Test   + Next friday   + Contents will be logistic regression, neural nets, backpropagation, and this weeks lecture (bayesian)   **Bias and variance**   * Variance - How much do predictions vary for different training sets?   + High variance leads to overfitting * Bias - How wrong is the average prediction for different training sets?   + High bias leads to underfitting * **See breakdown of error into bias and variance from slide 21-23**   + Testable content   **Maximum likelihood coin example**   * Maximum likelihood gives the probability that best accounts for the training data   + In this case, given n flips of a biassed coin, what is the bias on the coin? * A coin flip is a Bernoulli variable   + is the likelihood of flipping a head   + Individual probabilities:   + Combined probabilities: * Likelihood and log-likelihood loss functions   + **Likelihood function:**     - Function that gives the probability of seeing the training data given a probability   + **Log-likelihood function:**     - Taking the log does not change which gives us the max likelihood     - However it does make calculations simpler * **We want to find the which gives the maximum likelihood**   + To do this we find when the derivative of the log-likelihood function is 0     - = number of heads, = number of tails |
| --- |







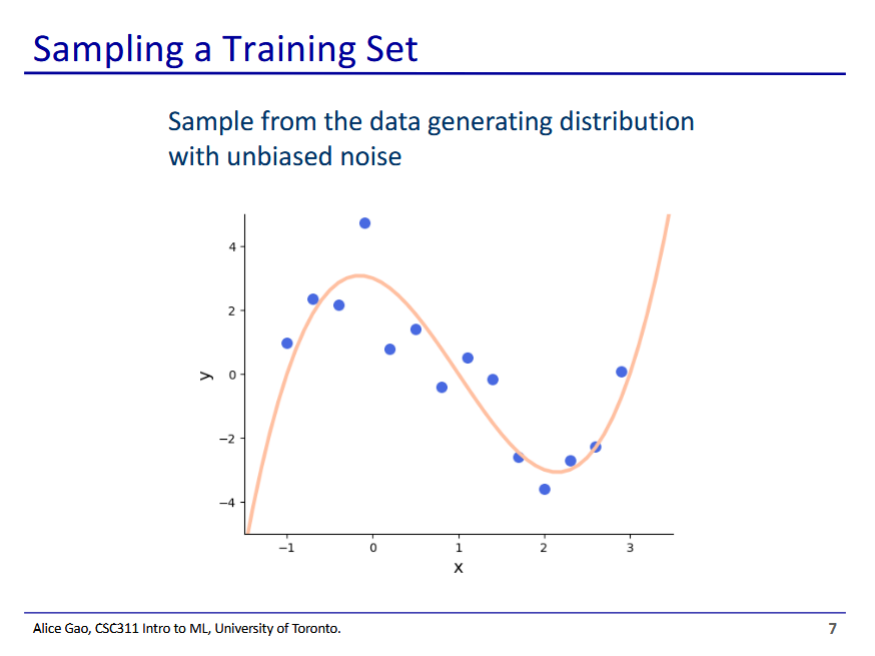




* Our desired distribution is the blue line, and our model is the orange line
* From looking at the images, we can get an intuitive idea about what each of these terms mean
  + But where do the errors actually come from?



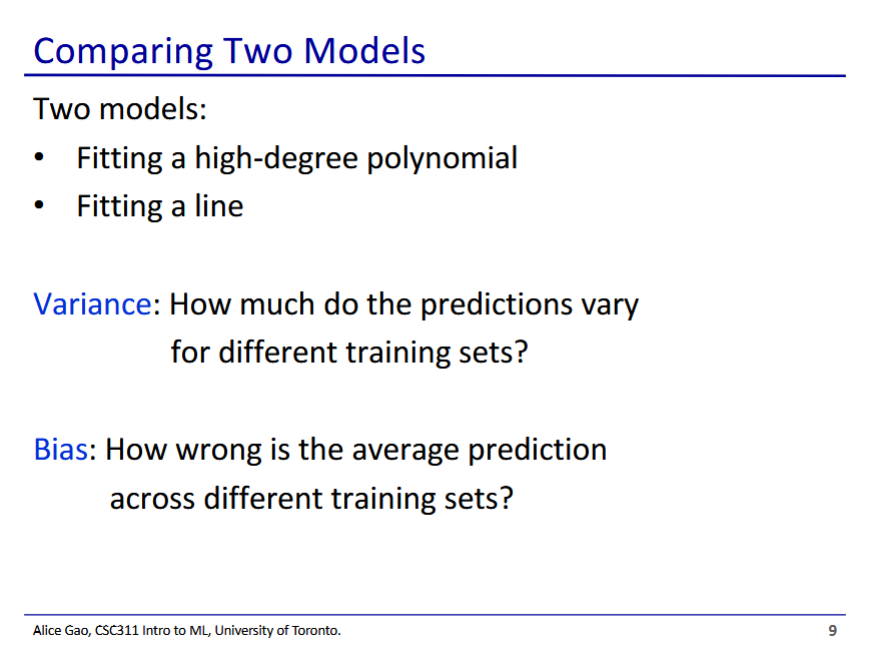
* We start with a data generating distribution - this is the true distribution that our training data gets sampled from
  + In reality we never get to see this true function, only points sampled from it



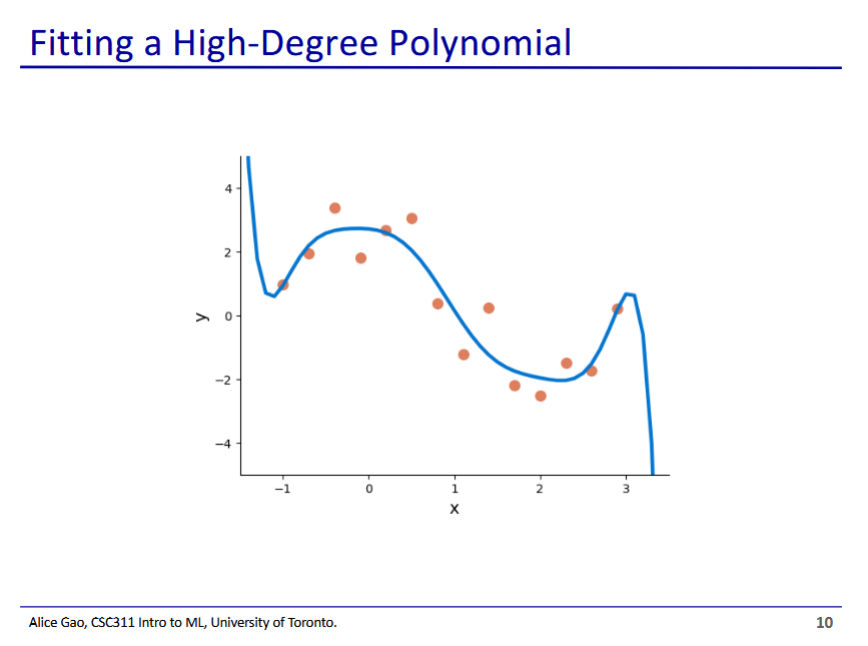
* Points sampled from our true distribution have some natural variance to them due to random noise
* Our machine learning model never sees the curve, but it does get to see these sampled points



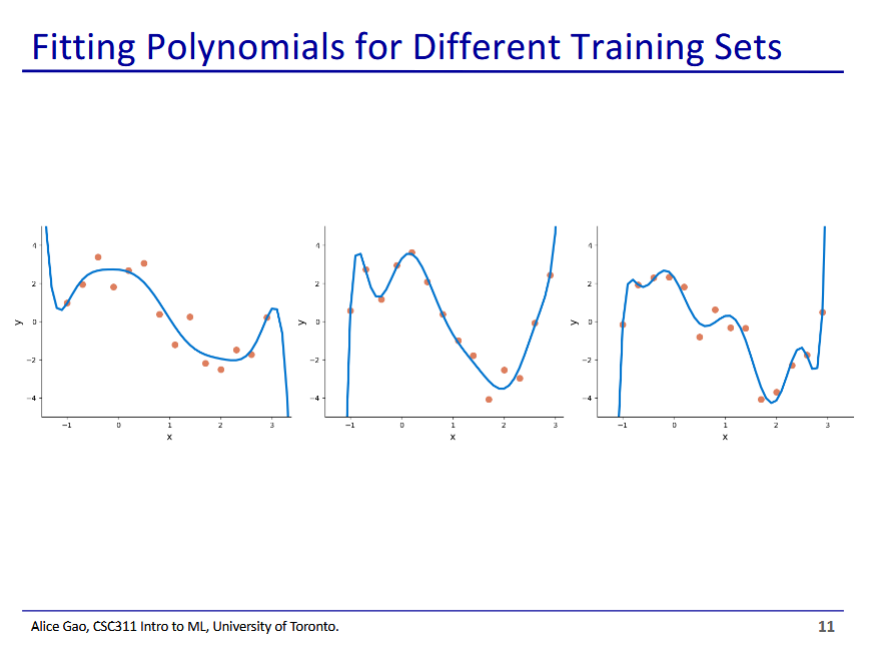
* Because the noise is random, each time we sample a dataset from our distribution we get a different training set
* Thus our training set is one of many possible training sets that could have been generated



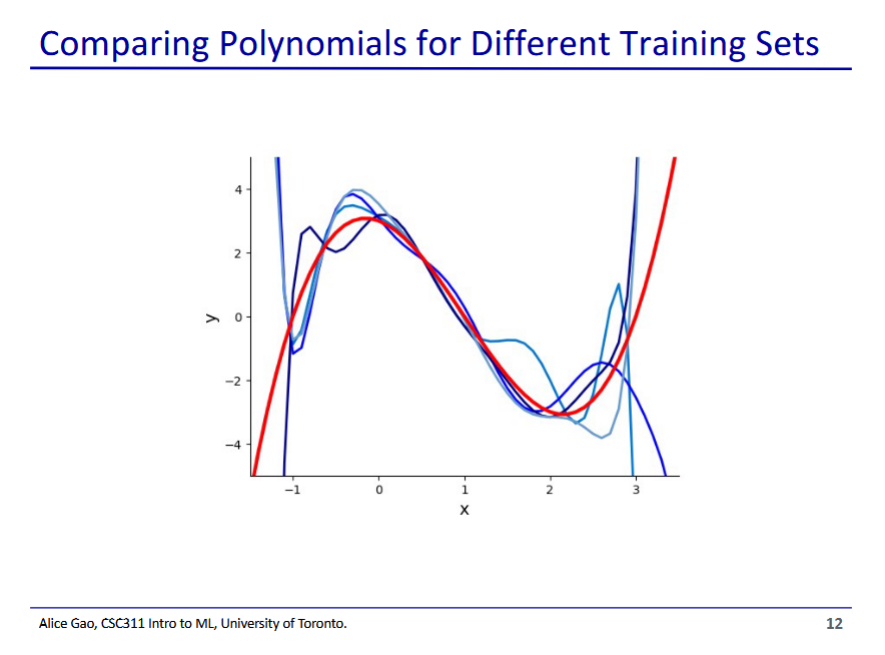
* Variance - for different training sets, we will get models with slightly different predictions
* In our example we had both extremes
  + A model that is a high-degree polynomial (causing overfitting)
  + A model that is a line (causing underfitting)



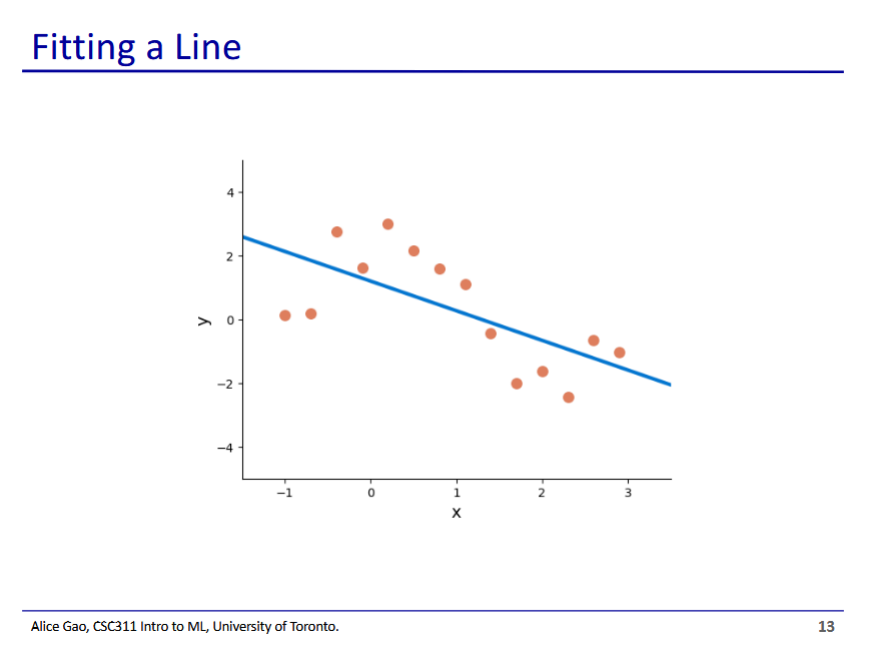
* A high degree polynomial does a good job at matching the training points



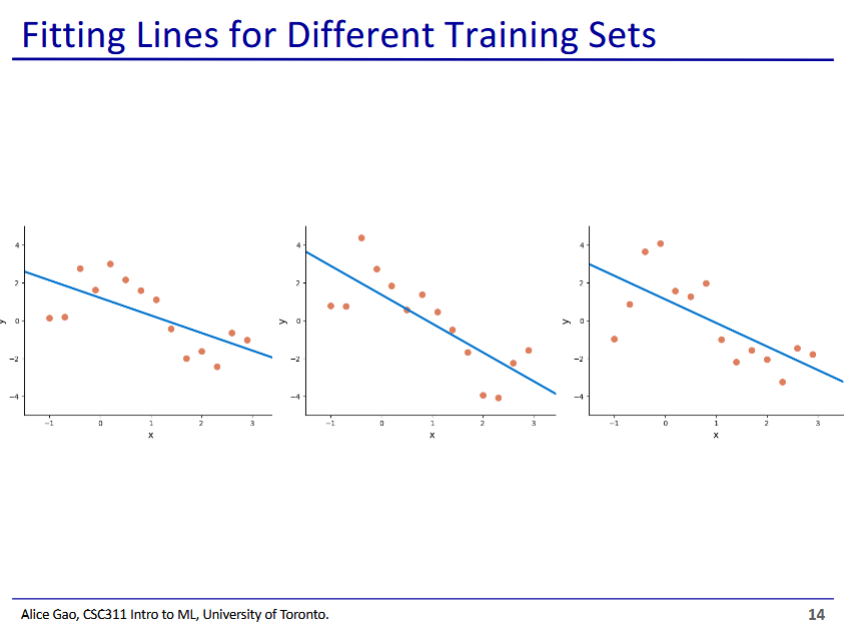
* These are 3 training sets generated from the same distribution
* The high-degree polynomial model looks different depending on the training set



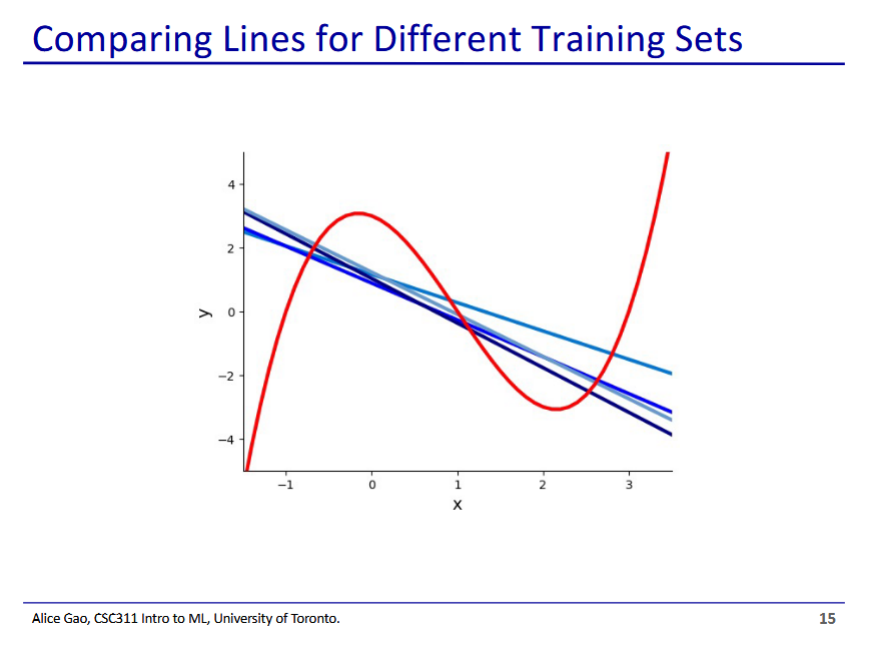
* Our red curve is the true function, the blue curves are the models generated by different training sets
* They are most notably different at the 2 ends



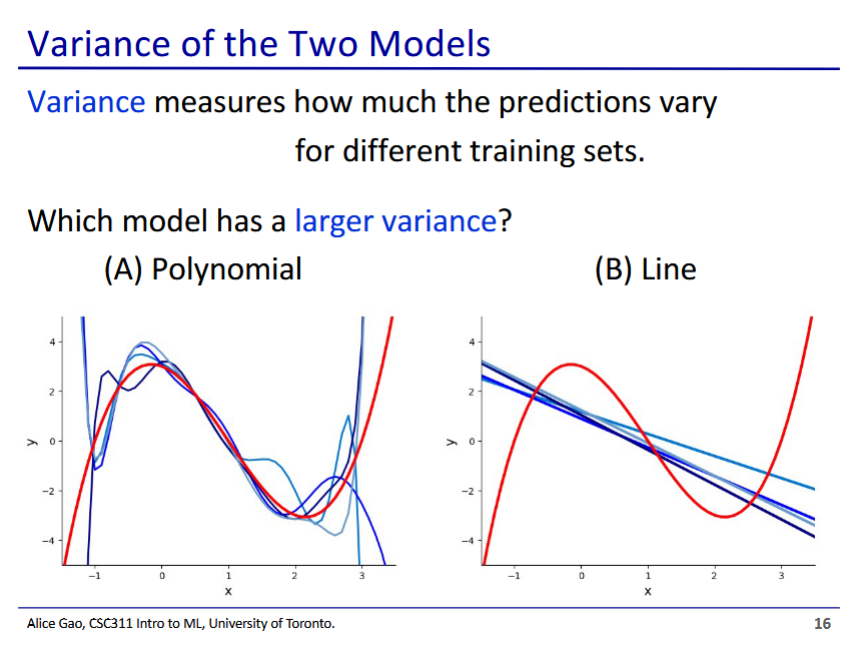
* Now we try a linear model



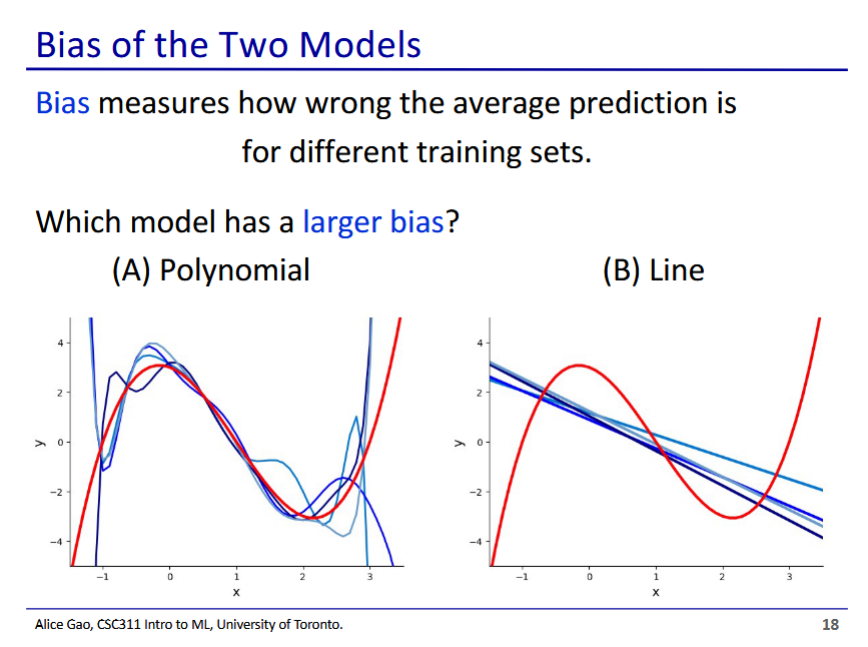
* Fitting a linear model to different training sets generated from our distribution
* Fitted models are slightly different from each other



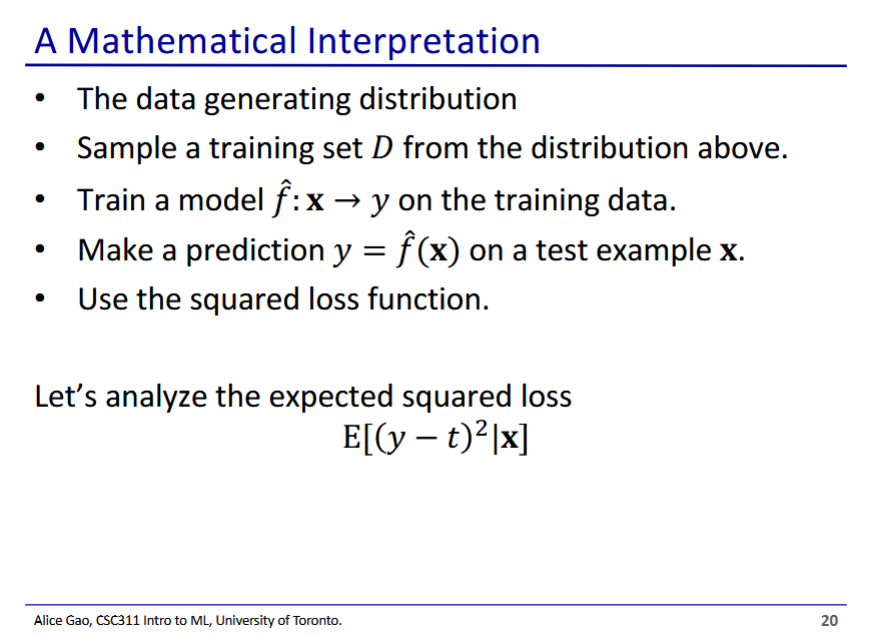
* Fitting a line results in much less variance in the line, as there is less degrees of freedom in how the line can move



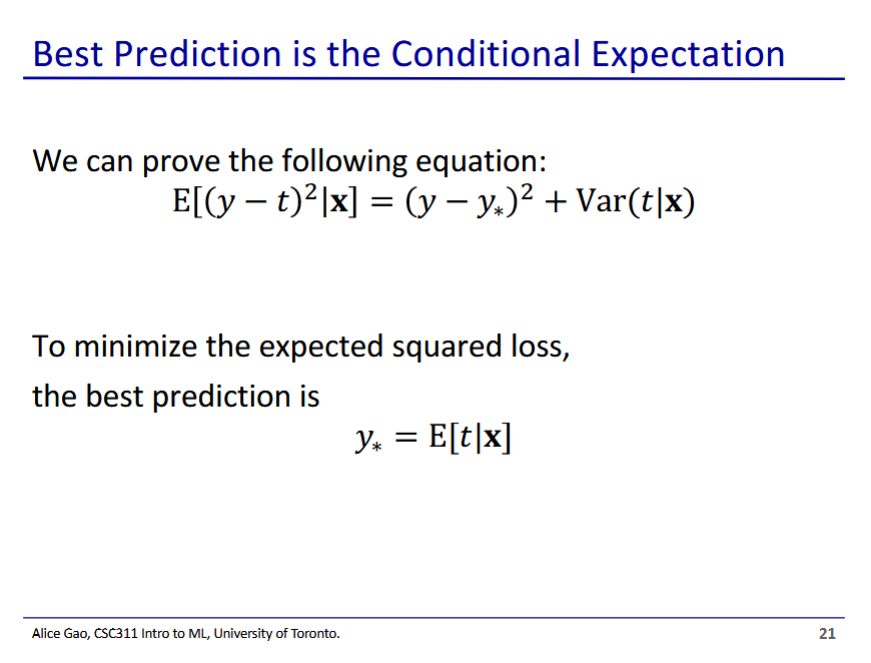
* Polynomial has a larger variance
  + The different polynomial models have a much wider variation than the linear models
* The line does not very much
* The polynomial has more degrees of freedom, there is much more room for it to vary



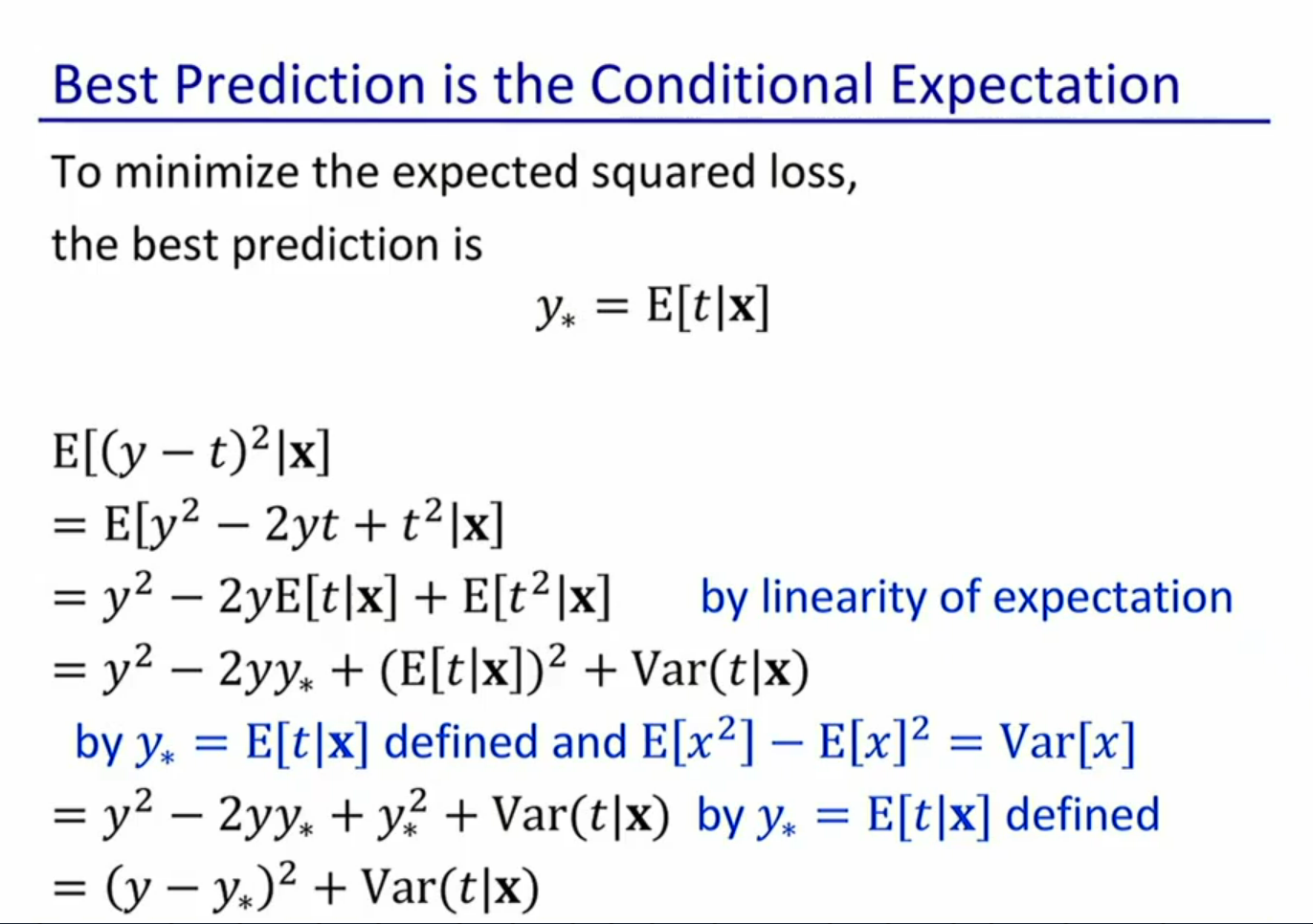
* The line has a larger bias
  + For the polynomial, if we average all our models, it's pretty close to the true function
  + The average of the line models is not very representative of the true function



* This is essentially what we just did, but in words
* We use the squared loss to analyse our model (could use any loss, but squared loss is easiest
  + - The expected squared error given x



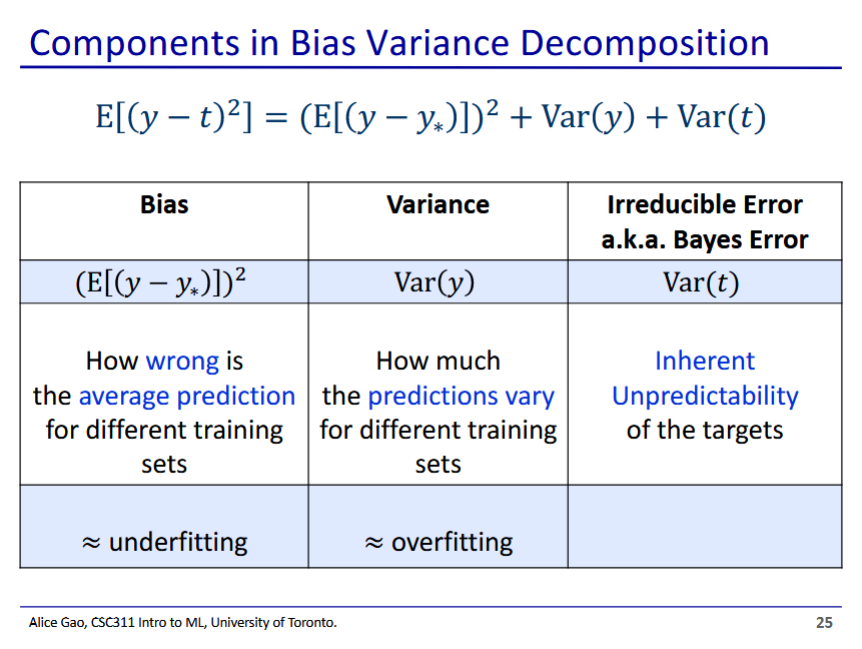
* We can prove that the expected square loss error given x is equal in the equation above
  + Math on next slide
* To minimise our loss, we need to set y to be the same as y\*
  + As y\* is the best that we can do (conditional expectation of t given x), since t randomly varies from the true distribution



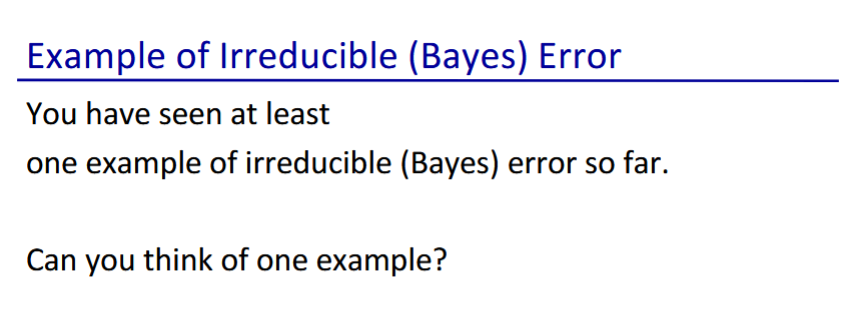
* Take some time to understand this
* Testable, but will not be tested heavily

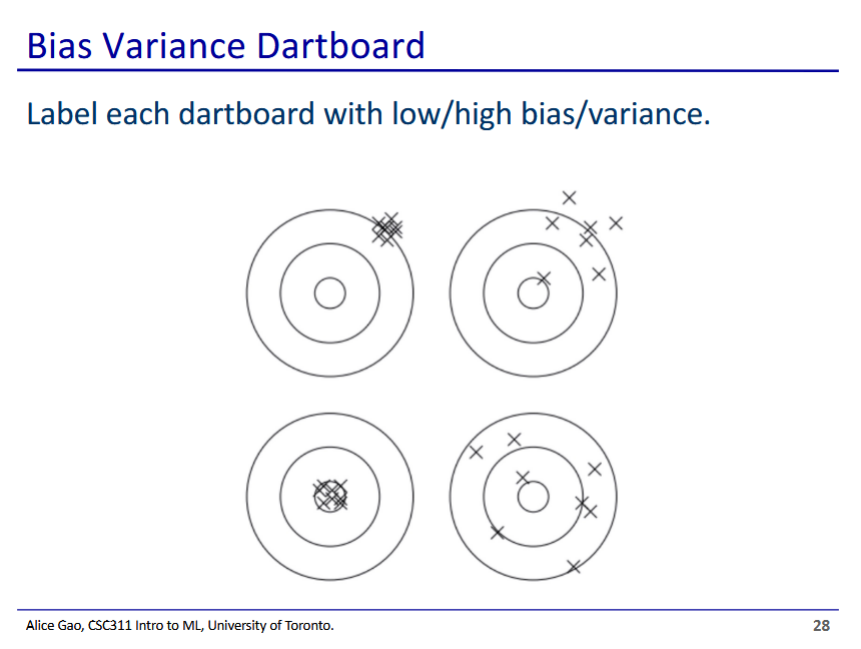


* But y is also a random variable, we don’t have a choice in y since we only choose the model, but the training sets are sampled and variable
  + Thus we can further break down our equation (math not shown)
* Bias - how different on average (expected value) is our prediction (y) from the ideal prediction (y\*)
* Var(y) is variance in y due to our sampling
* Var(t) irreducible error
  + We cannot change this error, this is the variability of the target value

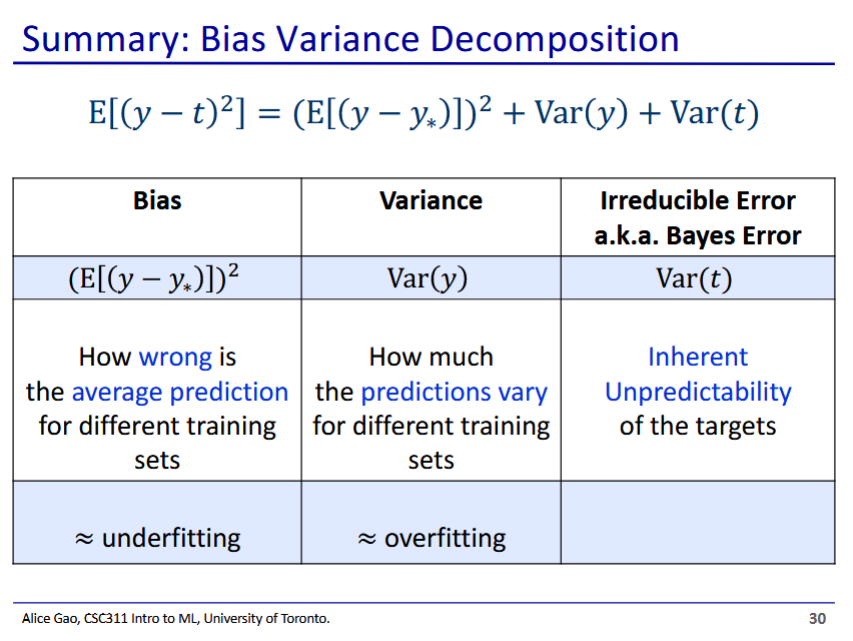


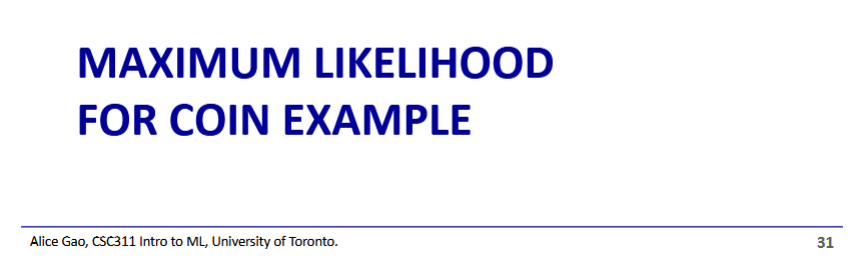
* High bias leads to underfitting
* High variance leads to overfitting
* If bias and variance are both high, we both overfit and underfit
  + Yes they can happen at once, but in practice we tend to only have one of them high or none of them high

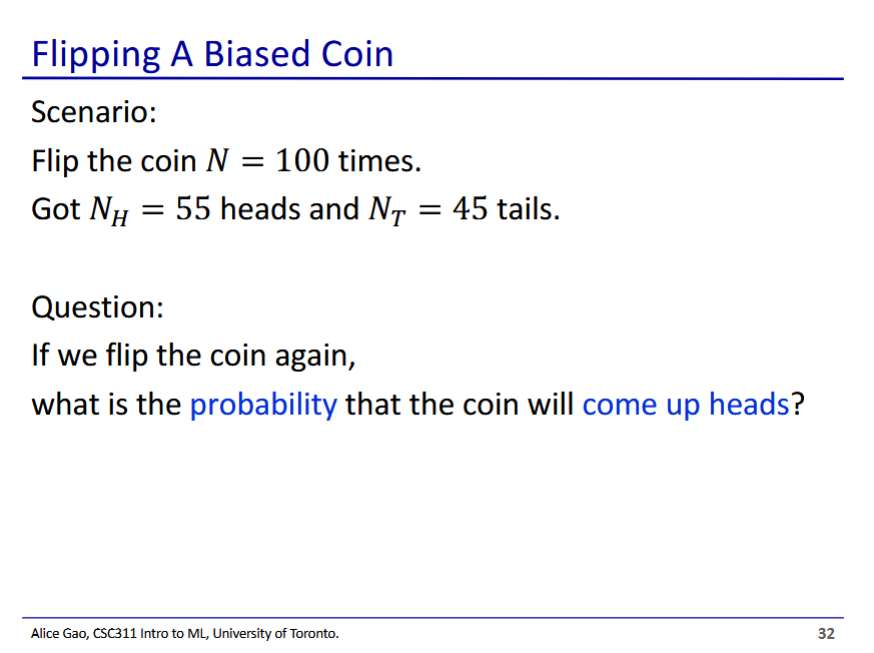


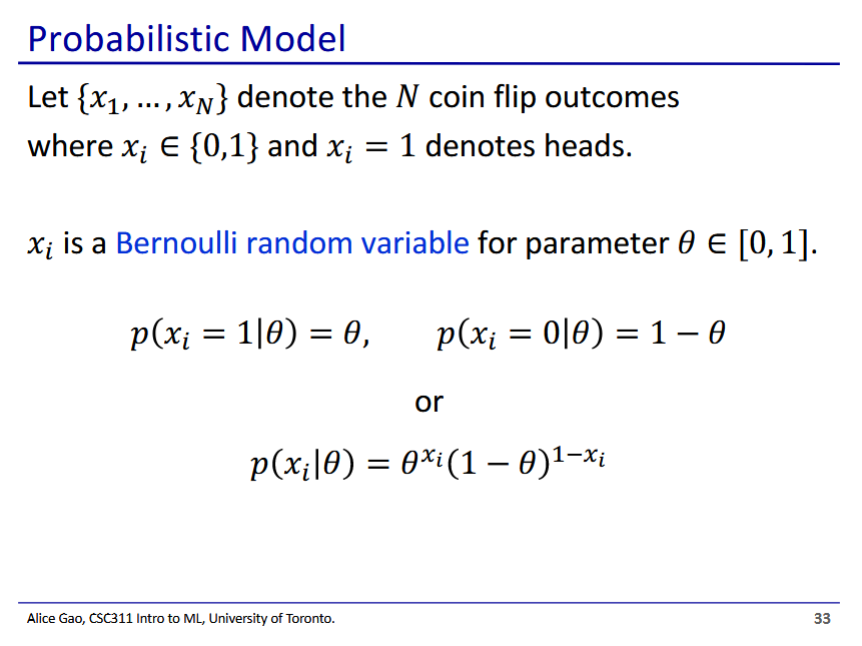


* Top left is high bias, low variance
* Top right is high bias, high variance
* Bottom left is low bias, low variance
* Bottom right is low bias, high variance









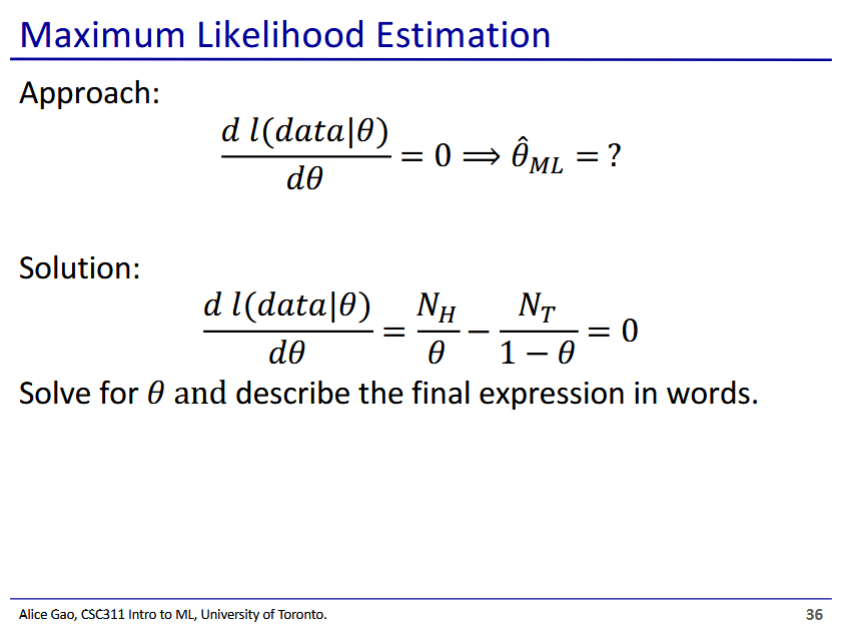
* Combined equation
  + When xi = 1, it becomes the left equation
  + When xi = 0, it becomes the right equation



* Likelihood is the probability that we see our training data given a certain probability
  + Is a function of and data
* Taking the natural log allows us to take the summation more efficiently

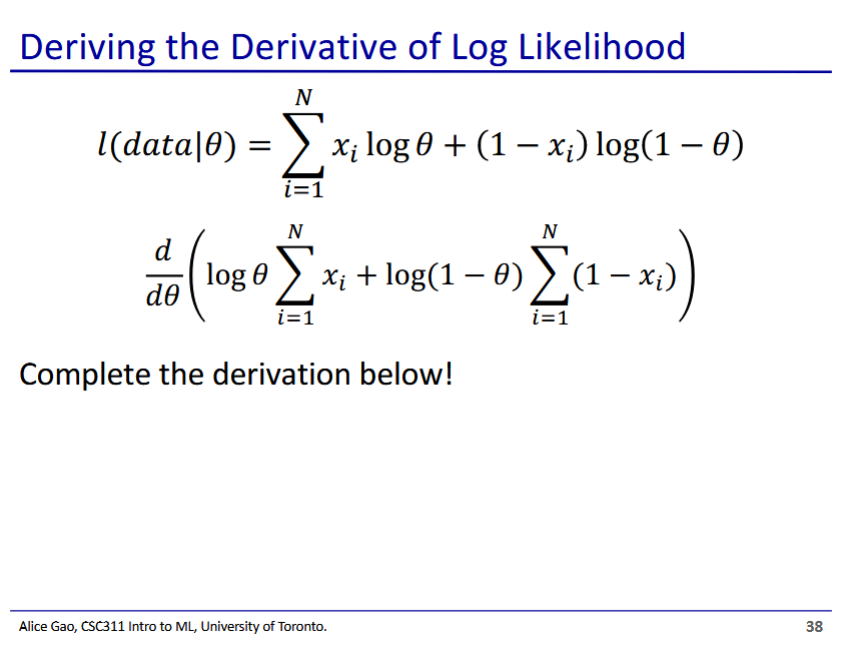


* We then pick the theta that maximise the log likelihood



* To get the optimised theta, we take the derivative with respect to theta and set it to 0

  + Solve for theta



(first sum counts only heads, second sum counts only tails)

