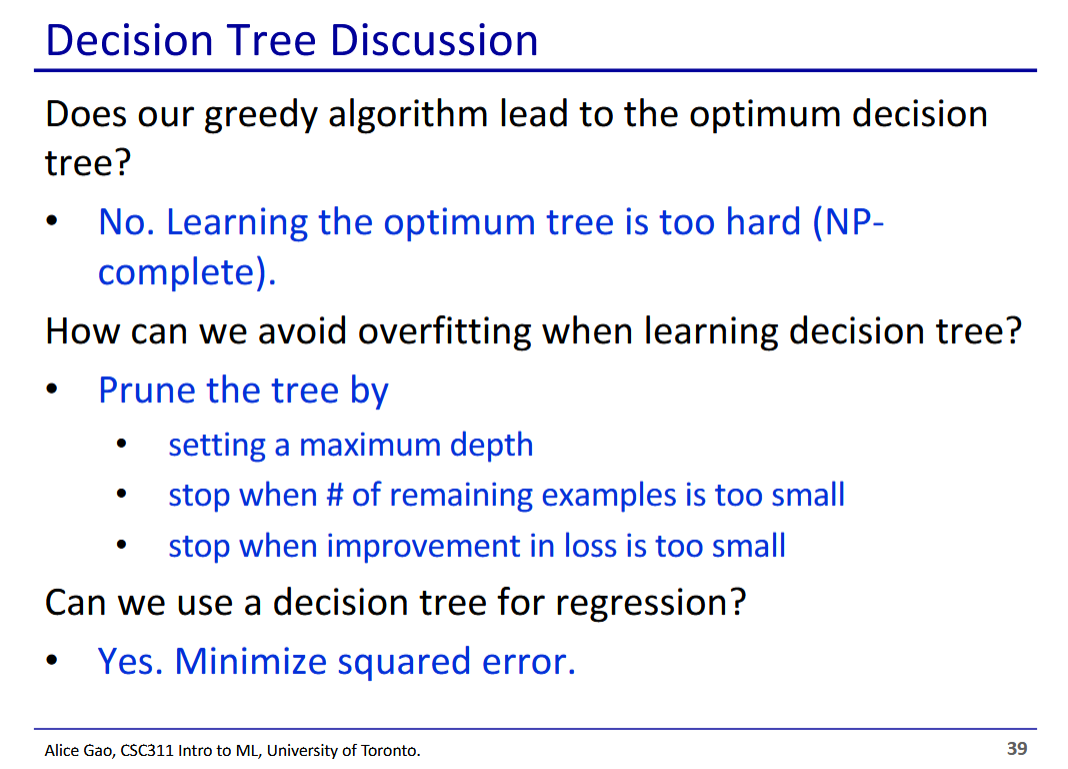
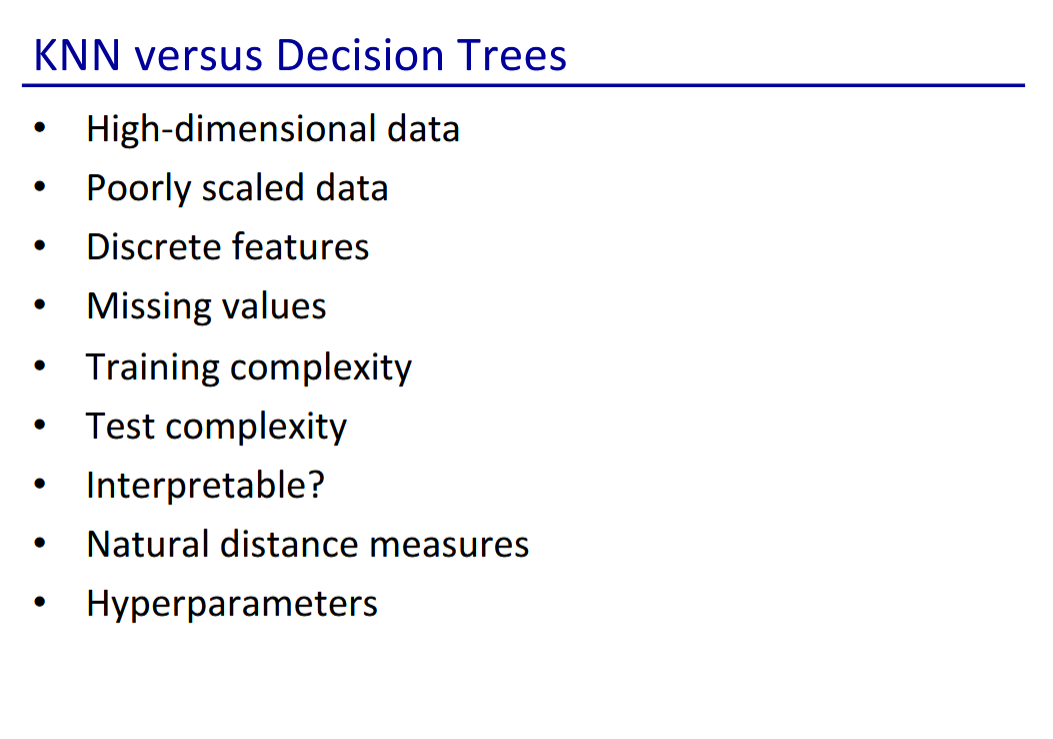
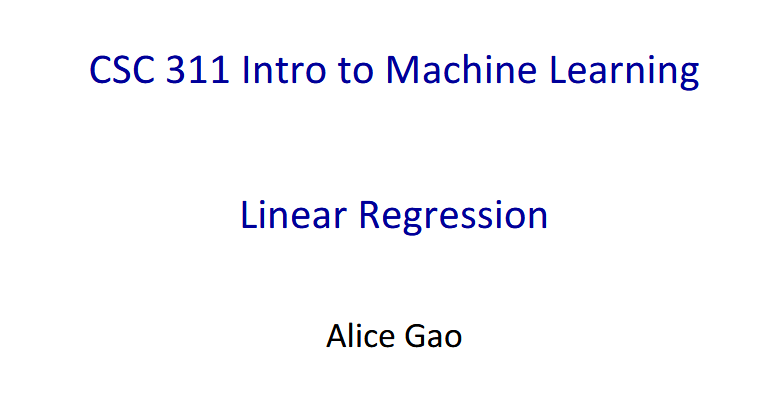
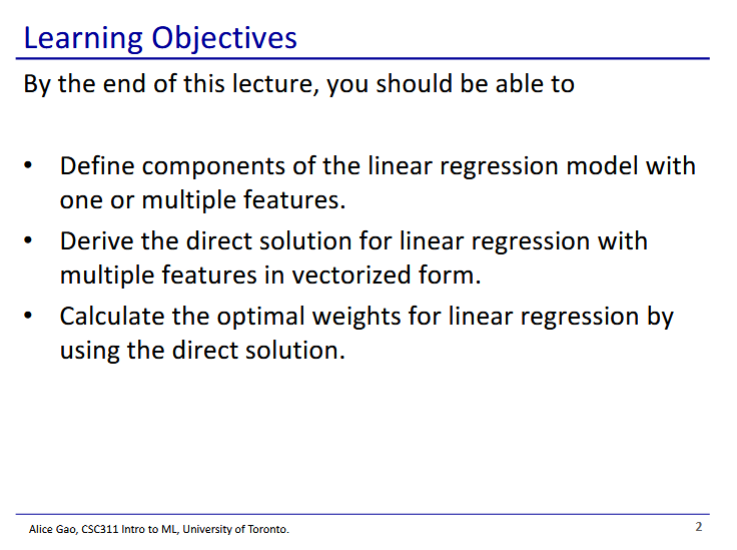
| **Admin stuff**   * Review videos and practice quizzes from UTM   + Good if you need extra help with base concepts   + **Derivatives and chain rule are very helpful for this topic** * Survey   + Due tomorrow   + Used to improve teaching of this course this session and future sessions * Midterm practice problems   + We will get a practice problem for linear regression   + A1 is practice problems for the test   **Wrapping up decision tree**   * Greedy algorithm does not find the optimum tree (NP-complete problem, takes too long)   + Instead it looks for an adequate tree * How to prevent overfitting   + We limit the size of the tree (set max depth, set lower limit for # of examples needed to make a split, stop when improvement is too small) * Decision trees can be used for regression, by minimising square error * Exercise on comparing KNN to decision trees on page 2   **Model vs hypothesis in regression**   * Hypothesis - any function that we can learn * Model - a set of assumption that limits what hypotheses we can learn   + No free lunch theorem - in order to learn anything, we need to set up some assumptions   **Linear regression**   * **Regression** - Learn a function that approximates our training data points   + **Linear regression** - this function needs to be linear * **Linear regression with one feature**   + Input is a scalar value x   + Hypothesis in the form of     - “w” is the weight and “b” is the bias   + **Loss function ()**     - Calculates the square distance between the predicted and actual result       * ½ is to make taking the derivative easier   + **Cost function ()**     - Average loss of hypothesis over entire training set   + **The best hypothesis is when cost is minimised**     - This occurs when the derivatives of with respect to w and b are 0     - Taking the derivatives will give us 2 linear equations to solve to get the best w and b       * **Math on slides 21-27, make sure you can do**       * Chain rule is very helpful, review if needed * **Linear regression with many features**   + Now the input is a vector with D features   + Hypothesis: (when vectorised)     - Our weights (w) is now a vector of size D representing weights applied to each feature     - We can apply weights to features using the dot product ()   + **Simplifying model**     - We can choose to add a dummy feature of 1 to the input and add an extra weight     - This incorporates the bias into the weights and simplifies calculations     - Simplified hypothesis:   + **Cost function**     - (vectorised)       * Also written as   + **Finding the optimal hypothesis**     - We first find the **gradient** of the cost function and solve for when it is 0       * Gradient is the derivative of with respect to the different weights     - Calculating derivative with respect to weight       * We first calculate gradient for loss function, then add together to get gradient for cost         + Math on slides 40-42 |
| --- |

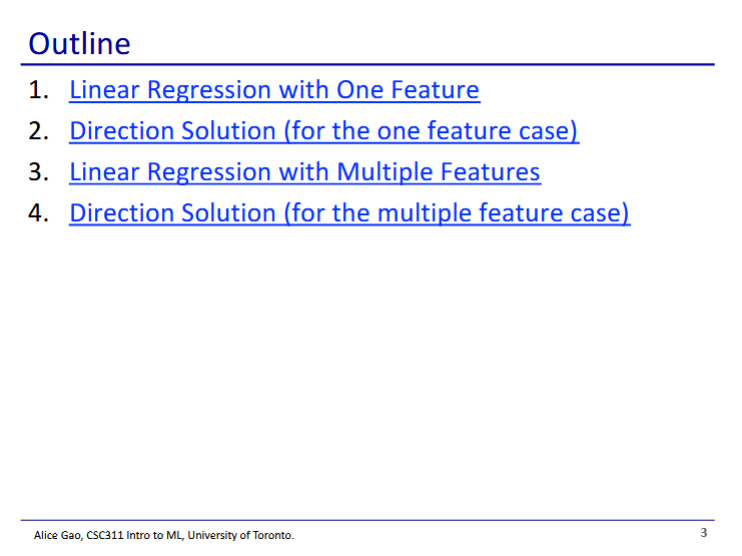




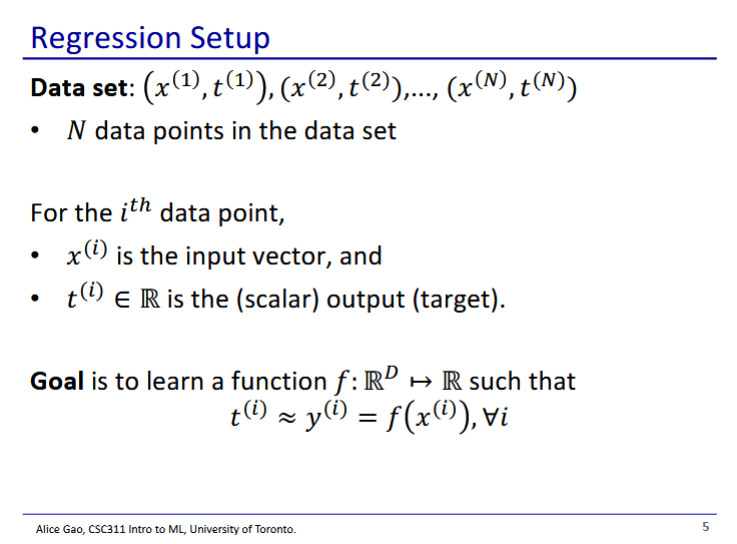
* These factors are where KNN and decision trees differ
* No answers on this slide, take some time to think about which would be better for each of these cases
  + Can ask on Piazza too



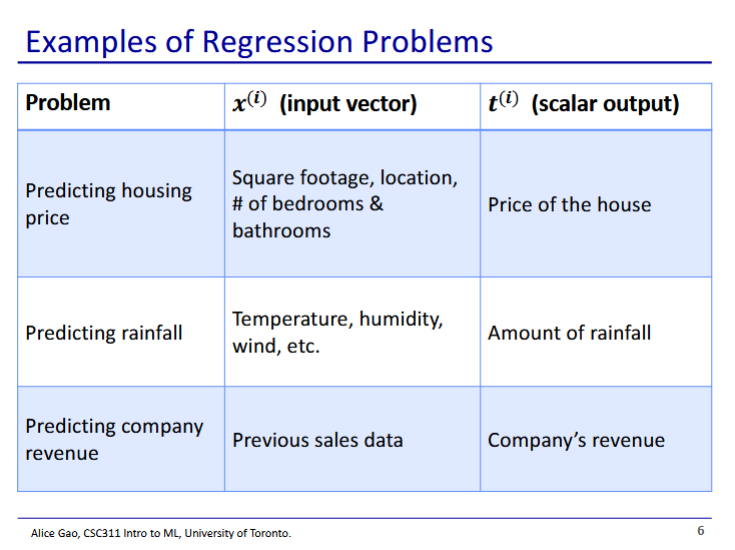


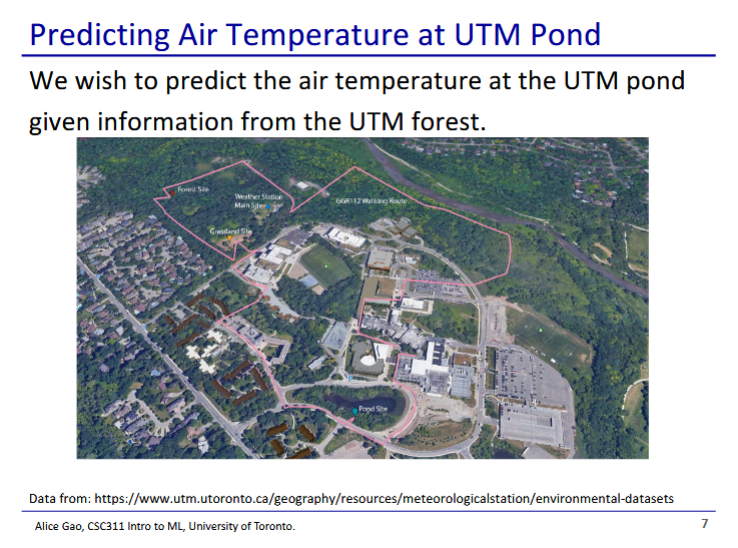


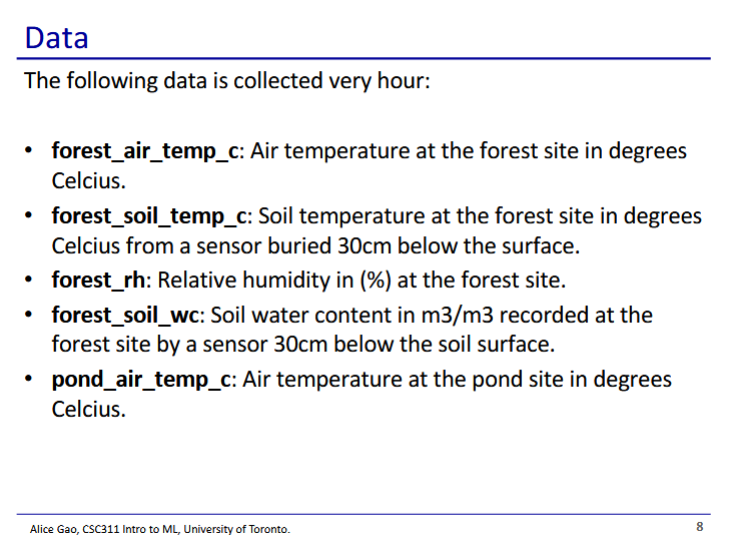




* We want to learn a function that approximates all of our data points

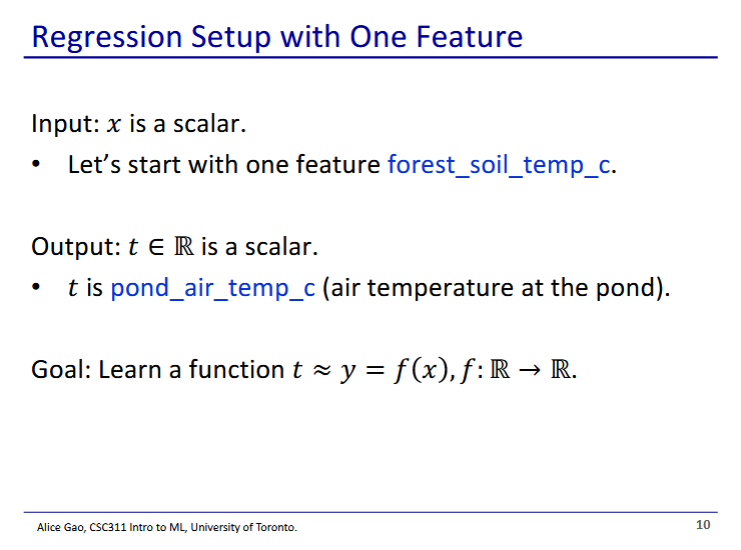




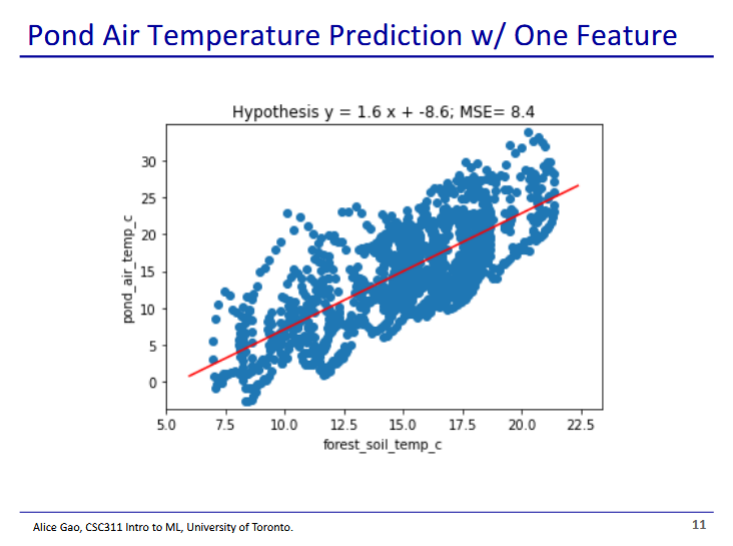


* We want to predict pond\_air\_temp\_c using the other variables

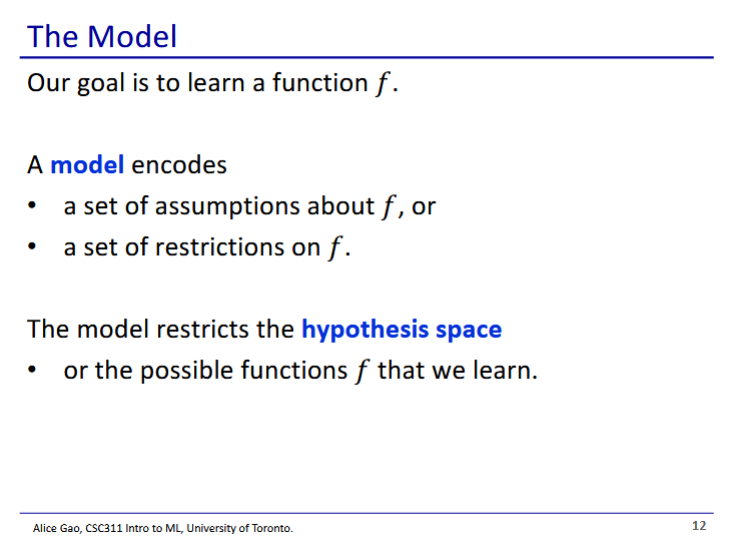




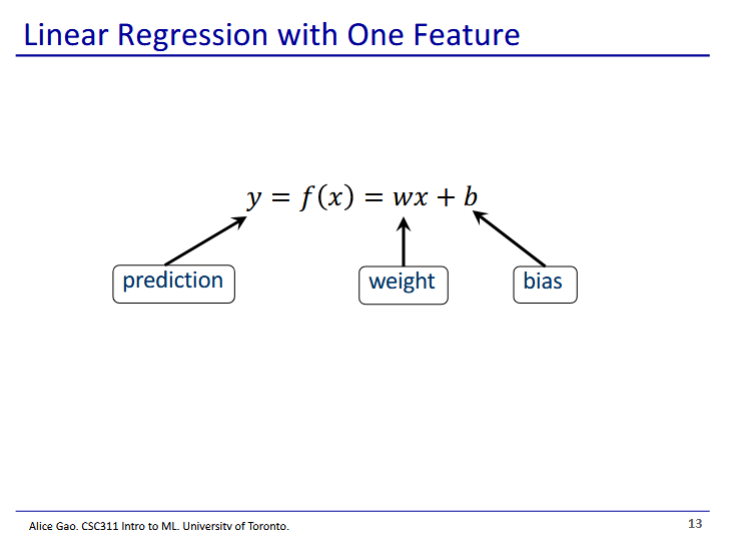
* One feature - our input is a scalar



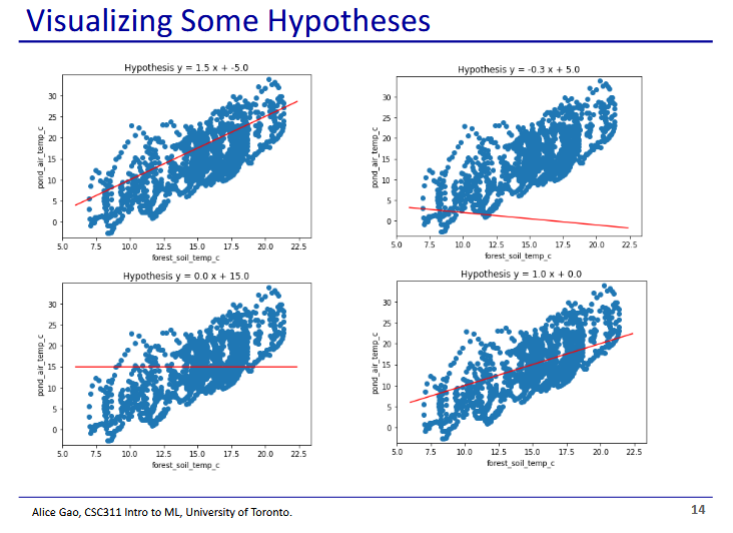
* Linear regression finds the best fit line in our graph



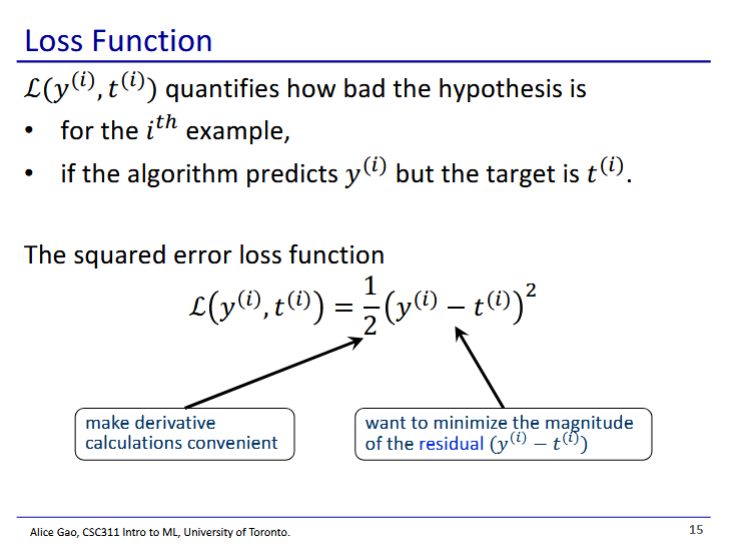
* Differentiating between model and hypothesis
  + Hypothesis - one specific function we can learn
  + Model - a set of functions
* When we choose the model, we put some restrictions on what hypotheses we can have
  + A linear regression model assumes that the function is linear
  + A decision tree model assumes the function can be expressed as a decision tree
* No free lunch principle - in order to learn anything, we must make assumptions



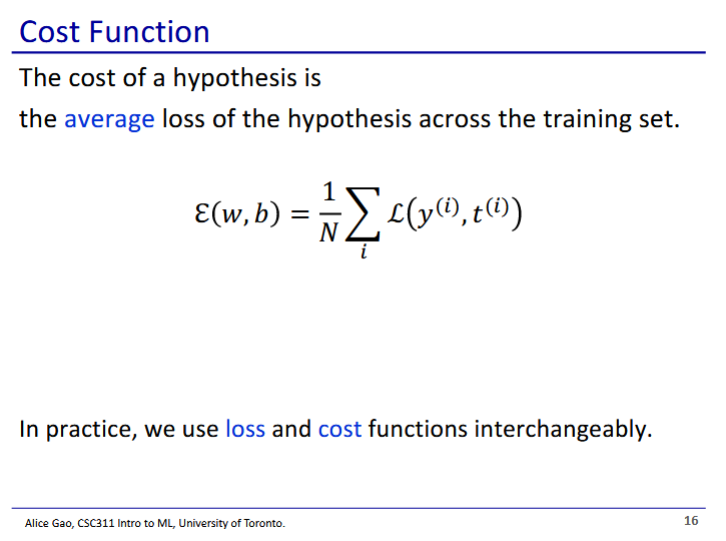
* This is a linear function:
  + Multiply the input by the weight, and add the bias to get the output
* A linear regression model restricts hypotheses to be linear functions



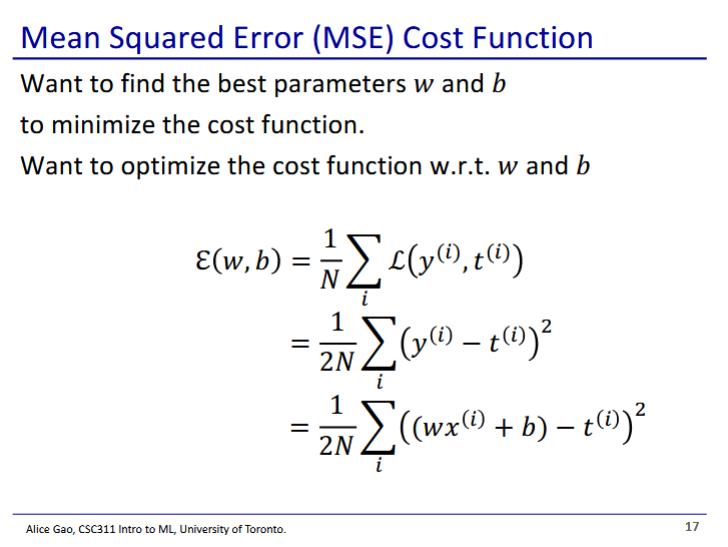
* All of these functions are hypotheses
  + Which one is the best?
* We need to define which hypothesis is best for the computer to pick one
  + We will use a loss function



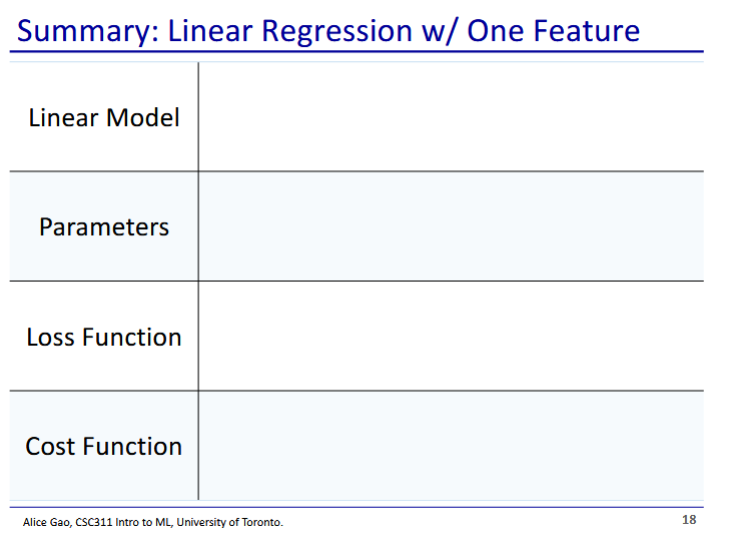
* Loss function evaluates how bad a hypothesis is
  + We want to minimise loss
* Residual - difference between the target and prediction
  + We want to minimise the magnitude
* Dividing everything by 2 makes derivative easier
  + Taking the derivative brings down the 2nd power, which cancels out if we divide by 2



* The cost function - calculates the total loss over all points in the training set
* People often refer to loss and cost interchangeably
  + But when they refer to minimising loss, it actually refers to minimising cost

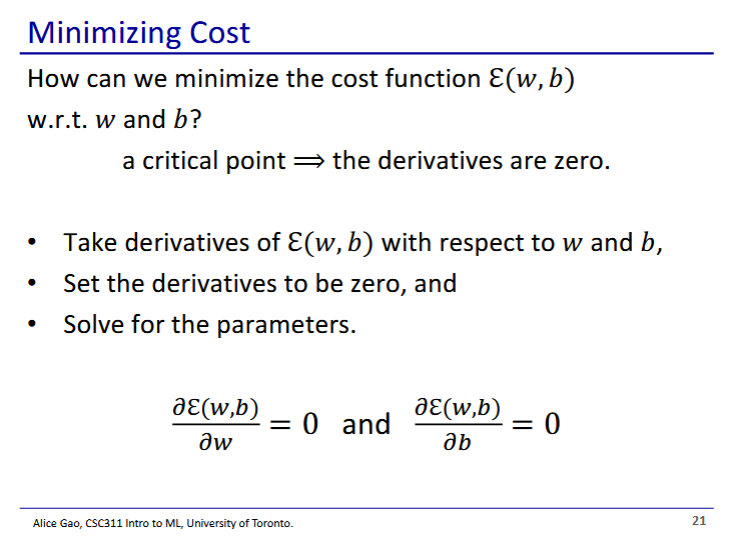


* Cost function is a function of 2 parameters w and b
* We optimise the cost function by choosing the best values for w and b

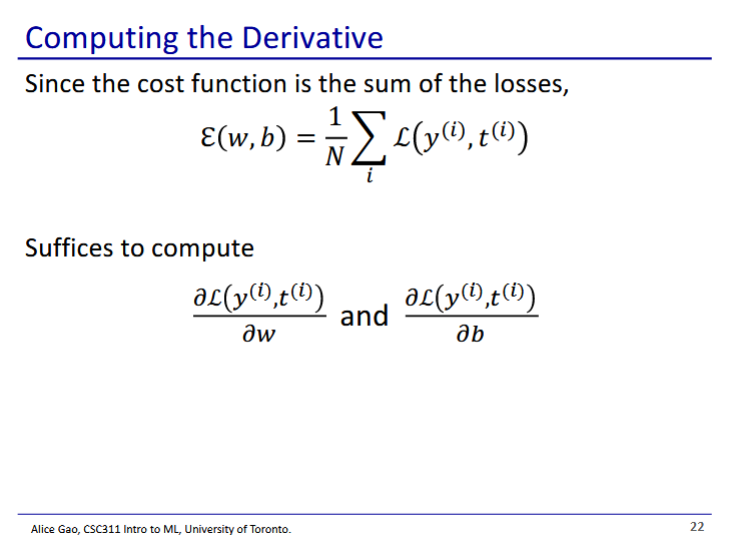


* Linear model:
* Parameters: w, b
* Loss function:
* Cost function:

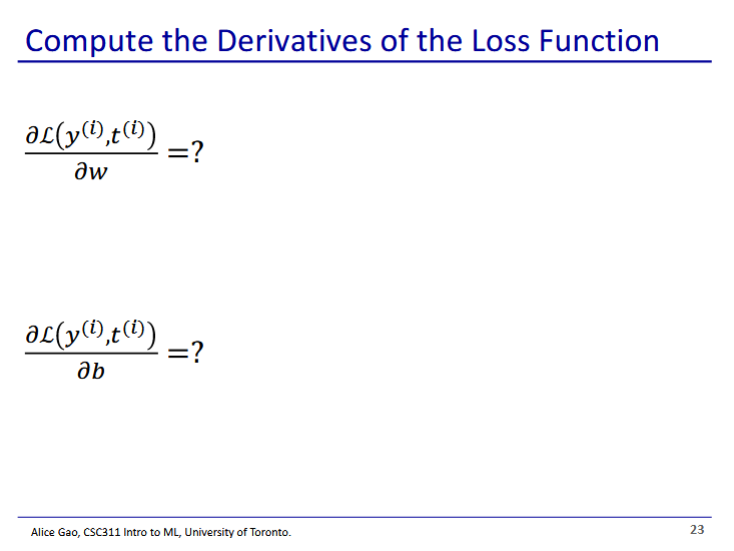




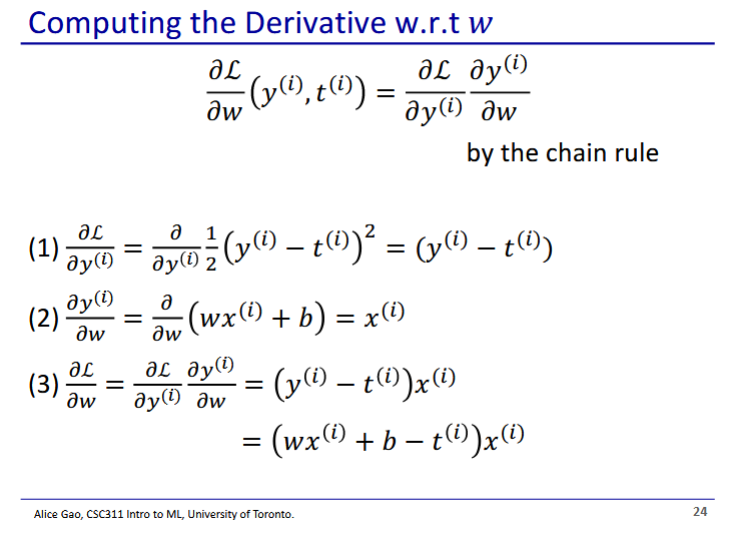
* To minimise the cost function, we need to find the critical points
  + maxima/minima have zero derivative
  + Thus we take the derivative of w and b

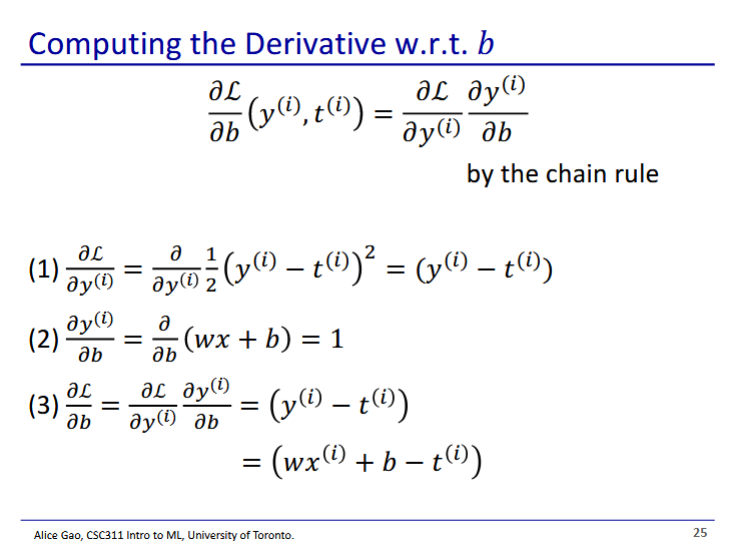


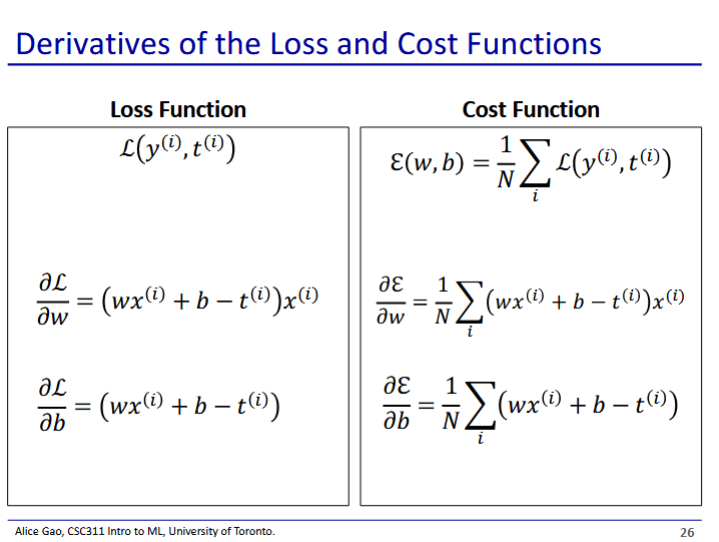
* We disregard the summation in the cost function for now, we just focus on the loss function



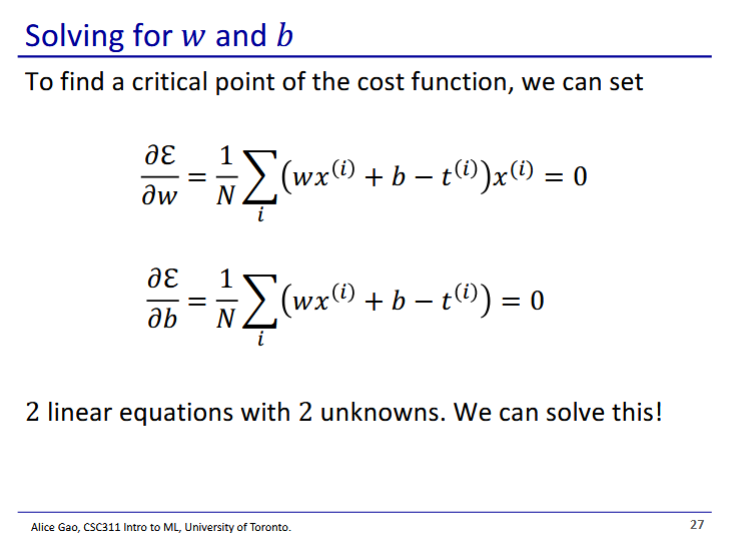
* Try this part on your own first
  + Chain rule will be useful
* Answers on next slides



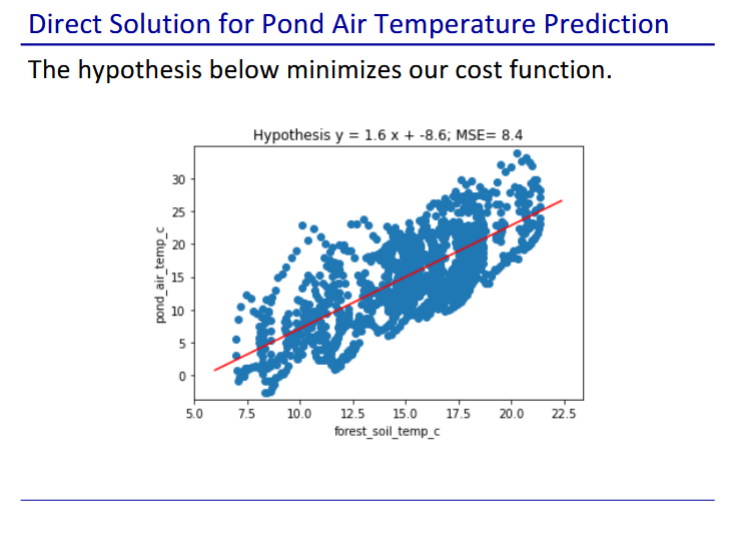




* Derivation of cost function is sum of derived loss function

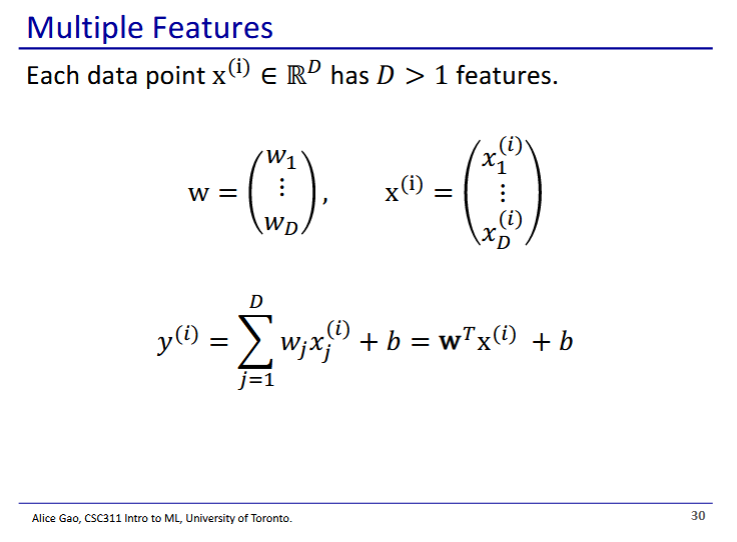


* Critical point is when the derivatives are 0
* We end up with 2 linear equations with 2 unknowns
  + Solve to get optimal w and b

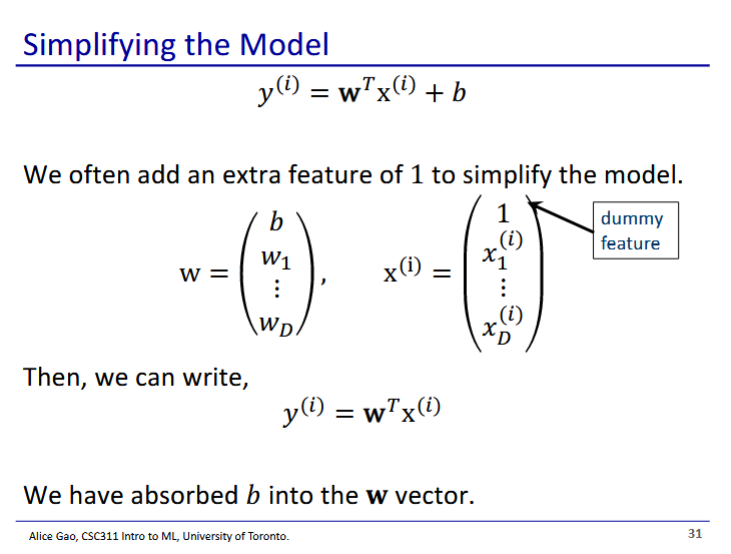


* Using cost function, we can get a good hypothesis

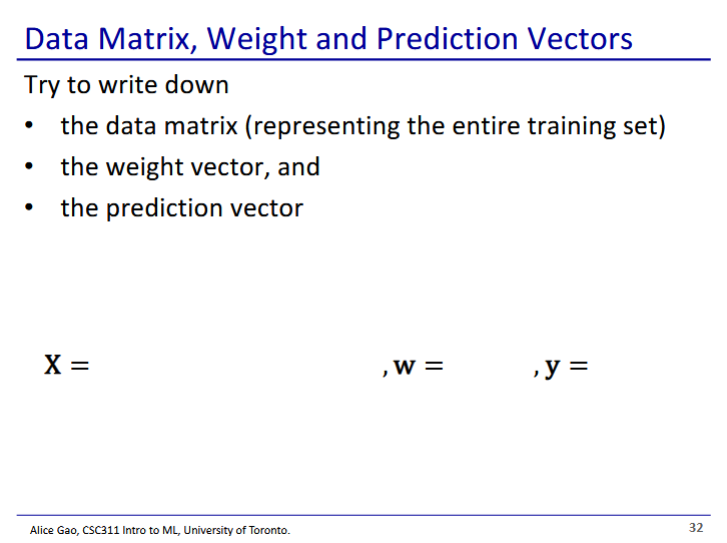




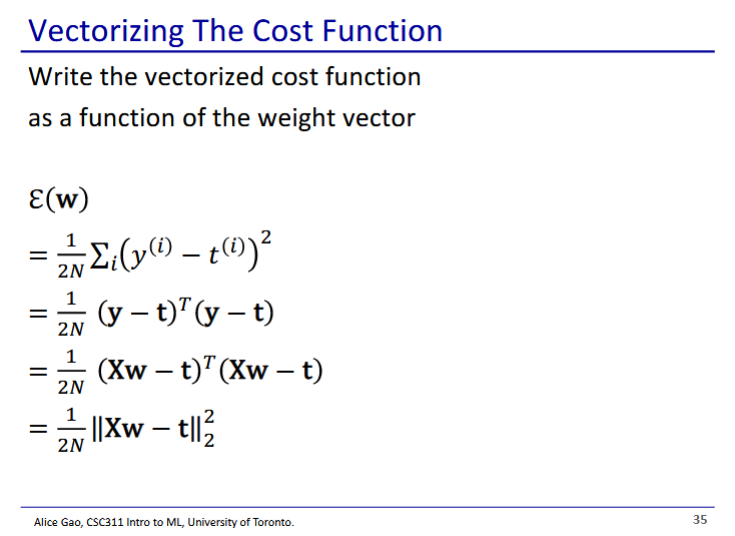
* We now have a set of weights
  + One weight for each feature of the input
* Unvectorised
* Vectorised
  + This is the dot product, will apply weights to each feature



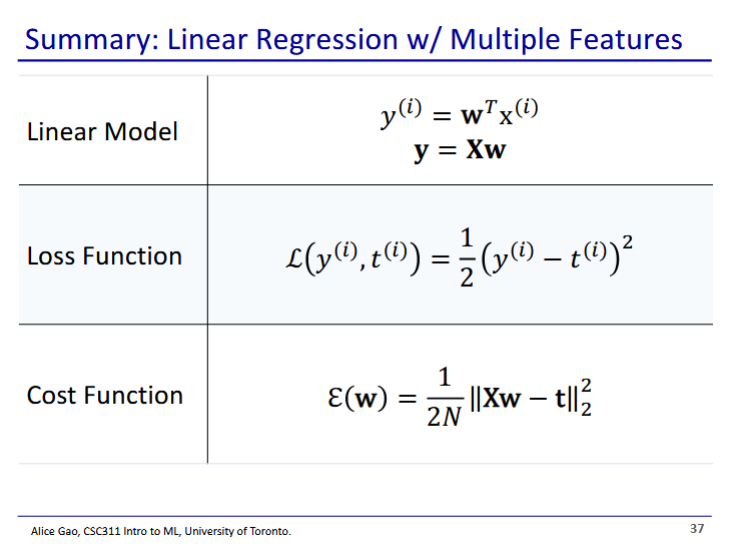
* We often have a dummy feature to simplify our model
* This allows us to include the bias as a weight
  + Thus we only have weights now, no biases



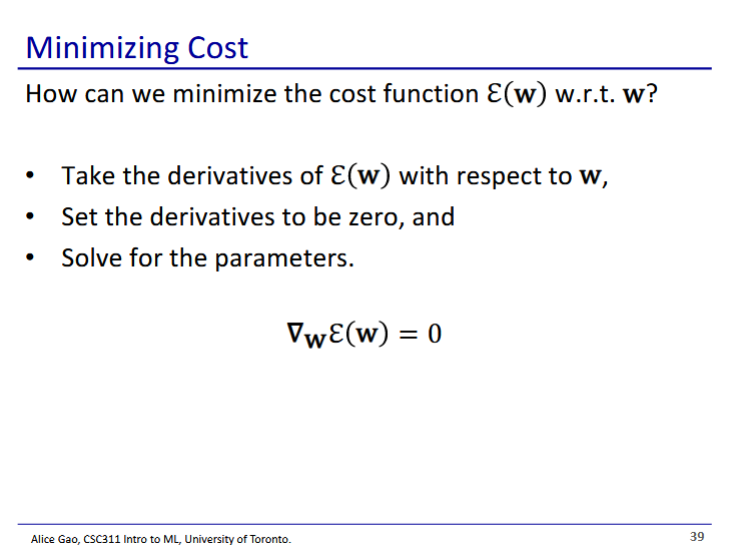
* X - data matrix
  + Dimensions of first row should match dimensions of weight vector
    - These are the features of
  + Each row represents 1 input
* W - weight vector
  + Column vector of each weight
* Y - prediction vector
  + Column vector with each feature of after multiplying by weight
* Since the dimensions of X and w match, we can take the dot product to get y



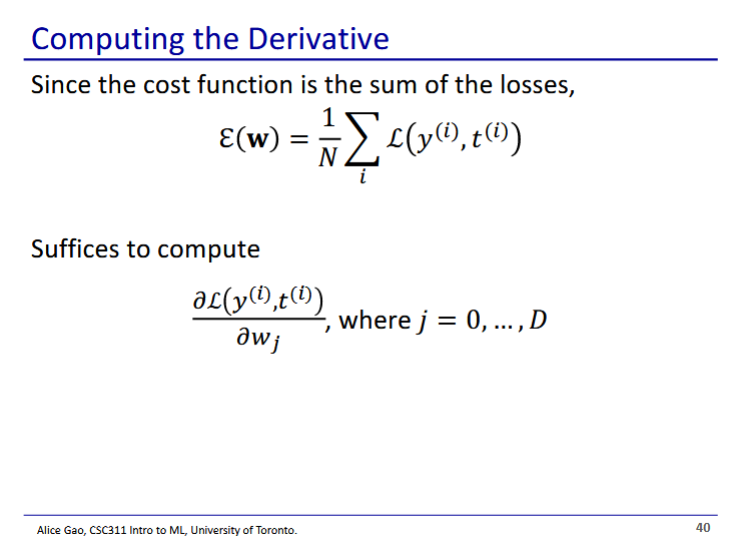
* Line 3 is line 2 but written using dot product
* Line 4 is line 3 with y expanded
* Line 5 is a fancier way to write line 4
  + || is euclidean norm

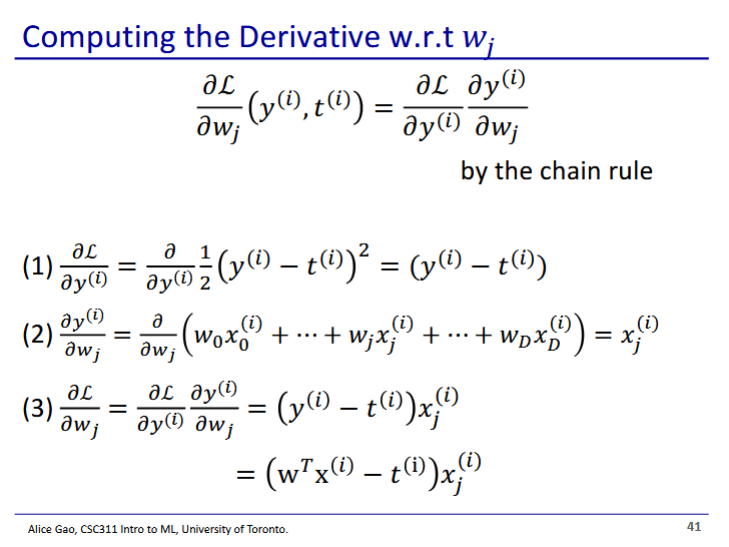




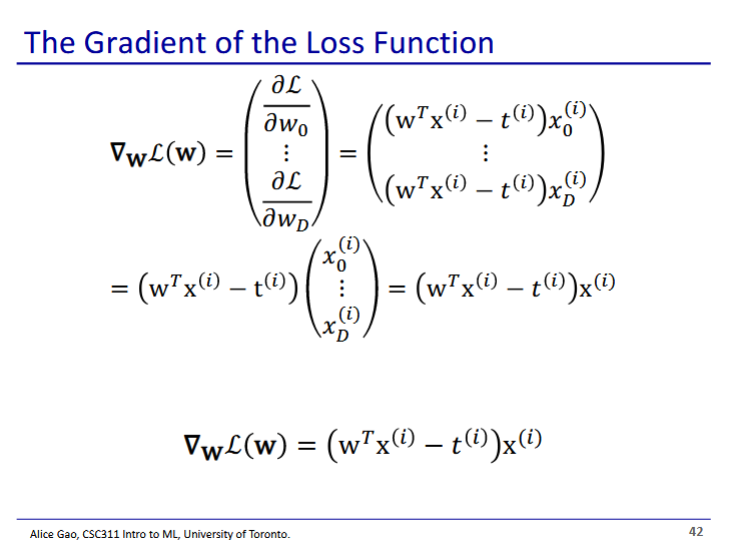


* Gradient - derivative of the loss function for various weights





* We can apply this formula for any weight
* This is one weight, we want to calculate for all weights
  + Calculating for all weights will give us a vector



* Note: the only difference between each element of the gradient vector is the last part
  + Thus we can factor out the first part out
* This gives us the equation for our gradient