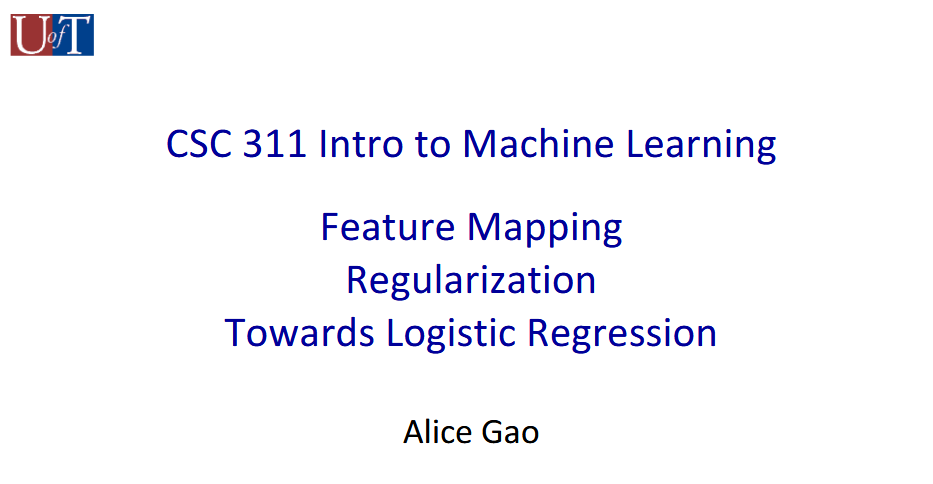
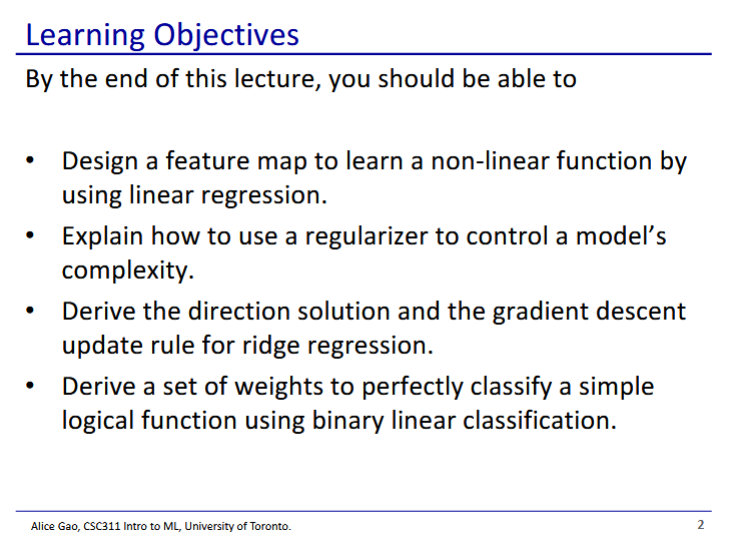
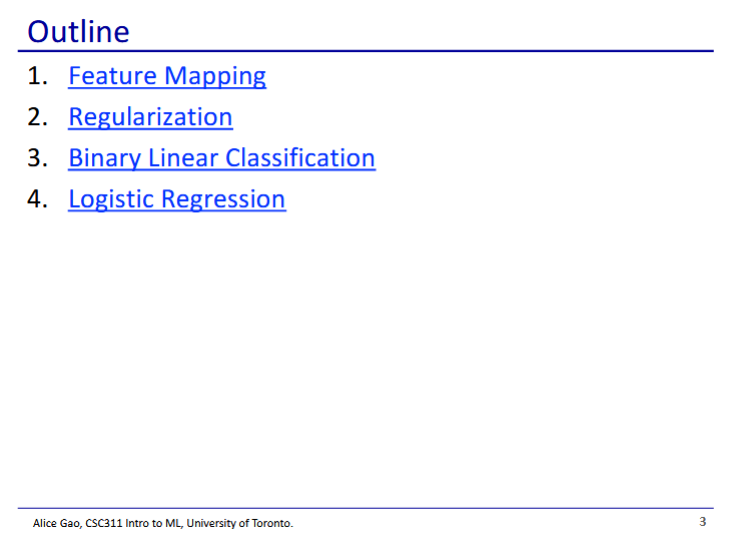
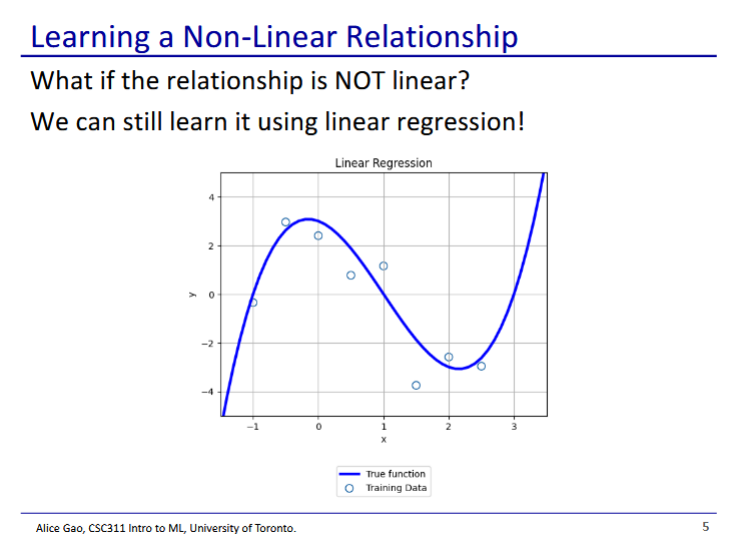
|  |
| --- |
| **Test info:**   * See info post on Piazza * Only covers KNN, decision trees, 1 question on linear regression   + Only theoretical questions, no writing code * Aids:   + Non-programmable calculator   + Aid sheet     - 1-sided letter paper, typed or written   **Feature mapping/basis expansion**   * Method for approximating non-linear relationships using the linear regression model * We use a function to map features into a different space   + The hypothesis is not linear in x, but it is linear in * **Polynomial feature mapping**   + maps a scalar feature to a vector representing the terms of a polynomial of degree M        * + Weights are then applied to features of the new vector to fit it to the data   + **Effects of hyperparameter M**     - Too low M and the model underfits     - Too high M and the model overfits       * As M increases the magnitude of the weights also increases     - We can control overfitting by tuning M using a validation set, however we can also use a **regularizer** to keep weights low   **Regularisation**   * Method for limiting overfitting by preferring certain hypotheses over others * Typically we will use regularizers to penalise having large weights   + Having large weights is a sign of overfitting as model fine-tunes to the data   + Models are biassed to features with very large weights     - We want the model to consider many features * **regularisation**   + We sum the squares of the weights to get a value we can add to the cost function   + **Regularised cost function**     - * (hyperparameter) controls how strong the regularizer is     - Causes the cost function to compete with the regularizer       * Results in a tradeoff between a simple function and a function with minimised loss     - When is too large, the regularisation term is too strong       * We have very small weights and the model underfits     - When is too small, the regularisation term is too weak       * We have very large weights and the model overfits   **Modular approach to ML**   * **Model** - describes relationship between variables * **Loss function** - quantifies hypothesis fit over data * **Regularizer** - expresses preferences over different hypotheses * **Optimisation algorithm** - fit a model that minimises loss and satisfies regularizer preferences   **Binary Linear Classification**  Method for using lin reg model to make bin class decisions  , Model returns 1 if z exceeds a threshold (r)    * **Simplification**   + We can eliminate the threshold r by merging it into the bias and making the new threshold 0   + We can then merge into the weights by adding a dummy feature to x and making the weight for |



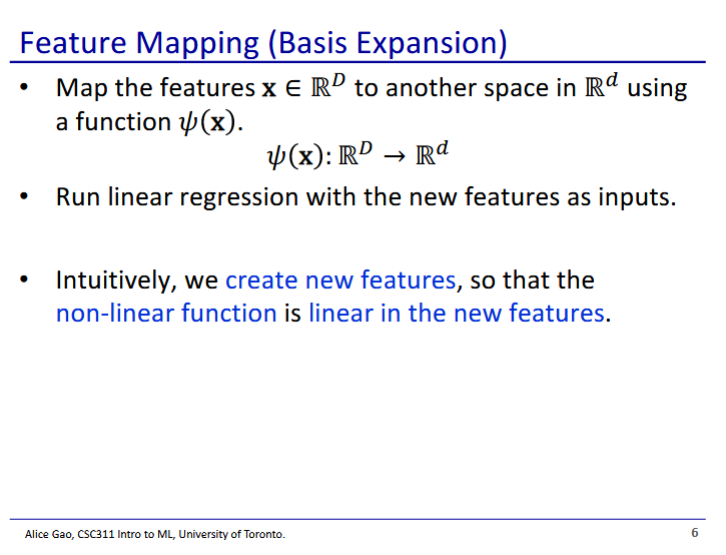




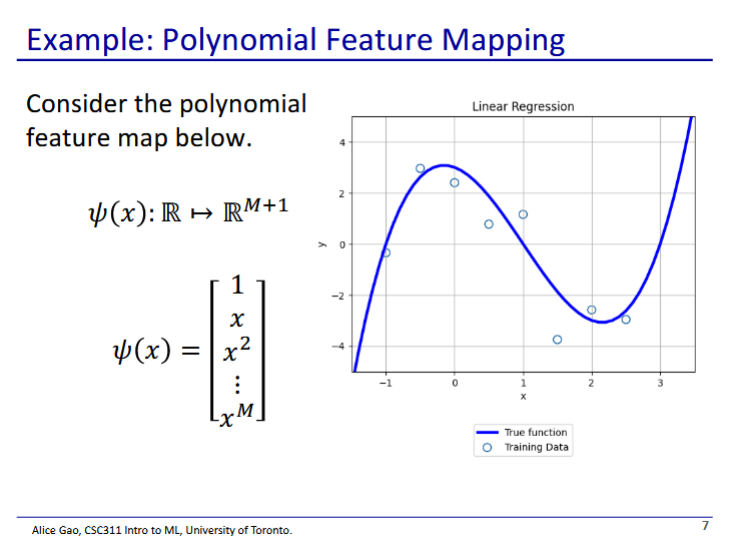




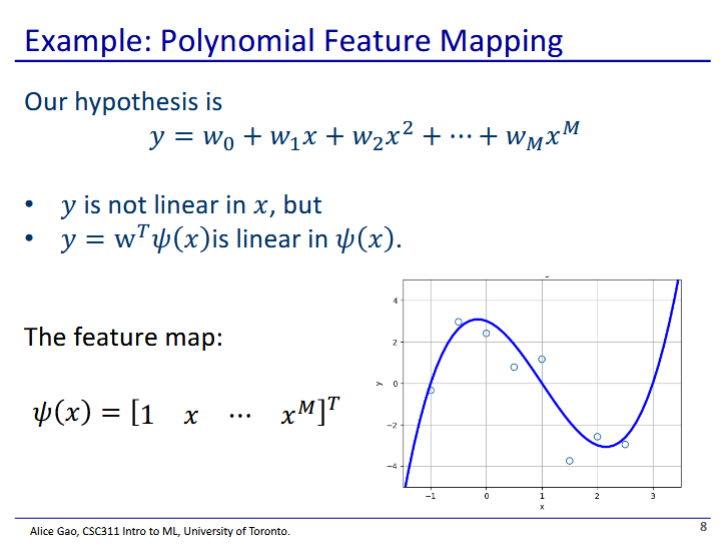
* Linear regression
  + We assume that a linear hypothesis can model the relationship we need
* But what if the relationship is not linear?
  + We can still use linear regression, using feature mapping



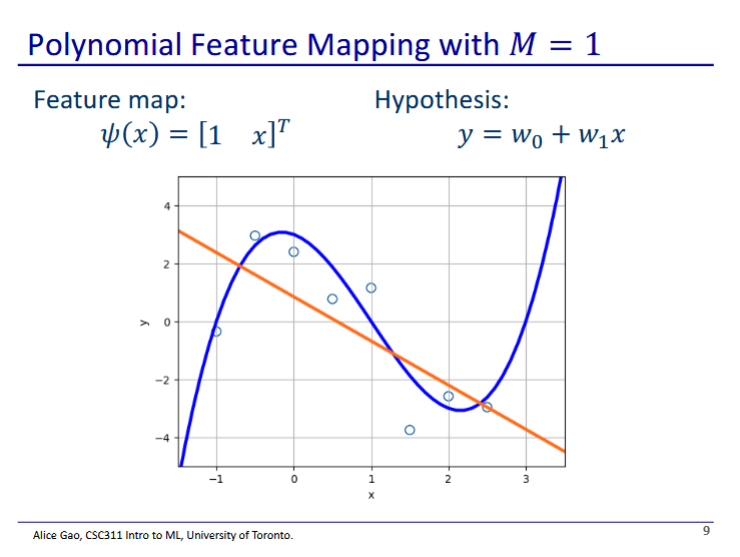
* We map our features to another space



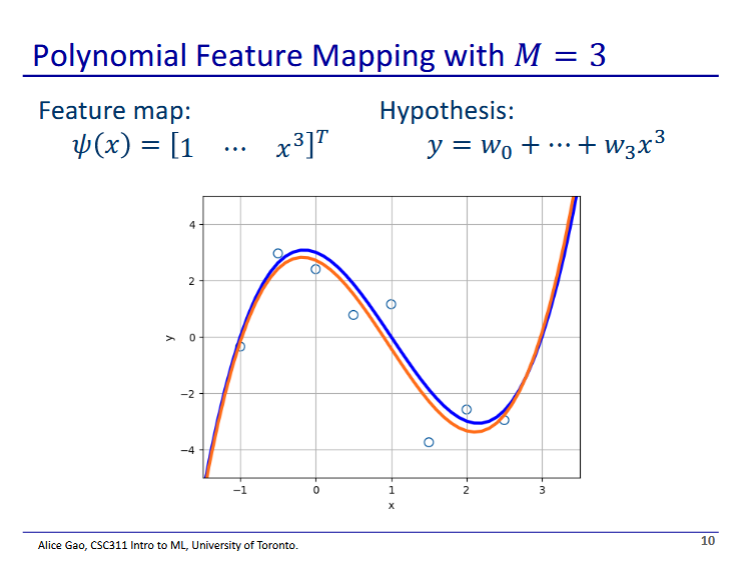
* , a function that maps one feature to a vector of multiple features
* In this case is polynomial, but can be any function of existing features



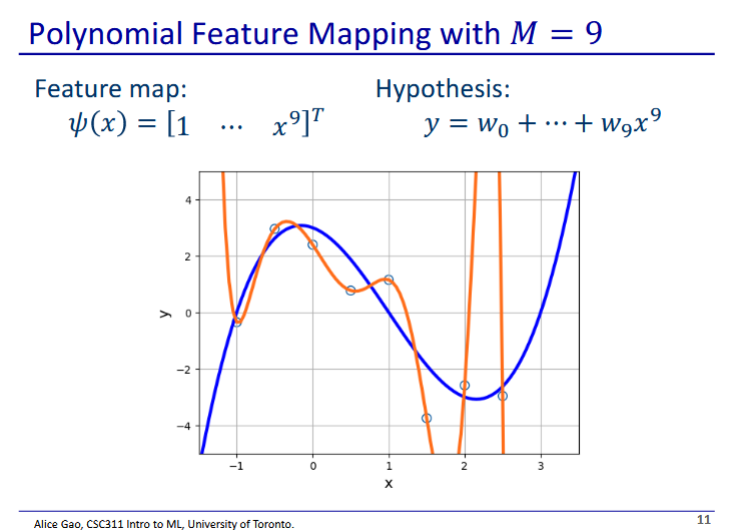
* How does our model change when we change the value of m?



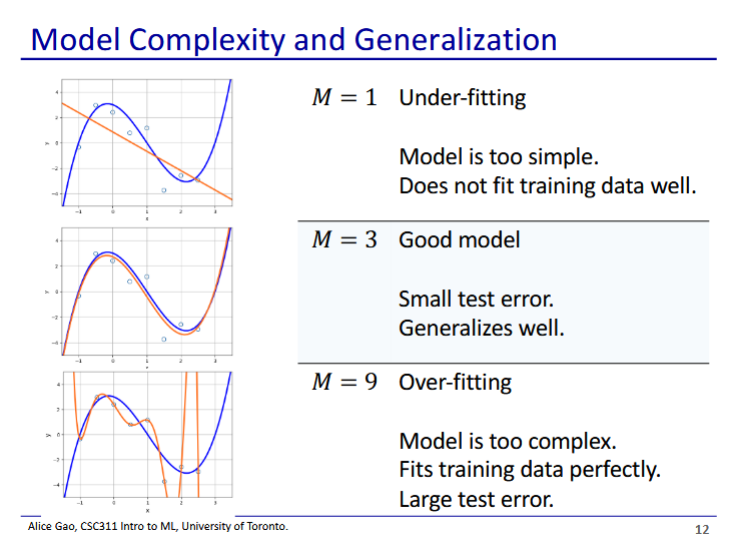
* m=1 is linear regression with 1 feature

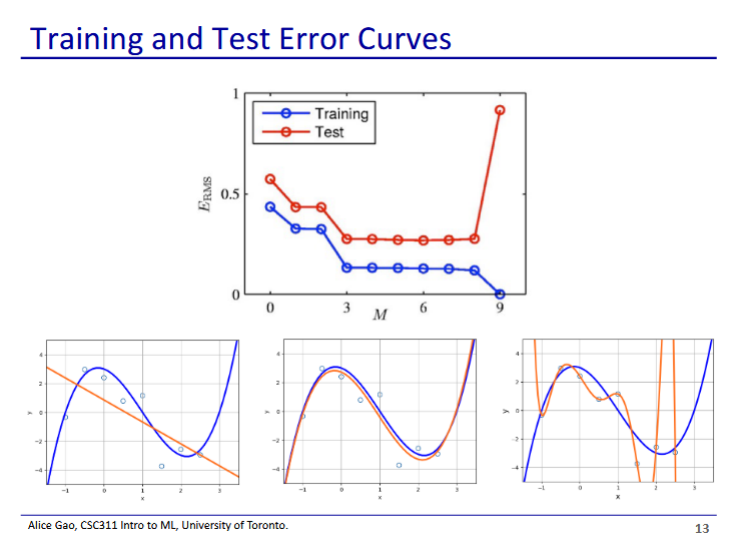


* When m is 3, our function is very good, close to the true curve
* This is likely as best as we can do, since we can never see the true function

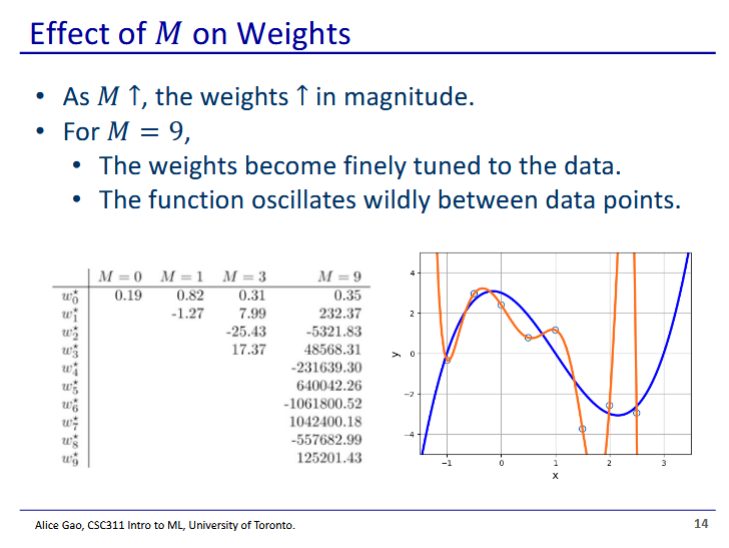


* What if we make M very large?
* We get a polynomial that perfectly hits every point in the dataset
  + But we have overfit, our curve doesn’t generalise well to test data

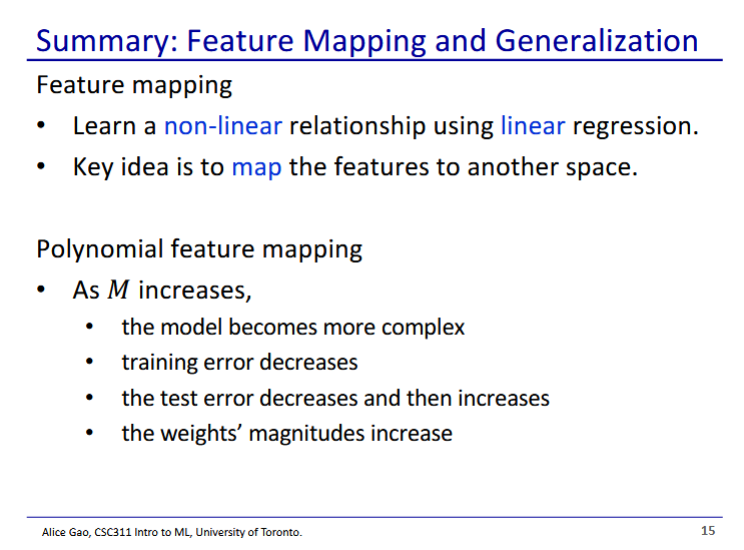




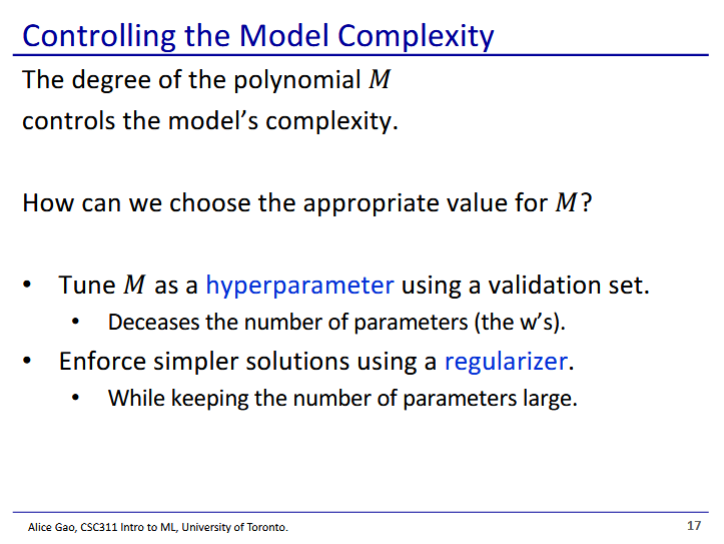
* As M increases, the training error decreases
  + Becomes 0 when m is 1 greater than the number of training points
  + Training error never increases when m increases
* As M increases, the test error first decreases and then increases
  + Decrease as the model begins fitting to data as m is increased
  + Then increases as model starts overfitting when m is too high
* To avoid overfitting, we should pick somewhere where the validation error is lowest

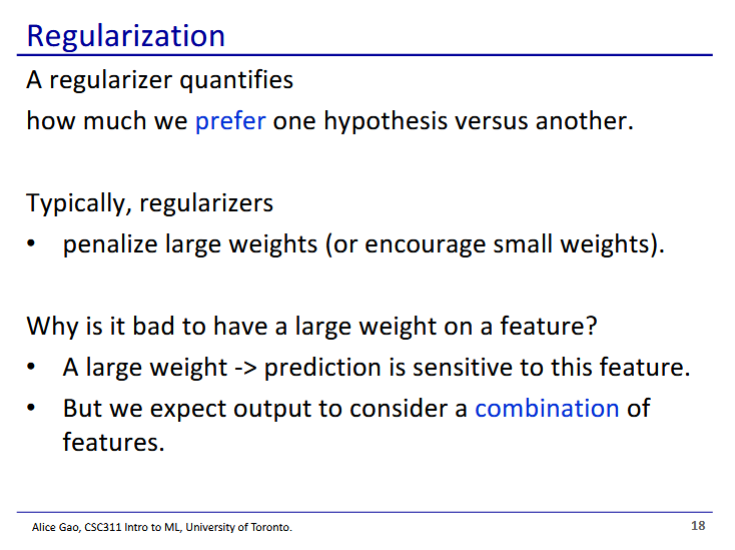


* As we increase M, the magnitude of the weights become very large
  + This is another way we can detect overfitting
  + Large magnitudes of weights is a sign the model is trying to finely tune to fit the training data exactly

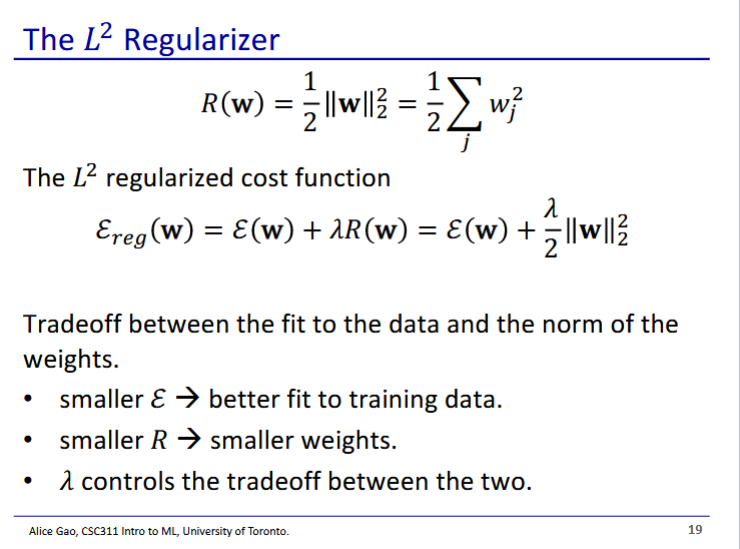




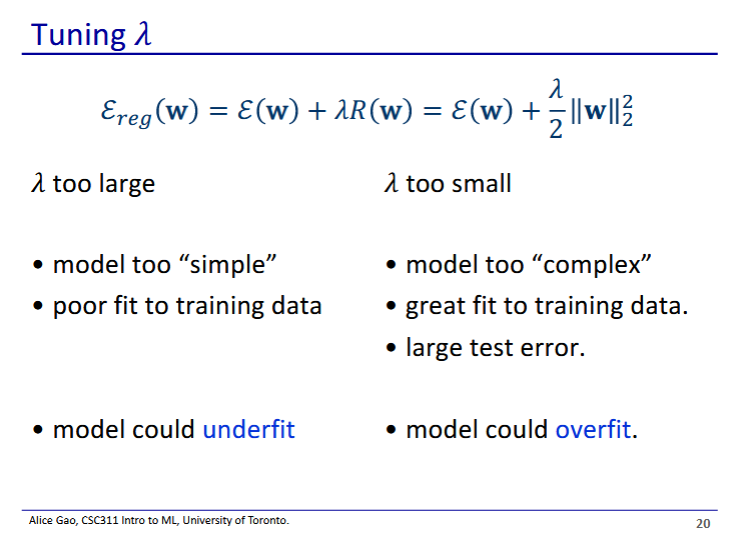




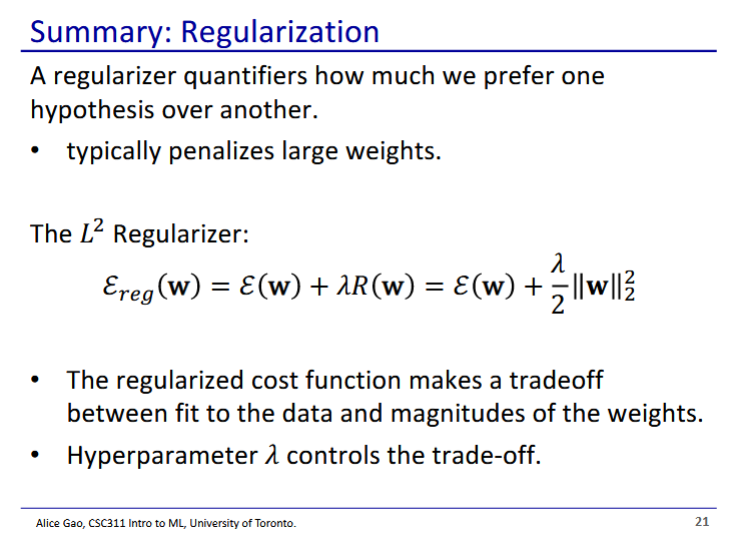
* Regularizer - we prefer one hypothesis versus another (typically the simpler ones)
  + We like some hypotheses more than others
* Why are large weights bad?
  + A feature with a larger weight is more important
  + This is not desirable as we want our model to consider many different features together

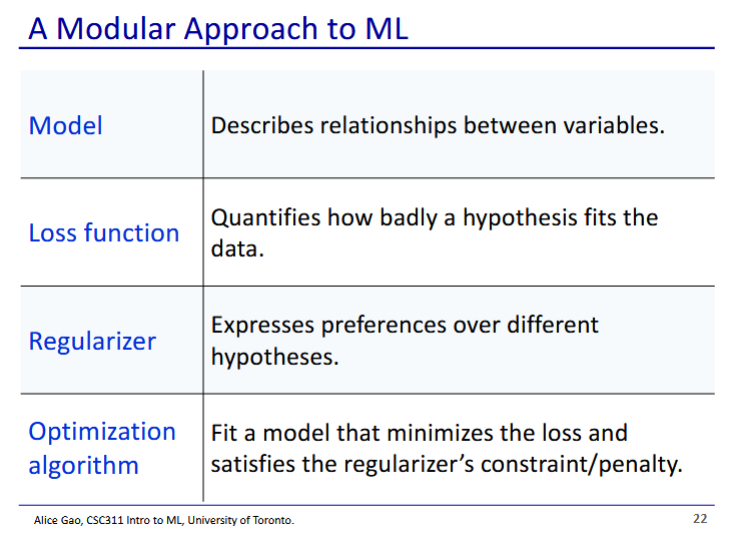


* We add a penalty term to our cost function to make a regularised cost function
  + The cost function is trying to fit the training data
  + The regularisation term is trying to limit the weights
  + These objectives compete, and force a tradeoff which hopefully prevents overfitting
    - Lambda controls the tradeoff



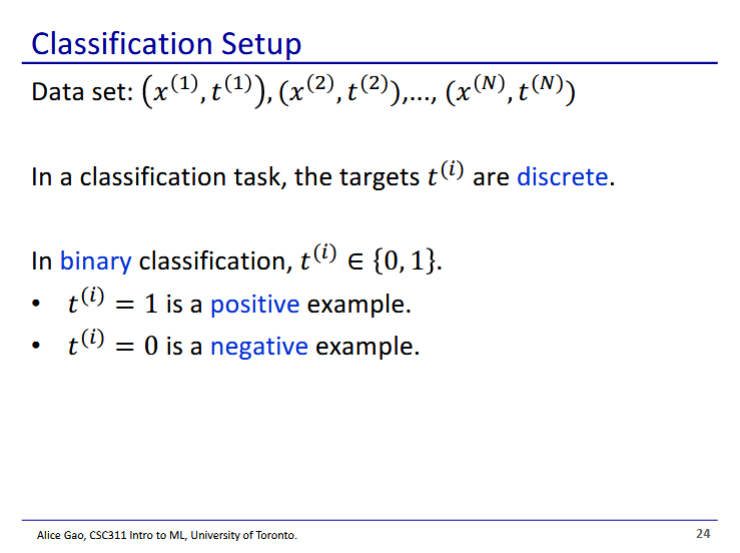
* Lambda is another hyperparameter
  + When lambda is too large, the regularisation term is too strong
    - We have very small weights and the model underfits
  + When lambda is too small, the regularisation term is too weak
    - We have very large weights and the model overfits



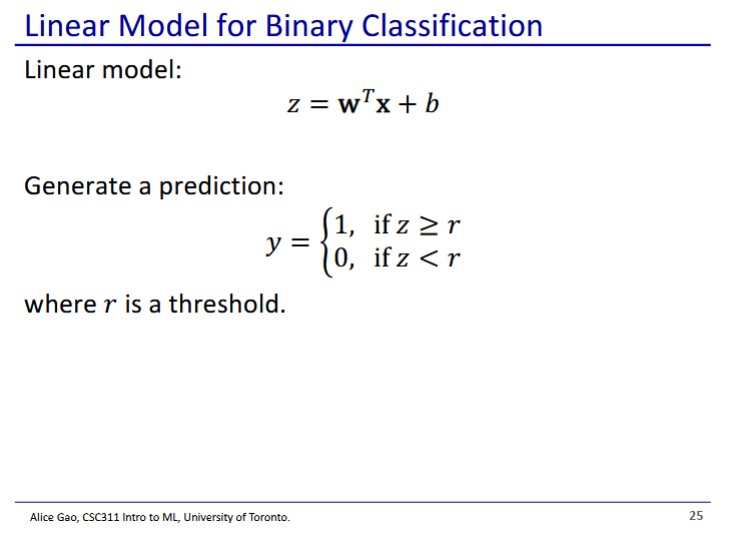


* Now we have a regulariser component that we can also add to our ML solutions
* We have only learned 2 optimisation algorithms so far
  + Gradient descent
  + Direct solution

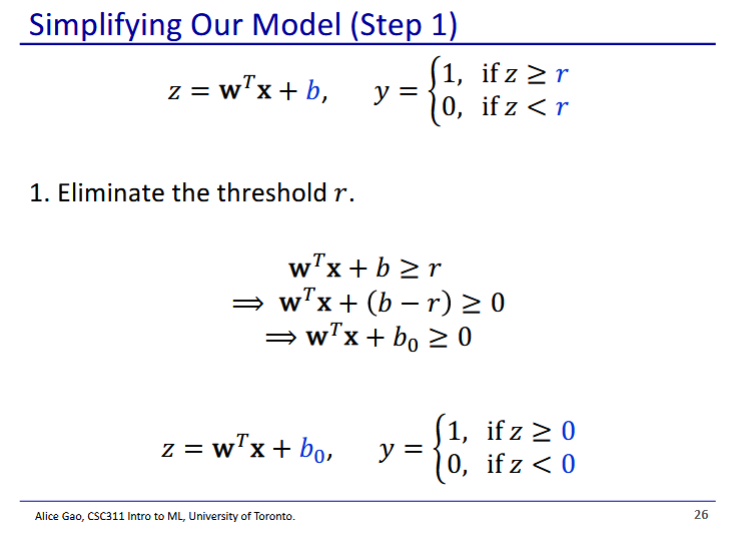




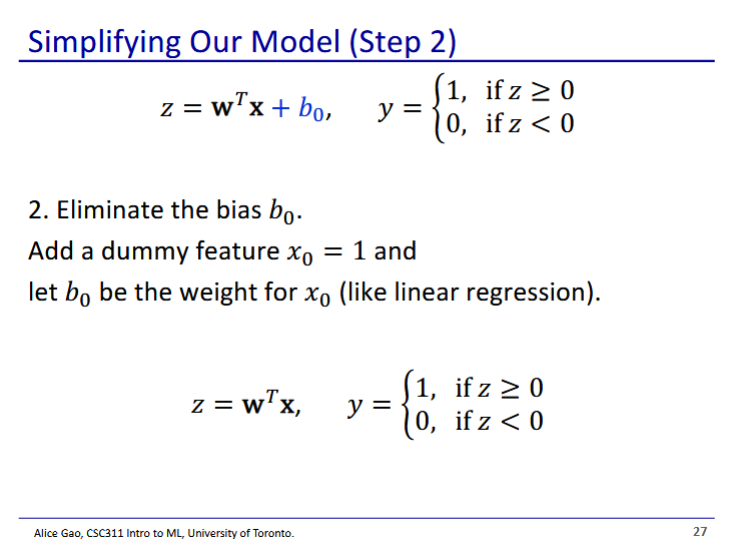
* Back to classification
* In this case is binary classification
  + Outputs can be 0 and 1, (1 and -1 is also sometimes used)



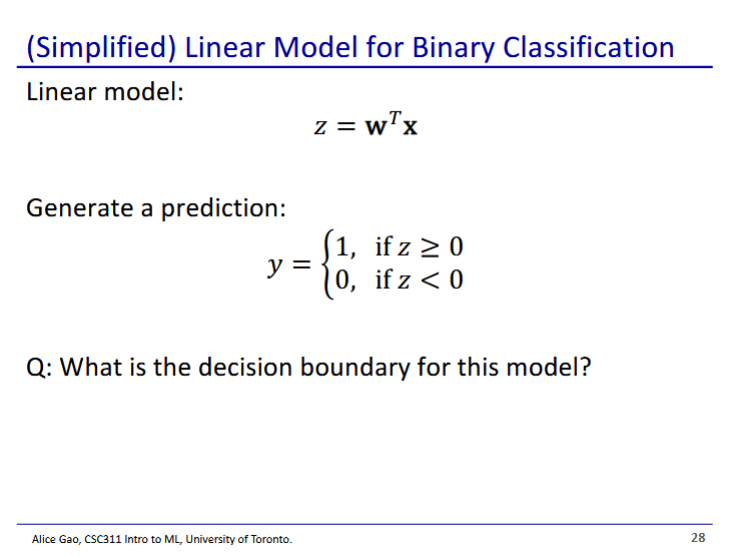
* We will use a linear model and a threshold to make our prediction



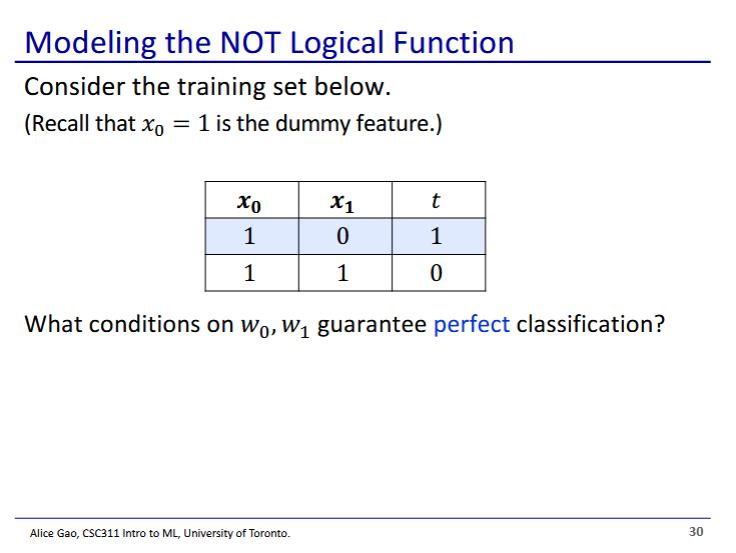
* First is to merge our bias and our threshold
  + Our new threshold is 0
  + Our new bias is
* Our bias and threshold are both arbitrary, so we can merge it into the bias



* Then we can do what we did before with linear regression and absorb the bias into the weights
  + Add a dummy feature, and apply the bias as a weight to the dummy feature



* The decision boundary is



* NOT logical function
  + When x1 is 0, t is 1
  + When x1 is 1, t is 0
  + x0 is the dummy feature
* Conditions:

To find weights for this bin. Clasif data set