+ Case study: Predicting sentiment: An intelligent restaurant review system. Goal: classifying sentiment of review. Input: sentences from reviews. Output: prediction of positive (+1) or negative (-1).

- A linear classifier associates every word with weight/coefficient, which says how this word is positively/negatively influential. The output is the weighted sum of the inputs. Then a score for that input is computed. If the score is greater than 0, the output y hat is +1; if the score is less than 0, the prediction is -1.

- For linear classifier, what we need to do is train the weights of these linear classifiers from data. Given some input data, we split it into training set and validation set. Then we feed the training set with some learning algorithm which is going to learn the weights associated with each word. Then we go back and evaluate it's accuracy on the validation set.

- Decision boundary (linear): Everything below the line has score greater than 0, and everything above the line has negative score.

- With two features/coefficients, the decision boundary is just a line in 2D plance. In general, we might have more coefficients than that. So, if you have three features that have no zero value, then what we really have is a plane that tries to separate the positive points from the negative ones. If you have more than three non-zero coefficients, then we are in high-dimensional space (hyperplane).

+ Exampel features: "awesome", "awful", "great".

+ What a linear classifier model does, is try to build a hyperplane that tries to separate the positives from the negatives. The hyperplane is associated with the score function:

$$Score(x_i) = w_0 + w_1*\#awesome + w_2*\#awful + w_3*\#great$$

- General notation:

- Output: y ---> {-1, +1}
- Inputs: x = (x[1], x[2], ..., x[d]) --- d-dim vector
- Notational conventions: x[j] = j'th input (scalar), h_j(x) = j'th feature (scalar), x_i = input of i'th data point (vector), x_i[j] = j'th input of i'th data point (scalar).
- Simple hyperplane: Model: y_i(hat) = sign(Score(x_i)), Score(x_i) = w_0 + w_1*x_i[1] + ... + w_d*x_i[d] = **w**_T***x_i**
- Our goal is to optimize the score above.

- More generic features (D_dimensional hyperplane)

- Model: y_i(hat) = sign(Score(x_i))
- Score(x_i) = w_0*h_0(x_i) + w_1*h_1(x_i) + ... + w_D*h_D(x_i) = sum(w_i*h_j(x_i)) = **w**_T***h*(**x_i**)

- In logistic regression, we don't just predict +1 or -1, we predict a probability.

+ Conditional probability example: The probability that "a review with 3 "awesome" and 1 "awful" is positive" is 0.9. So, out of the rows that have 3 awesomes and 1 awful, I expect on average that 90% of these are going to be positive reviews.

- Logistic regession:

- Score(x_i) = w_0*h_0(x_i) + w_1*h_1(x_1) + ... + w_D*h_D(x_i) = **w**_T***h(x_i)** --- How do we relate these scores which could be somewhere between -inf. and +inf.? Answer: a link function!
- A link function: P(hat)(y=+1|x_i) = g(**w**_T***h(x_i)**) <--- Generalized linear model.
- Generalized linear model: Squeezing (-inf., +inf.) to the interval [0, 1]. (There are other types of generalized linear models that don't squeeze scores to [0, 1].)

- The sigmoid/logit link function (for logistic regression):

sigmoid(Score) = 1/(1+e^(-Score))

- Logistic regression model:

- Score(x_i) = **w**_T***h(x_i)**
- P(y=+1|x_i, w) = Sigmoid(Score(x_i)) = 1/(1+e^(-Score(x_i))) = 1/(1+e^(-w_T*h(x_i))) = 1/(1+e^(-(w_0*h_0(x_i)+w_1*h_1(x_1)+...+w_D*h_D(x_i)))).

- When inputs are categorical variables (e.g., male/female) ---> Encoding ---> One-hot encoding: only one of these features has value 1, adn everything else is 0.

- Multiclass classification (1 versus all Approach)

- It has more than two classes and not just +1/01, but maybe have a thousand different categories.
- You train a classifier for each category. For example, +1 class for "points with y_i = triangle", -1 class for "points with y_i = heart OR donuts".
- We train classifier: P(hat)(y=+1|x).