Linear Regression

- *Classification models* predict a discrete class label ***y*** for an input observation ***x***.
- *Regression models* predict a real-valued outcome ***y*** for an input observation ***x***.
- Given a set of "training" observations, linear regression models produce a regression line that best fits the observed data.

  The equation for a line is: ***y = mx + b***
  so values of ***m*** and ***b*** are assigned to fit the data.

- The regression line is used to predict the output value for a new instance ***x***.

Multiple Linear Regression

In machine learning, we use *multiple* features to represent each instance (observation). This scenario is called *multiple linear regression* (but often just called linear regression).

$$y = w_0 + \sum_{i=1}^{N} w_i * f_i$$

We can write this formula more generally by assuming there is a special feature f_0 that has value 1.

$$y = \sum_{i=0}^{N} w_i * f_i \quad = W \bullet F$$

(called the *dot product*)

Learning for Linear Regression

- Weights are learned to produce estimates of y that are close to the true values of y in the training data.
- We want to minimize the difference between the *predicted* value of y and the *observed* value of y.
- A cost function, such as *sum-squared error*, is applied to the set of weights W:

$$\text{cost}(W) = \sum_{i=0}^{N} (y^i_{predicted} - y^i_{observed})^2$$

Logistic Regression

- Most NLP problems are classification tasks where we want a class label and ideally a probability of being in the class.
- But linear regression models produce a real number, <u>not</u> a probability.
- **Logistic regression** models use a linear function to estimate the probability of a class label.

  For binary classification, we want: P(*y* = true | *x*)

The Odds Ratio

An **odds ratio** is the ratio of two probabilities: the probability of being in a class vs. the probability of not being in the class:

$$\frac{P(y = \text{true} \mid x)}{1 - P(y = \text{true} \mid x)} \qquad \text{Range} = [0, \infty]$$

But we want to learn this with a linear predictor w * f, which has Range = [-∞, ∞]. So we use a logarithm:

$$\ln\left(\frac{P(y = \text{true} \mid x)}{1 - P(y = \text{true} \mid x)}\right) = W \bullet F$$

In general, we're using the **logit (log odds)** function:

$$\text{logit}(P(x)) = \ln\left(\frac{P(x)}{1 - P(x)}\right)$$

The Logistic Function

Using algebraic manipulation, we can solve the previous equation for the probabilities we want:

$$P(y = \text{true} \mid x) = \frac{1}{1 + e^{-W \bullet F}}$$

$$P(y = \text{false} \mid x) = \frac{e^{-W \bullet F}}{1 + e^{-W \bullet F}}$$

We can now use the linear function W • F for classification (see textbook for derivation):

$$\sum_{i=0}^{N} w_i * f_i > 0 \qquad \text{predicts } y = \text{true}$$

Maximum Entropy (MaxEnt) Modeling

- In NLP, we often have classification tasks that involve many categories (e.g., POS tags or Named Entity Types).
- **Multinomial logistic regression** (also called **maximum entropy modeling** or **MaxEnt**) generalizes to multiple classes.
- The family of classifiers that combine weights linearly and use the sum as an exponent are called ***exponential*** or ***log-linear*** models.
- The probability of class *c* given an input observation *x* is:

$$P(c \mid x) = 1/Z \;\; \exp\left(\sum_{i=0}^{N} w_i * f_i\right)$$

- Z is a normalizing factor that ensures the probabilities sum to 1. NOTE: exp(x) is the same as e^x

The MaxEnt Formula

$$P(c \mid x) = \frac{\exp\left(\sum_{i=0}^{N} w_{ci} * f_i\right)}{\sum_{c' \varepsilon C} \exp\left(\sum_{i=0}^{N} w_{c'i} * f_i\right)}$$

We define the **normalization factor Z** = $\sum_{c' \varepsilon C} \exp\left(\sum_{i=0}^{N} w_{c'i} * f_i\right)$

$$P(c \mid x) = \frac{1}{Z} \exp\left(\sum_{i=0}^{N} w_{ci} * f_i\right)$$

MaxEnt probability example

Suppose these weights have been learned:

	f1	f2	f3	f4	f5	f6
w_{VB}	.2	.8	4	.01	.1	.5
w_{NN}	.8	.07	-.2	.33	6	-1.3

And you have an example *x* that you want to classify, which has the following feature values:

	f1	f2	f3	f4	f5	f6
VB	0	1	0	1	1	0
NN	1	0	0	0	0	1

$$P(VB \mid x) = e^{(.8+.01+.1)} / (e^{(.8+.01+.1)} + e^{(.8-1.3)}) = .80$$

$$P(NN \mid x) = e^{(.8-1.3)} / (e^{(.8+.01+.1)} + e^{(.8-1.3)}) = .20$$

Sliding Window Classifiers

- For tagging problems, one option is to use a regular (non-sequential) classifier that looks at features surrounding the targeted word.
- We can create a classifier that encodes features for k words preceding w and k words following w. For example, if k=3 then:

$$w_{-3}\ w_{-2}\ w_{-1}\ \mathbf{w}\ w_1\ w_2\ w_3$$

- The classifier can then be applied to each word, one at a time, sliding this window from left to right.
- This approach can work well. But the decisions are local: the classifier must make a hard decision about a word before making decisions about subsequent words.

MEMMs vs. HMMs

- Maximum entropy Markov models (MEMMs) extend the MaxEnt classification model for sequence tagging.
- HMMs incorporate two probabilities: $P(label_i \mid label_{i-1})$ and $P(word_i \mid label_i)$. MEMMs allow us to encode a larger set of features into a sequential model.
- MEMMs make decisions for the entire sequence at once, like Viterbi decoding with HMMs.
- HMMs are a *generative model* that optimize for P(W|T), because we flipped the equation with Bayes Rule: argmax P(W|T)*P(T)
- MEMMs are a *discriminative model* that optimize for P(T|W).

MEMM modeling

- MEMMs train a single probabilistic model to estimate:

$$\underset{T}{\text{argmax}}\ P(T|W) = \underset{T}{\text{argmax}} \prod_i P(tag_i \mid word_i, tag_{i-1})$$

- MaxEnt is used to estimate the probability of a tag for word given the tag for the previous word as well as other features. Q is the set of states and O is the set of observations (words):

$$P(Q \mid O) = \prod_{i=1}^{N} P(q_i \mid q_{i-1}, o_i)$$

- More generally, we can encode multiple features as:

$$P(q \mid q', o) = \frac{1}{Z(o,q')} \exp\left(\sum_i w_i * f_i(o,q)\right)$$

Summary

- Logistic regression classifiers are commonly used for binary classification tasks because they are simple and provide probability estimates for a class.

- MaxEnt classifiers are also log-linear models that provide probabilities, but also allow for many category labels.

- MEMMs are widely used sequential tagging models that allow for rich feature sets and work quite well for many tasks.

- Conditional Random Fields (CRF) models are discriminative undirected probabilistic graphical models that are also widely used for sequence tagging. They work well, but training can be slow.