Why does it work?

- We have not addressed the question of why does this classifier performs well, given that the assumptions are unlikely to be satisfied.
- The linear form of the classifiers provides some hints.
-

Naïve Bayes: Two Classes

- In the case of two classes we have that:

$$\log \frac{P(v_j = 1 \mid x)}{P(v_j = 0 \mid x)} = \sum_i w_i x_i - b$$

- but since

$$P(v_j = 1 \mid x) = 1 - P(v_j = 0 \mid x)$$

- We get (plug in (2) in (1); some algebra):

$$P(v_j = 1 \mid x) = \frac{1}{1 + \exp(-\sum_i w_i x_i + b)}$$

- Which is simply the logistic (sigmoid) function

We have:
A = 1-B; Log(B/A) = -C.
Then:
Exp(-C) = B/A =
= (1-A)/A = 1/A – 1
= + Exp(-C) = 1/A
A = 1/(1+Exp(-C))

Why Does it Work?

- **Learning Theory** $Err_S(h) = |\{x \in S \mid h(x) \neq l\}| / |S|$
  - Probabilistic predictions are done via Linear [Statistical Queries] Models [Roth'99]
  - The low expressivity explains Generalization + Robustness
- **Expressivity (1: Methodology)**
  - Why is it possible to (approximately) fit the data with these models? Is there a reason to believe that these hypotheses minimize the empirical error?
  - In General, No. (Unless it some probabilistic assumptions happen to hold).
  - But: if the hypothesis does not fit the training data, augment set of features
  - But now, you actually follow the Learning Theory Protocol:
    - Try to learn a hypothesis that is consistent with the data
    - Generalization will be a function of the low expressivity
- **Expressivity (2: Theory)** [Garg&Roth (ECML'01)]:
  - Product distributions are "dense" in the space of all distributions. Consequently, for most generating distributions the resulting predictor's error is close to optimal classifier (that is, given the correct distribution)

What's Next?

- (1) If probabilistic hypotheses are actually like other linear functions, can we interpret the outcome of other linear learning algorithms probabilistically?
  - Yes
- (2) If probabilistic hypotheses are actually like other linear functions, can you train them similarly (that is, discriminatively)?
  - Yes.
  - Classification: Logistics regression/Max Entropy
  - HMM: can be learned as a linear model, e.g., with a version of Perceptron (Structured Models class)

Recall: Naïve Bayes, Two Classes

In the case of two classes we have:

$$\log \frac{P(v_j = 1 \mid x)}{P(v_j = 0 \mid x)} = \sum_i w_i x_i - b$$

but since

$$P(v_j = 1 \mid x) = 1 - P(v_j = 0 \mid x)$$

We get (plug in (2) in (1); some algebra):

$$P(v_j = 1 \mid x) = \frac{1}{1 + \exp(-\sum_i w_i x_i + b)}$$

Which is simply the logistic (sigmoid) function used in the neural network representation.

Conditional Probabilities

- (1) If probabilistic hypotheses are actually like other linear functions, can we interpret the outcome of other linear learning algorithms probabilistically?
  - **Yes**
- General recipe
  - Train a classifier f using your favorite algorithm (Perceptron, SVM, Winnow, etc). Then:

$$\Pr(y = 1|x) \approx P_{A,B}(f) \equiv \frac{1}{1 + \exp(Af + B)}, \text{ where } f = \mathrm{f}(x).$$

  - A, B can be tuned using a held out that was not used for training.
  - Done in LBJava, for example

Logistic Regression

- (2) If probabilistic hypotheses are actually like other linear functions, can you actually train them similarly (that is, discriminatively)?
- The logistic regression model assumes the following model:

$$P(y= +/-1 \mid x,w)= [1+\exp(-y(w^T x + b)]^{-1}$$

- This is the same model we derived for naïve Bayes, only that now we will not assume any independence assumption. **We will directly find the best w.**

How?

- Therefore training will be more difficult. However, the weight vector derived will be more expressive.
  - It can be shown that the naïve Bayes algorithm cannot represent all linear threshold functions.
  - On the other hand, NB converges to its performance faster.

Logistic Regression (2)

- Given the model:

$$P(y= +/-1 \mid x,w)= [1+\exp(-y(w^T x + b)]^{-1}$$

- The goal is to find the (w, b) that maximizes the log likelihood of the data: $\{x_1, x_2 \ldots x_m\}$.
- We are looking for (w,b) that minimizes the negative log-likelihood

$$\min_{w,b} \sum_1^m \log P(y= +/-1 \mid x,w)= \min_{w,b} \sum_1^m \log[1+\exp(-y_i(w^T x_i + b)]$$

- This optimization problem is called Logistics Regression

- Logistic Regression is sometimes called the Maximum Entropy model in the NLP community (since the resulting distribution is the one that has the largest entropy among all those that activate the same features).

Logistic Regression (3)

- Using the standard mapping to linear separators through the origin, we would like to minimize:

$$\min_w \sum_1^m \log P(y= +/-1 \mid x,w) = \min_w \sum_1^m \log[1+\exp(-y_i(w^T x_i)]$$

- To get good generalization, it is common to add a regularization term, and the regularized logistics regression then becomes:

$$\min_w f(w) = \tfrac{1}{2} w^T w + C \sum_1^m \log[1+\exp(-y_i(w^T x_i)]$$

Regularization term — Empirical loss

Where C is a user selected parameter that balances the two terms.

Comments on discriminative Learning

$$\min_w f(w) = \tfrac{1}{2} w^T w + C \sum_1^m \log[1+\exp(-y_i w^T x_i)],$$

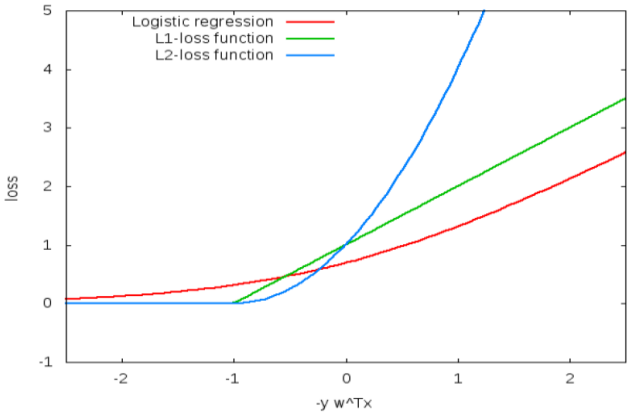
Regularization term — Empirical loss

Where C is a user selected parameter that balances the two terms.

- Since the second term is the loss function
- Therefore, regularized logistic regression can be related to other learning methods, e.g., SVMs.
- L_1 SVM solves the following optimization problem:

  $\min_w f_1(w) = \tfrac{1}{2} w^T w + C \sum_1^m \max(0,1-y_i(w^T x_i))$
- L_2 SVM solves the following optimization problem:

  $\min_w f_2(w) = \tfrac{1}{2} w^T w + C \sum_1^m (\max(0,1-y_i w^T x_i))^2$



Optimization: How to Solve

- All methods are iterative methods, that generate a sequence w_k that converges to the optimal solution of the optimization problem above.
- Many options within this category:
  - Iterative scaling: Low cost per iteration, slow convergence, updates each w component at a time
  - Newton methods: High cost per iteration, faster convergence
    - non-linear conjugate gradient; quasi-Newton methods; truncated Newton methods; trust-region newton method.
    - Limited memory BFGS is very popular
  - Stochastic Gradient Decent methods
    - The runtime does not depend on n=#(examples); advantage when n is very large.
    - Stopping criteria is a problem: method tends to be too aggressive at the beginning and reaches a moderate accuracy quite fast, but it's convergence becomes slow if we are interested in more accurate solutions.

Summary

- (1) If probabilistic hypotheses are actually like other linear functions, can we interpret the outcome of other linear learning algorithms probabilistically?
  - **Yes**

- (2) If probabilistic hypotheses are actually like other linear functions, can you train them similarly (that is, discriminatively)?
  - **Yes.**
    - **Classification: Logistic regression/Max Entropy**
    - **HMM: can be trained via Perceptron (Structured Learning Class: Spring 2016)**

Conditional Probabilities

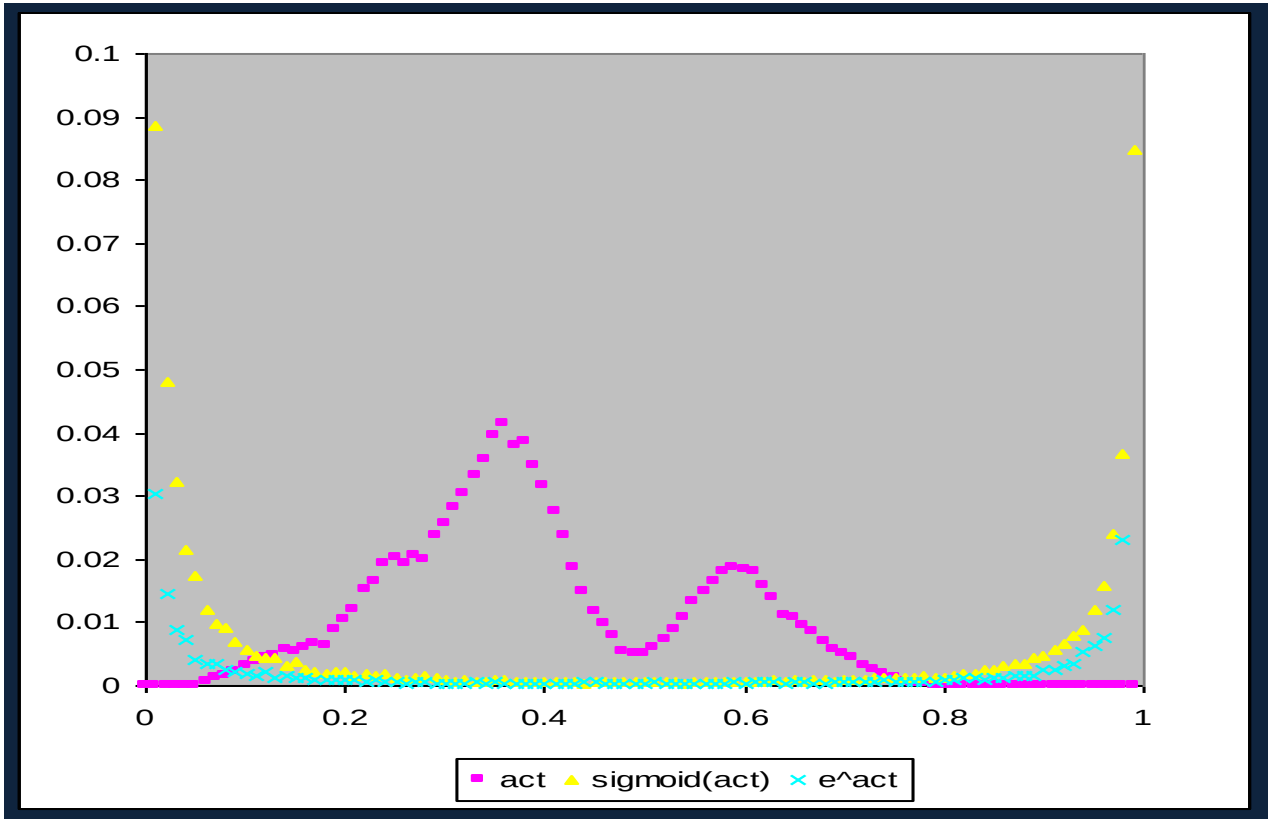
- Data: Two class

The plot shows a (normalized) histogram of examples as a function of the dot product

$$act = (w^T x + b)$$

and a couple other functions of it.

In particular, we plot the positive Sigmoid:



$$P(y= +1 \mid x,w)= [1+\exp(-(w^T x + b)]^{-1}$$

Is this really a probability distribution?

Conditional Probabilities

Plotting: For example z:

y=Prob(label=1 | f(z)=x)

(Histogram: for 0.8, # (of examples with f(z)<0.8))

Claim: Yes; If Prob(label=1 | f(z)=x) = x

Then f(z) = f(z) is a probability dist.

That is, **yes,** if the graph is linear.

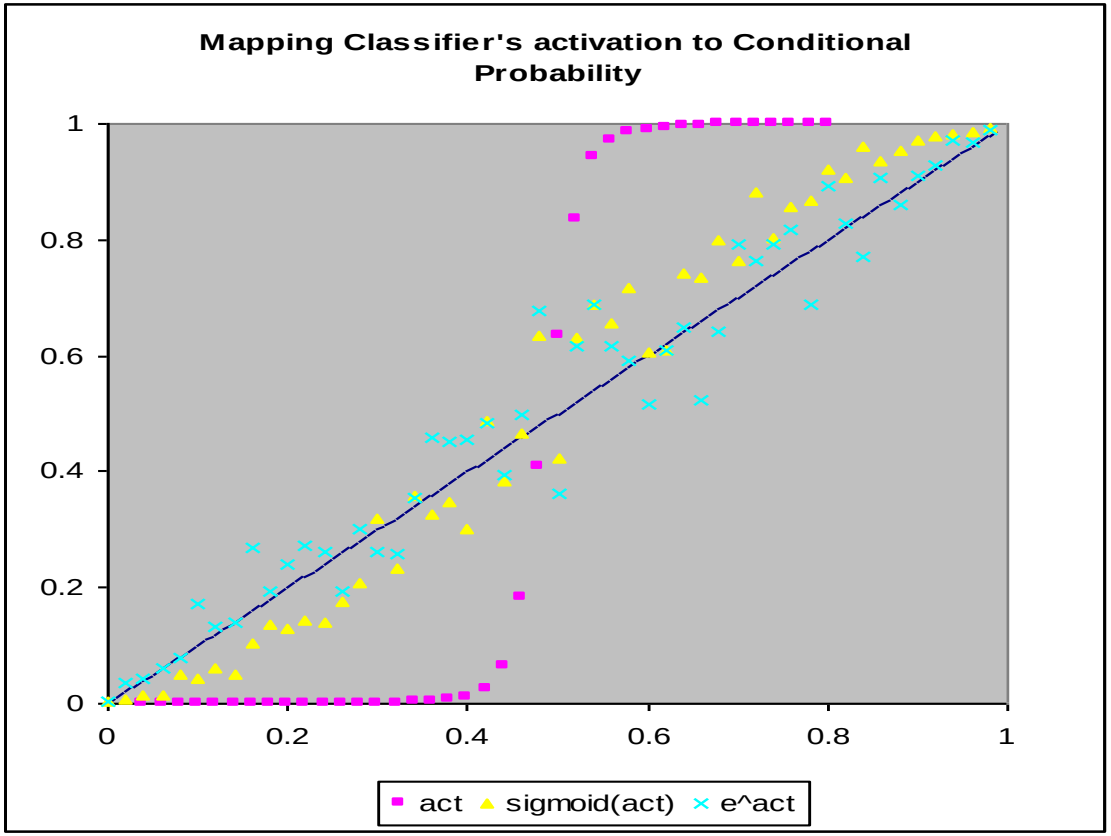
Theorem: Let X be a RV with distribution F.

(1) F(X) is uniformly distributed in (0,1).

(2) If U is uniform(0,1), $F^{-1}(U)$ is distributed F, where $F^{-1}(x)$ is the value of y s.t. F(y) =x.

Alternatively:

f(z) is a probability if: $Prob_U \{z | Prob[(f(z)=1 \leq y]\} = y$



Plotted for SNoW (Winnow). Similarly, perceptron; more tuning is required for SVMs.