Probabilistic Classification

Previous lecture: the perceptron

Advantages:

- Intuitive
- Simple to implement

Disadvantages:

- No probabilities
- Can’t handle non-linear datasets

Why probabilities?

- Interpretability: scores don’t tell us much about the confidence of the model
- Knowing what the model knows (and what it doesn’t)
- Ability to incorporate prior knowledge

Two approaches in today’s lecture:

- Generative: Naive Bayes
- Discriminative: Logistic regression

Classification with Bayes

What we want to do:

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} P(y|x)$$

Bayes Rule:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

In plain English:

$$posterior = \frac{likelihood * prior}{evidence}$$

Should we care about the evidence?

- No if we only want the class prediction
- Yes if we want to know what kind of inputs our model should be good at

Naive Bayes

Add the feature function:

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} P(\phi(x)|y)P(y)$$

Naive Bayes: assume each feature φ_i is **independent given the class:**

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} P(y) \prod_i P(\phi_i(x)|y)$$

How do we train the model?

Supervised learning, in this case: **Count** and **divide!**

Maximum likelihood estimate for Naive Bayes

Given labeled training data of the form: $D = \{(x^1, y^1), \dots (x^M, y^M)\}$

Find the parameters w_1, w_2 that maximize the likelihood L of D under the model:

Probability of one instance? $P(x, y) = P(x|y)P(y)$

$$w_1^\star, w_2^\star = \arg\max_{w_1, w_2} \prod_{(x,y) \in D} P(w_1; y) \prod_i P(w_2; \phi_i(x)|y)$$

- w_1: **Count** the times each class appears, **divide** with the number of instances
- w_2: **Count** the times each feature appears in instances of each class, **divide** with sum for all classes

What did we get by being naive?

Generative vs Discriminative

Generative models like Naive Bayes can generate text/instance given the class:

- Can ask the model what an instance of a certain class looks like
- it can be seen as a class conditional language model
- Help learning when we don’t have much training data

Discriminative models:

- Model the class prediction directly
- More flexibility in modelling features

From generative (back) to discriminative

All we want is to predict the class:

$$\hat{y} = \underset{y \in \mathcal{Y}}{\arg\max}\, P(y|x)$$

And we are still happy to use a linear model

$$w \cdot \phi(x)$$

Recall the binary linear classifier
we learned with the perceptron:

$$\hat{y} = sign(w \cdot \phi(x))$$

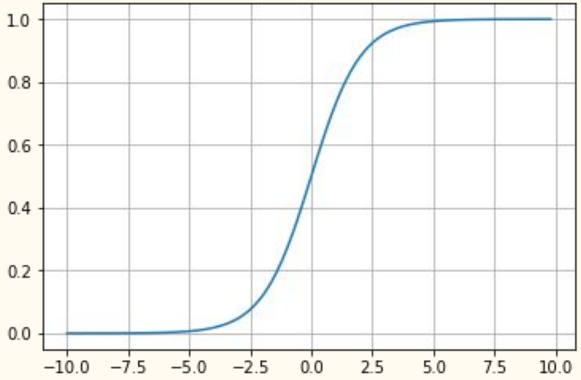
Logistic regression

Recall the binary linear classifier we learned with the perceptron:

$$\hat{y} = sign(w \cdot \phi(x))$$

Push the dot product through the sigmoid function:

$$\sigma(z) = \frac{1}{1+exp(-z)}$$



The binary logistic regression classifier (labels are 0, 1):

$$P(\hat{y} = 1) = \sigma(w \cdot \phi(x))$$

How to learn the parameters?

Supervised learning! Given labeled training data of the form:

$$D = \{(x^1, y^1), \ldots (x^M, y^M)\}$$

Learn weights w that **generalize** well to new instances

Naive Bayes has closed form solution for MLE (count and divide using the data)

In the case of the logistic regression this is not possible; we will define a learning **objective** but then look at the algorithm to find the weights.

Perceptron is a particular objective-algorithm combination

Objective

Maximize the likelihood of the data under the model:

$$L(\hat{y}, y) = P(\hat{y} = 1)^y (1 - P(\hat{y} = 1))^{1-y}$$

Recall that P(ŷ) has two discrete options, 1 and 0

Minimize the negative log likelihood (NLL):

$$NLL(\hat{y}, y) = -y \log P(\hat{y} = 1) - (1 - y) \log(1 - P(\hat{y} = 1))$$

Often referred to as the cross entropy loss

Objective

Plugging in the logistic regression function: $P(\hat{y} = 1) = \sigma(w \cdot \phi(x))$

And all the training data:

$$D = \{(x^1, y^1), \ldots (x^M, y^M)\}$$

$$w^\star = \arg\min_w \sum_{(x,y) \in D} -y \log \sigma(w \cdot \phi(x)) - (1 - y) \log(1 - \sigma(w \cdot \phi(x)))$$

Unlike the perceptron it is not enough for the correct label to be the highest scoring; the incorrect one must score as low as possible

Optimizing the objective

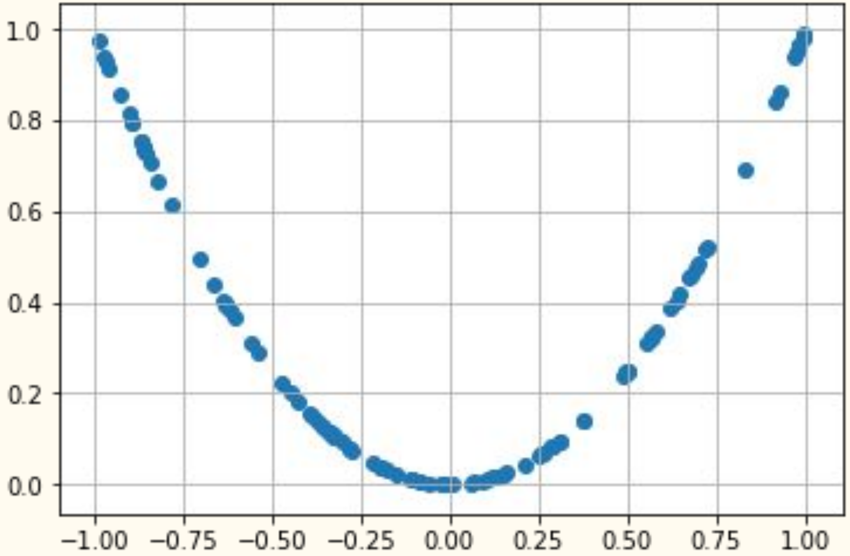
We have a function that we want to minimize wrt some parameters.

Sometimes there are closed form solutions (naive Bayes), otherwise?

- Random guesses at parameters and objective evaluations
- Take into account the shape of the function

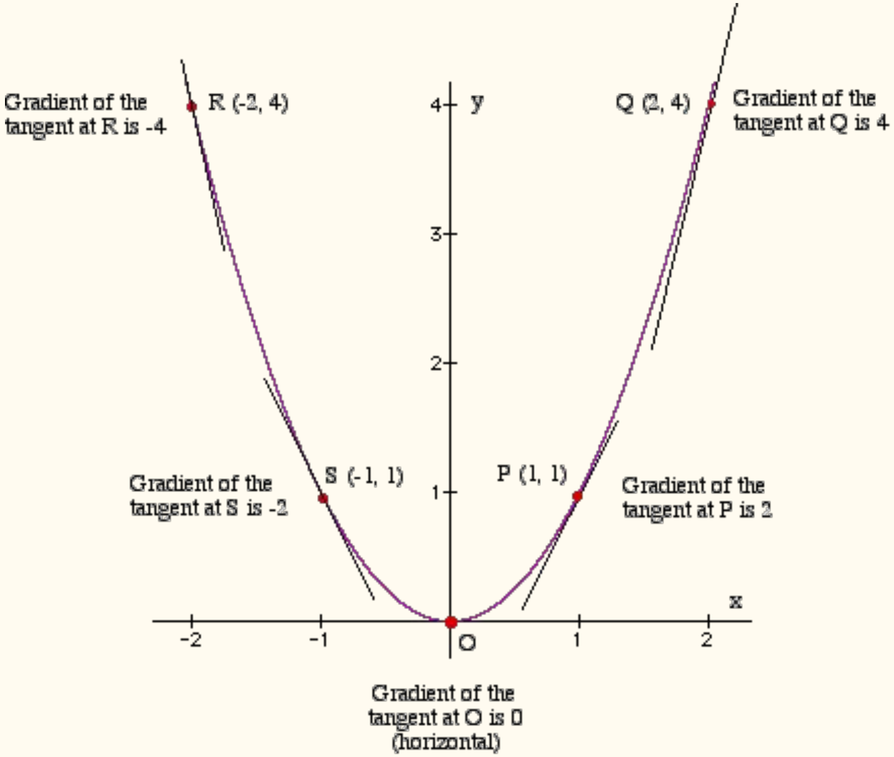
Optimizing a simple function

$$f(x) = x^2$$



With a number of random guesses at x we can get close to the minimum

Gradients



$$f(x) = x^2$$

$$\nabla_x f(x) = 2x$$

Gradients guide us to the minimum, where the gradient in this case is 0

http://bestmaths.net/online/index.php/year-levels/year-12/year-12-topic-list/first-principles/

Gradients for logistic regression

Objective (reminder):

$$NLL(w;\mathcal{D}) = \sum_{(x,y)\in\mathcal{D}} -y \log \sigma(w \cdot \phi(x)) - (1-y)\log(1-\sigma(w \cdot \phi(x)))$$

Gradient with respect weight for feature φ_j:

$$\frac{\partial NLL(w;\mathcal{D})}{\partial w_j} = \sum_{(x,y)\in D} \sigma(w \cdot \phi(x)) - y)\phi_j(x)$$

Interpretation: the weight should be updated proportionally to the loss of the model multiplied by the value of the feature for each instance

Binary to Multiclass

The sigmoid "squishes" a real number z to the 0..1 range

$$\sigma(z) = \frac{1}{1+exp(-z)}, z \in \Re$$

The softmax "squishes" a vector $\boldsymbol{z}$ of k real numbers to the probability simplex

$$softmax(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{k} \exp(z_j)}, z \in \Re^k$$

Multinomial logistic regression:

$$P(\hat{y} = y) = \frac{\exp(w_y \cdot \phi(x))}{\sum_{y' \in \mathcal{Y}} \exp(w_{y'} \cdot \phi(x))}$$

Still a linear classifier:

$$\arg\max_{y \in \mathcal{Y}} P(\hat{y} = y) = \arg\max_{y \in \mathcal{Y}} w_y \cdot \phi(x)$$

Generative vs Discriminative

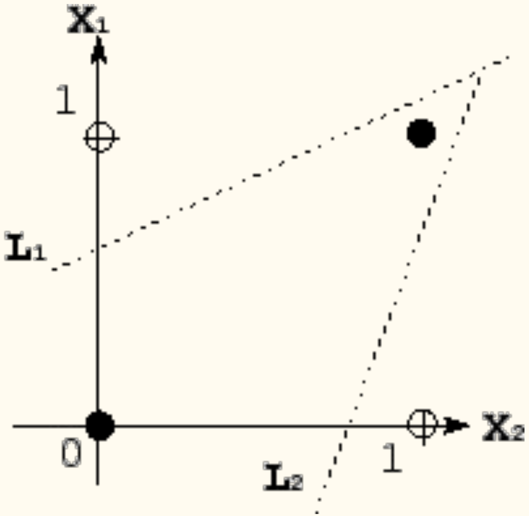
Which one would you choose?

- If not a lot of training data, generative can avoid overfitting (must learn to generate the data too)
- If a lot of features are likely to matter and not sure about their correlations, discriminative can be good
- Naive Bayes is trivial to train
- Logistic regression is the standard at this point for linear classifiers

Limitations of linear classifiers

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$$Y = X_1 \oplus X_2$$



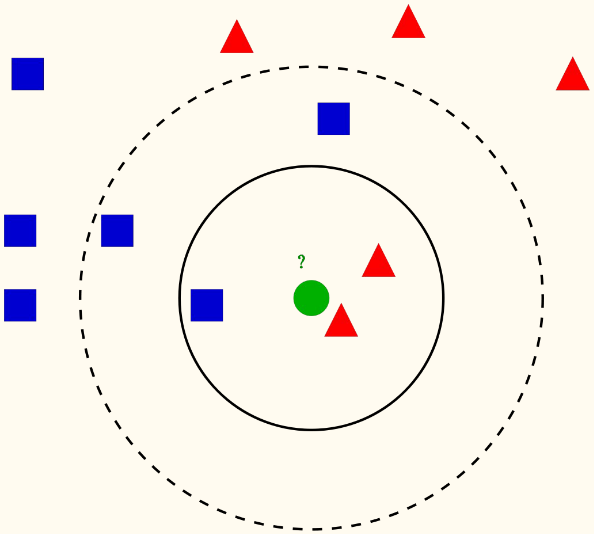
They assume that our data as represented by $\varphi(x)$ is linearly separable. It is easy construct dataset that are not.

http://www.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk-thesis-html/node19.html

Alternatives?

- Engineer better features
- K nearest neighbors
- Kernel methods
- Neural networks

K-nearest neighbors



- Assumes a distance metric (can be learned)
- No training, just memorize the training data
- Classify according to the nearest neighbor(s)

https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

Advantages:

- Non-linear
- Non-parametric

Disadvantages:

- With large datasets can be computation/memory heavy
- No feature learning

Kernel methods

We can replace the weights with calculations involving the training instances:

Binary perceptron

$$\hat{y} = sign(w \cdot \phi(x))$$

α's represent the mistakes made in training

$$\hat{y} = sign \sum_{(x,y)^i \in \mathcal{D}} \alpha^i y^i (\phi(x^i) \cdot \phi(x))$$

Perceptron with (non-linear) kernels

$$\hat{y} = sign \sum_{(x,y)^i \in \mathcal{D}} \alpha^i y^i K(x^i, x)$$

Support vector machines also use kernels, but in addition they find the max margin separating hyperplane

Bibliography

- Jurafsky and Martin chapter 4 (Naive Bayes) and chapter 5 (logistic regression)
- André Martins gave a 3hr lecture covering a lot of what we discussed at LxMLS
- Recent work on modelling the evidence by Nalisnick et al. (2019) with a lot of references on learning the evidence P(x)
- Ng and Jordan (2002) on generative vs discriminative
- Historical note: until 2010 or so, logistic regression was referred to as maximum entropy or maxent