CS5489
Lecture 2.2: Logistic Regression

Kede Ma

City University of Hong Kong (Dongguan)

Slide template by courtesy of Benjamin M. Marlin

Classification with Generative Models

- Steps to build a classifier
  1. Collect training data (features $\mathbf{x}$ and class labels y)
  2. Learn class-conditional distribution, $p(\mathbf{x}|y)$
  3. Use Bayes' rule to calculate class probability, $p(y|\mathbf{x})$
- **Note**: The data is used to learn the class-conditional distribution; the classifier is secondary
  - Density estimation is an "ill-posed" problem - which density to use? how much data is needed?
- Advice from Vladimir Vapnik (inventor of SVM)

> When solving a problem, try to avoid solving a more general problem as an intermediate step

- Discriminative solution
  - Solve for the classifier $p(y|\mathbf{x})$ directly!

Terminology

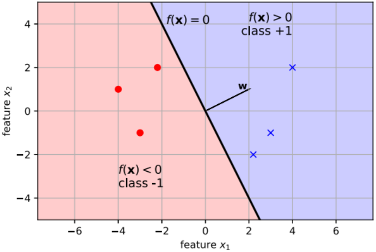
- **"Discriminative"** - learn to directly discriminate the classes apart using the features
- **"Generative"** - learn models of how the features are generated from different classes

Linear Classifier

- Setup
  - Observation (feature vectors) $\mathbf{x} \in \mathbb{R}^N$
  - Class $y \in \{-1, +1\}$
- Goal
  - Calculate a linear function of the feature vector $\mathbf{x}$:
    $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{j=1}^{N} w_j x_j + b$
    - $\mathbf{w} \in \mathbb{R}^N$ is the weight vector of the linear function
    - Multiply each feature value with a weight, and then add together
  - Predict from the value:
    - If $f(\mathbf{x}) > 0$ then predict Class $y = 1$
    - If $f(\mathbf{x}) < 0$ then predict Class $y = -1$
    - Equivalently, $y = \text{sign}(f(\mathbf{x}))$

Geometric Interpretation

- The linear classifier separates the feature space into 2 half-spaces
  - Each corresponds to feature values belonging to Class $+1$ and Class -1
  - The class boundary is normal (orthogonal or perpendicular) to $\mathbf{w}$
    - Also called the **separating hyperplane**
- Example: $\mathbf{w} = [2, 1]^T$, $b = 0$



Separating Hyperplane

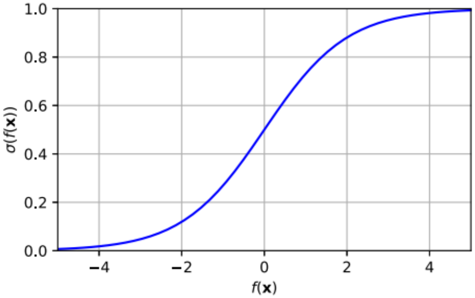
- In an N-dimensional feature space, the parameters are $\mathbf{w} \in \mathbb{R}^N$
- The equation $\mathbf{w}^T\mathbf{x} + b = 0$ defines an $N-1$-dimensional linear surface:
  - For $N = 2$, $\mathbf{w}$ defines a 1-D line
  - For $N = 3$, $\mathbf{w}$ defines a 2-D plane
  - ...
  - In general, we call it a hyperplane

Learning the Classifier

- How to set the classifier parameters $(\mathbf{w}, b)$?
  - Learn them from training data!
- Classifiers differ in the objectives used to learn the parameters $(\mathbf{w}, b)$
  - We will look at two examples:
    - Logistic regression
    - Support vector machine (SVM)

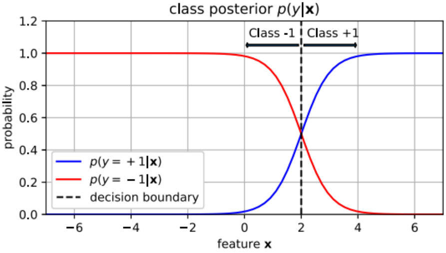
Logistic Regression

- Logistic regression takes a probabilistic approach
- Need to map the function value $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ to a probability value between 0 and 1
  - Sigmoid function maps from real number to interval $[0, 1]$
  - $\sigma(z) = \frac{1}{1+e^{-z}}$, for $z \in \mathbb{R}$



Logistic Regression

- Given a feature vector $\mathbf{x}$, the probability of a class is
  - $p(y=+1|\mathbf{x}) = \sigma(f(\mathbf{x}))$
  - $p(y=-1|\mathbf{x}) = 1 - \sigma(f(\mathbf{x})) = \sigma(-f(\mathbf{x}))$
  - Equivalently, $p(y|\mathbf{x}) = \sigma(yf(\mathbf{x}))$
- Note: here we are directly modeling the class posterior probability!
  - Not the class-conditional $p(\mathbf{x}|y)$



Learning the Parameters

- Given the data set $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)}), i = 1, \ldots, M\}$, learn the function parameters $(\mathbf{w}, b)$ using MLE
- Maximize the conditional log likelihood of the data $\mathcal{D}$:

$$(\mathbf{w}^\star, b^\star) = \arg\max_{\mathbf{w},b} \frac{1}{M}\sum_{i=1}^{M} \log p(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}; b)$$
$$= \arg\max_{\mathbf{w},b} -\frac{1}{M}\sum_{i=1}^{M} \log\left(1 + \exp\left(-y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b)\right)\right)$$

- To prevent **overfitting**, add a prior distribution on $\mathbf{w}$
  - Assume Gaussian distribution on $p(\mathbf{w})$ with variance $C/2$

$$p(\mathbf{w}) \propto \exp(-\frac{1}{C}\mathbf{w}^T\mathbf{w})$$

Learning the Parameters

- Now maximize

$$
\begin{aligned}
(\mathbf{w}^\star, b^\star) &= \arg\max_{\mathbf{w},b} \frac{1}{M}\sum_{i=1}^{M} \log p(\mathbf{w}|y^{(i)}, \mathbf{x}^{(i)}; b)\\
&= \arg\max_{\mathbf{w},b} \frac{1}{M}\sum_{i=1}^{M} \log \frac{p(\mathbf{w}, y^{(i)}|\mathbf{x}^{(i)}; b)}{p(y^{(i)}|\mathbf{x}^{(i)})}\\
&= \arg\max_{\mathbf{w},b} \frac{1}{M}\sum_{i=1}^{M} \log \frac{p(\mathbf{w})p(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}; b)}{p(y^{(i)}|\mathbf{x}^{(i)})}\\
&= \arg\max_{\mathbf{w},b} \log p(\mathbf{w}) + \frac{1}{M}\sum_{i=1}^{M} \log p(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{w}; b)
\end{aligned}
$$

Learning the Parameters

- Equivalently,

$$(\mathbf{w}^\star, b^\star) = \arg\min_{\mathbf{w},b} \frac{1}{C}\mathbf{w}^T\mathbf{w} + \frac{1}{M}\sum_{i=1}^{M} \log\left(1 + \exp\left(-y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b)\right)\right)$$

- The first term is the regularization term
  - Note: $\mathbf{w}^T\mathbf{w} = \sum_{j=1}^{N} w_j^2$
  - Penalty term keeps entries in $\mathbf{w}$ from getting too large
  - C is the regularization **hyperparameter**
    - Larger C values allow large values in $\mathbf{w}$
    - Smaller C values discourage large values in $\mathbf{w}$

Learning the Parameters

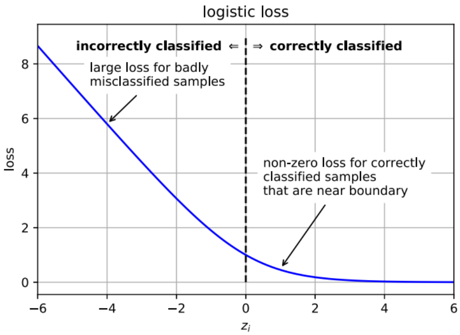
$$(\mathbf{w}^\star, b^\star) = \arg\min_{\mathbf{w},b} \frac{1}{C}\mathbf{w}^T\mathbf{w} + \frac{1}{M}\sum_{i=1}^{M} \log\left(1 + \exp\left(-y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b)\right)\right)$$

- The second term is the data-fit term
  - Wants to make the parameters $(\mathbf{w}, b)$ to well fit the data
  - Defining $z^{(i)} = y^{(i)} f(\mathbf{x}^{(i)})$, we have the following interesting observation
    - $z^{(i)} > 0$, when sample $\mathbf{x}^{(i)}$ is classified correctly
    - $z^{(i)} < 0$, when sample $\mathbf{x}^{(i)}$ is classified incorrectly
    - $z^{(i)} = 0$, when sample $\mathbf{x}^{(i)}$ is on classifier boundary

Logistic Loss Function

- Definition

$$L(z) = \log(1 + \exp(-z))$$



Learning the Parameters

$$(\mathbf{w}^\star, b^\star) = \arg\min_{\mathbf{w},b} \underbrace{\frac{1}{C}\mathbf{w}^T\mathbf{w} + \frac{1}{M}\sum_{i=1}^{M}\log\left(1 + \exp\left(-y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b)\right)\right)}_{\ell(\mathbf{w},b)}$$

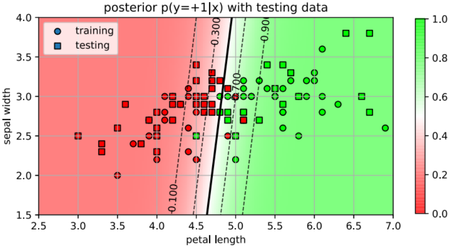
- **No closed-form solution** (i.e., no simple one-line equation to solve for $(\mathbf{w}, b)$)
  - Use an iterative optimization algorithm to find the optimal solution
  - E.g., gradient descent - step downhill in each iteration
    - $\mathbf{w} \leftarrow \mathbf{w} - \eta\frac{\partial \ell}{\partial \mathbf{w}}$
    - ℓ is the objective function
    - η is the **learning rate** (how far to step in each iteration)

Example: Iris Data

- Equation:

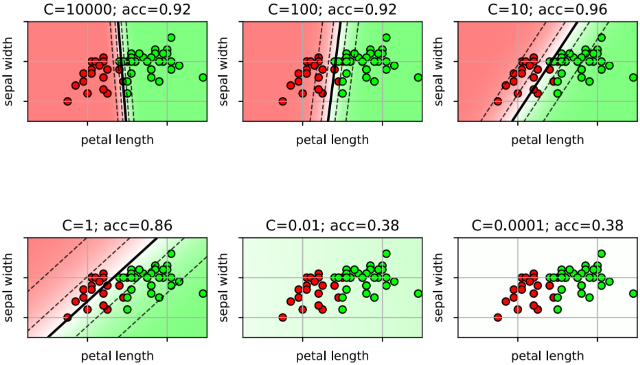
$$f(\mathbf{x}) = 4.87 \times \text{petal\_length} - 0.62 \times \text{sepal\_width} - 21.68$$

- Interpretation:
  - Large petal length makes $f(\mathbf{x})$ positive, so large petal length is associated with Class $+1$



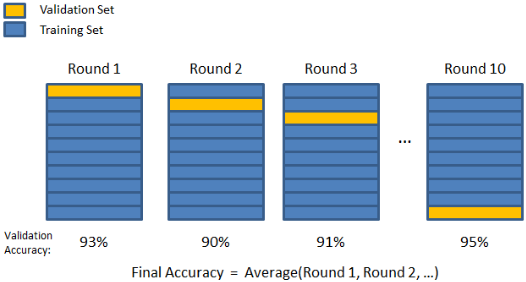
Selecting the Regularization Hyperparameter

- The regularization hyperparameter C has a big effect on the decision boundary and the accuracy
- How to select the value of C?



Cross-Validation

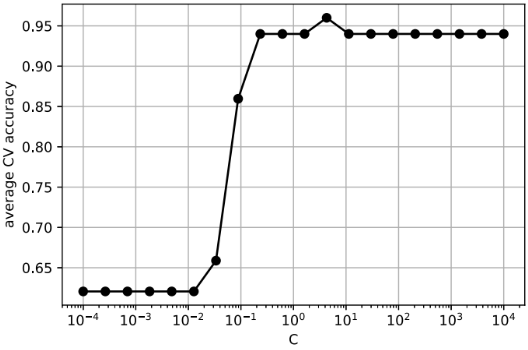
- Use **cross-validation** on **training** set to select the best value of C
- Run many experiments on the training set to see which parameters work on different versions of the data
  - Partition the data into folds of training and validation data
  - Try a range of C values on each fold (as validation set)
  - Pick the value that works best over all folds



Cross-Validation

- Procedure
  1. Select a range of C values to try
  2. Repeat K rounds
     1. Split the training set into training data and validation data
     2. Learn a classifier for each value of C
     3. Record the accuracy on the validation data for each C
  3. Select the value of that has the highest average accuracy over all K rounds
  4. Retrain the classifier using all data and the selected C

Which C to Select?

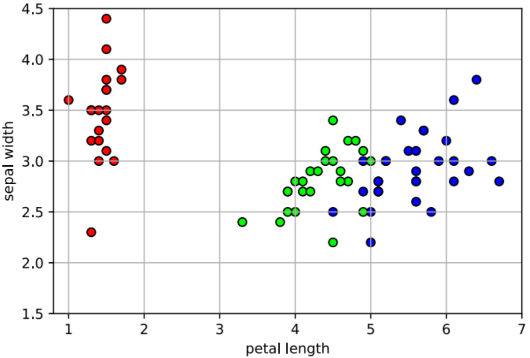


Multiclass Classification

- So far, we have only learned a classifier for 2 classes $\{-1, 1\}$
  - Called a **binary classifier**
- For more than 2 classes, split the problem into several binary classifier problems
- **One-vs-rest**
  - *Training*: for each class, train a classifier for that class versus the other classes
    - For example, if there are 3 classes, then train 3 binary classifiers: 1 vs $\{2, 3\}$; 2 vs $\{1, 3\}$; 3 vs $\{1, 2\}$
  - *Prediction*: calculate probability for each binary classifier. Select the class with the highest probability

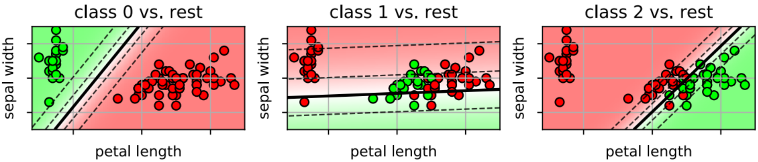
Example on 3-Class Iris Data

- $\mathcal{Y} = \{$"setosa", "versicolor", "virginica"$\}$

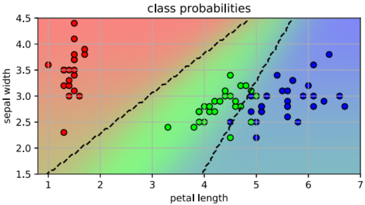


Example on 3-Class Iris Data

- The individual 1-vs-rest binary classifiers



- The final classifier, combining all 1-vs-rest classifiers



Multiclass Logistic Regression

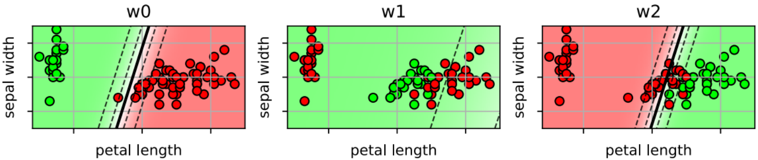
- Another way to get a multi-class classifier is to define a multi-class objective
  - One weight vector $\mathbf{w}_c$ for each Class c
    - We omit the bias b_c for each class for notation simplicity
- Define probabilities with **softmax** function
  - Analogous to sigmoid function for binary logistic regression

$$p(y = c|\mathbf{x}) = \frac{\exp(\mathbf{w}_c^T\mathbf{x})}{\exp(\mathbf{w}_1^T\mathbf{x}) + \cdots + \exp(\mathbf{w}_C^T\mathbf{x})}$$

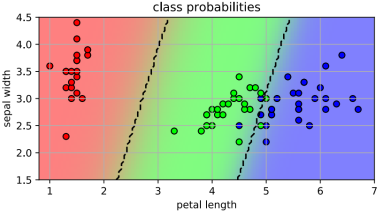
  - The class with the largest response of $\mathbf{w}_c^T\mathbf{x}$ will have the highest probability
- Estimate the $\{\mathbf{w}_c\}_{c=1}^C$ parameters using MLE as before

Multiclass Logistic Regression on 3-Class Iris Data

- The "binary" classifiers based on individual weight vectors



- Individual weight vectors work together to partition the space



Geometry

- Logistic regression is explicitly designed to have a linear decision boundary in the binary case
- In the multiclass case, the decision boundary is piece-wise linear

Logistic Regression Versus NB and LDA

- Speed: Learning logistic regression requires iterative numerical optimization, which will be slower than NB and LDA
- Storage: The model requires $O(N)$ parameters, the same order as NB, but much less than LDA's $O(N^2)$
- Interpretability: The "importance" of feature x_j can be understood in terms of the corresponding learned weight w_j

Logistic Regression Summary

- Classifier:
  - Linear function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$
  - Given a feature vector $\mathbf{x}$, the probability of a class is
    - $p(y=+1|\mathbf{x}) = \sigma(f(\mathbf{x})), p(y=-1|\mathbf{x}) = 1 - \sigma(f(\mathbf{x}))$
    - Sigmoid function: $\sigma(z) = \frac{1}{1+e^{-z}}$
  - Logistic loss function: $L(z) = \log(1 + \exp(-z))$
- Training:
  - Maximize the likelihood of the training data
  - Use regularization to prevent overfitting
    - Use cross-validation to pick the regularization hyperparameter C
- Classification:
  - Given a new sample $\mathbf{x}^*$, pick class with the highest $p(y|\mathbf{x}^*)$

$$y^* = \begin{cases} +1, & \frac{p(y=+1|\mathbf{x}^*)}{p(y=-1|\mathbf{x}^*)} > 1 \\ -1, & \text{otherwise} \end{cases} \text{ or } y^* = \begin{cases} +1, & f(\mathbf{x}^*) > 0 \\ -1, & \text{otherwise} \end{cases} = \text{sign}(f(\mathbf{x}^*))$$