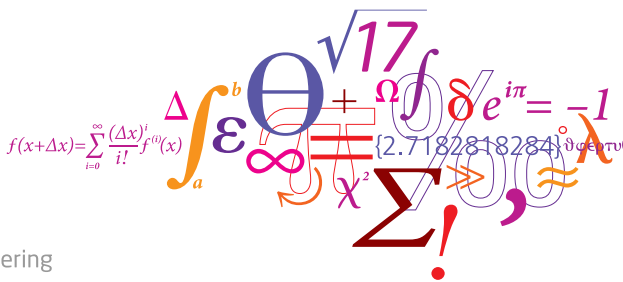
Lecture 3 - Classification



DTU Management Engineering
Department of Management Engineering

Outline

- Recap
- Introduction
- Models
  - Logistic Regression
  - Support Vector Machines

Scaling and Linear Regression

- Multivariate linear regression model (linear in the parameters β_i)

$$\boxed{Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_r X_r + \varepsilon}$$

Scaling and Linear Regression

- Multivariate linear regression model (linear in the parameters β_i)

$$\boxed{Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_r X_r + \varepsilon}$$

- Standard least squares coefficient estimates are *scale equivariant*.

Scaling and Linear Regression

- Multivariate linear regression model (linear in the parameters β_i)

$$\boxed{Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_r X_r + \varepsilon}$$

- Standard least squares coefficient estimates are *scale equivariant*.

- Multiplying X_j by a constant c simply leads to a scaling of the least squares coefficient estimate $\hat{\beta}_j$ by a factor of $1/c$.

Scaling and Linear Regression

- Multivariate linear regression model (linear in the parameters β_i)

$$\boxed{Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_r X_r + \varepsilon}$$

- Standard least squares coefficient estimates are *scale equivariant*.

- Multiplying X_j by a constant c simply leads to a scaling of the least squares coefficient estimate $\hat{\beta}_j$ by a factor of $1/c$.

- In other words, regardless of how the jth predictor is scaled, $X_j\hat{\beta}_j$ will remain the same
- However, this is not true with Ridge regression and the Lasso !

Back to Ridge Regression and the Lasso

- Recall the objective function in Ridge regression .

$$S = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2$$

Back to Ridge Regression and the Lasso

- Recall the objective function in Ridge regression .

$$S = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2$$

- In this case, $X_j\hat{\beta}_{j,\lambda}$ depends not only on the value of λ, but also on the scaling of X_j and of the other predictors!

Back to Ridge Regression and the Lasso

- Recall the objective function in Ridge regression .

$$S = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2$$

- In this case, $X_j\hat{\beta}_{j,\lambda}$ depends not only on the value of λ, but also on the scaling of X_j and of the other predictors!
- Hence, it is important to *standardize* the predictors before applying ridge regression or the Lasso.

Standardization and Normalization (feature scaling)

- Standardization and Normalization are two common ways of scaling features

Standardization and Normalization (feature scaling)

- Standardization and Normalization are two common ways of scaling features
- Standardization transforms a variable so it has a zero mean and unit standard deviation

$$\tilde{x}_{ij} = \frac{x_{ij} - \bar{x_j}}{\sigma_{x_j}} = \frac{x_{ij} - \bar{x_j}}{\sqrt{\frac{\sum_{i=1}^{n}(x_{ij} - \bar{x_j})^2}{n}}}$$

Standardization and Normalization (feature scaling)

- Standardization and Normalization are two common ways of scaling features
- Standardization transforms a variable so it has a zero mean and unit standard deviation

$$\tilde{x}_{ij} = \frac{x_{ij} - \bar{x_j}}{\sigma_{x_j}} = \frac{x_{ij} - \bar{x_j}}{\sqrt{\frac{\sum_{i=1}^{n}(x_{ij} - \bar{x_j})^2}{n}}}$$

- Normalization transforms a variable so it is between 0 and 1

$$\tilde{x}_{ij} = \frac{x_{ij} - x_j^{\text{MIN}}}{x_j^{\text{MAX}} - x_j^{\text{MIN}}}$$

Handling Categorical Predictors

- Suppose we are modeling credit card balance of a customer and we have information about whether a customer owns a house or not

Handling Categorical Predictors

- Suppose we are modeling credit card balance of a customer and we have information about whether a customer owns a house or not
- We can incorporate this variable in the model by creating a indicator or dummy variable

$$x_i = \begin{cases} 1 & \text{if person } i \text{ owns a house.} \\ 0 & \text{if person } i \text{ does not own a house.} \end{cases}$$

Handling Categorical Predictors

- Suppose we are modeling credit card balance of a customer and we have information about whether a customer owns a house or not
- We can incorporate this variable in the model by creating a indicator or dummy variable

$$x_i = \begin{cases} 1 & \text{if person } i \text{ owns a house.} \\ 0 & \text{if person } i \text{ does not own a house.} \end{cases}$$

- If we have a qualitative variable with k possible values, we create $k-1$ dummy variables. For example if we know the location of the house (say South, West or East), we could create two variables

$$x_{i1} = \begin{cases} 1 & \text{if person } i \text{ lives in the South.} \\ 0 & \text{if person } i \text{ does not live in the South.} \end{cases}$$

$$x_{i2} = \begin{cases} 1 & \text{if person } i \text{ lives in the West.} \\ 0 & \text{if person } i \text{ does not live in the West.} \end{cases}$$

Introduction

- Linear regression $\rightarrow$ Predicting a **continuous** dependent variable from a set of independent variables.
- Classification $\rightarrow$ Predicting a **discrete** dependent variable from a set of independent variables.
- Identifying to which of a set of classes, $\{\pi_1, ..., \pi_g\}$, a new observation belongs, on the basis of a training set of data containing observations whose classes are known.

Classification

Examples:

- Vehicle identification - determine type of vehicle from camera
- Incident detection - determine incident/no-incident from flow and speed data
- Spam filter - Allocate emails to spam/no-spam
- Handwritten Digit Recognition - Allocate the images of handwritten zip codes on mail to digits

Example: Vehicle Type Identification

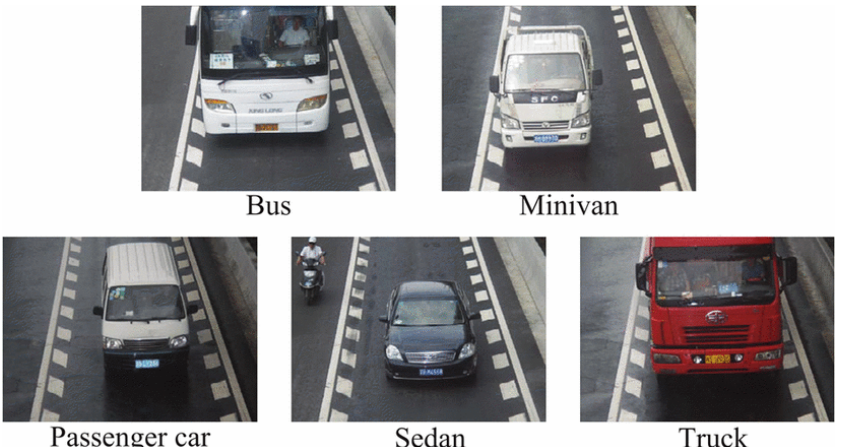


Image source: Dong, Zhen et al. "Vehicle Type Classification Using Unsupervised Convolutional Neural Network." IEEE, 2014. 172–177. Print.

Example: Vehicle Type Identification

- Images were captured from traffic cameras
- The task is to identify the types of vehicles in these images
- Each image represents one observation
- Dependent variable $Y \in \{Bus, MiniVan, PassengerCar, Sedan, Truck\}$
- Applications: traffic volume estimation, illegal vehical type identification

Regression for Discrete Dependent Variable?

- Can we directly use linear regression models to predict the discrete dependent variable?
- Linear regression model:

$$\mathbf{Y} = \mathbf{X}\beta + \boldsymbol{\epsilon}$$

- We used least squares method to estimate the parameters β

Regression for Discrete Dependent Variable?

- Can we directly use linear regression models to predict the discrete dependent variable?
- Linear regression model:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

- We used least squares method to estimate the parameters $\boldsymbol{\beta}$
- This model predicts continuous variable with values ranging from $-\infty$ to ∞ i.e, $Y \in (-\infty, \infty)$
- We need another way to use a 'regression like' approach for discrete dependent variables.

Binary Logistic Regression

- Dependent variable, Y, is restricted to two values.
- Examples: $\{\text{Incident}, \text{NoIncident}\}$, $\{\text{Spam}, \text{NotSpam}\}$, $\{\text{Employed}, \text{NotEmployed}\}$
- Denote the two cases as 0 and 1

Binary Logistic Regression

- Dependent variable, Y, is restricted to two values.
- Examples: $\{\text{Incident}, \text{NoIncident}\}$, $\{\text{Spam}, \text{NotSpam}\}$, $\{\text{Employed}, \text{NotEmployed}\}$
- Denote the two cases as 0 and 1
- For a given observation, we want to estimate the **probability** that Y belongs to either case.

Binary Dependent Variables

- We need only focus on predicting $p(y_i = 1)$

  $->$ $p(y_i = 0) = 1 - p(y_i = 1)$

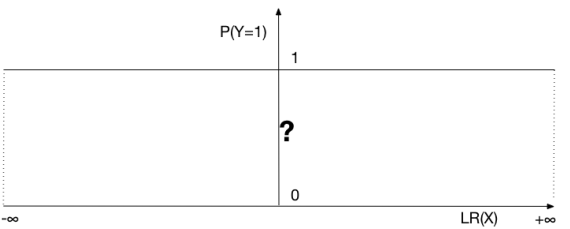
Binary Dependent Variables

- We need only focus on predicting $p(y_i = 1)$

  $->$ $p(y_i = 0) = 1 - p(y_i = 1)$

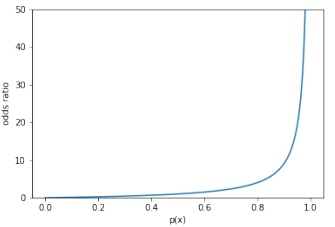
- Linear regression model still cannot be used since the probability $p(y_i = 1), \in [0, 1]$

- Need to establish a function that links $p(y_i = 1)$ to a continuous variable that ranges from $-\infty$ to $+\infty$, possibly obtained by a linear model (LR(X)).



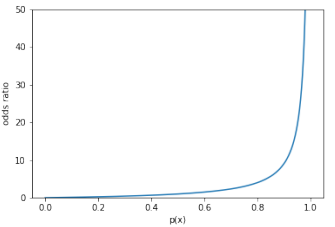
Odds ratio

$$odds = \frac{p(y_i = 1)}{1 - p(y_i = 1)}$$



Odds ratio

$$odds = \frac{p(y_i = 1)}{1 - p(y_i = 1)}$$



- not symmetric
- $[0, +\infty[$, but it should be $]-\infty, +\infty[$

Logit model

Take the natural logarithm of the odds ratio

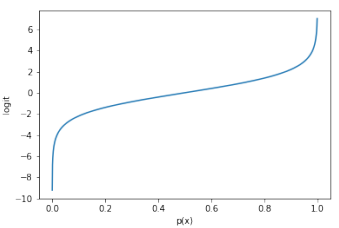
$$\text{logit}(p(y_i = 1)) = \ln(odds) = \ln\left(\frac{p(y_i = 1)}{1 - p(y_i = 1)}\right)$$

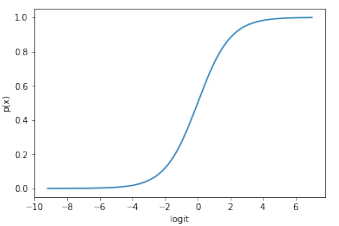
Logit model

Take the natural logarithm of the odds ratio

$$\text{logit}(p(y_i = 1)) = \ln(odds) = \ln\left(\frac{p(y_i = 1)}{1 - p(y_i = 1)}\right)$$

Log odds ratio (*logit*) provides a monotonically increasing function of $p(y_i = 1)$ with range $(-\infty, \infty)$





Binary Logit Model

Now we can use a linear function of the predictors to model the log odds ratio (*logit*)

$$\text{logit}(p(y_i = 1)) = \ln\left(\frac{p(y_i = 1)}{1 - p(y_i = 1)}\right) = \boldsymbol{\beta}^T \mathbf{x}_i$$

Binary Logit Model

Now we can use a linear function of the predictors to model the log odds ratio (*logit*)

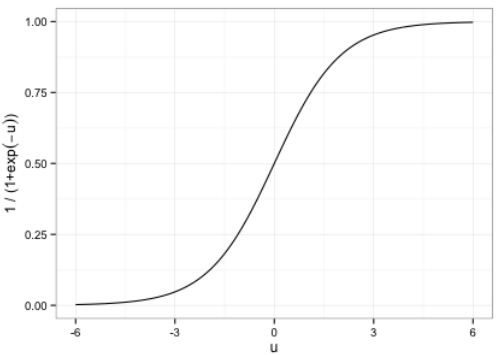
$$\text{logit}(p(y_i = 1)) = \ln\left(\frac{p(y_i = 1)}{1 - p(y_i = 1)}\right) = \boldsymbol{\beta}^T \mathbf{x}_i$$

From the predicted value of the *logit*, we can obtain the probability of $y_i = 1$ through the *logistic sigmoid function*:

$$p(y_i = 1) = \frac{1}{1 + \exp(-\boldsymbol{\beta}^T \mathbf{x_i})}$$

Logistic Sigmoid Function

$$p(y_i = 1) = \frac{1}{1 + \exp(-\boldsymbol{\beta}^T \mathbf{x_i})}$$



Parameter Interpretation

- β_0 is the intercept
- $\beta_1, \beta_2, ..., \beta_r$ are the coefficients
- The sign of β_i indicates the direction of change in $p(y)$ with respect to change in x_i

Parameter Interpretation

- β_0 is the intercept
- $\beta_1, \beta_2, ..., \beta_r$ are the coefficients
- The sign of β_i indicates the direction of change in $p(y)$ with respect to change in x_i
- The magnitude of β_i affects the steepness of the sigmoid function

Model Estimation

- We need to estimate β

Model Estimation

- We need to estimate $\boldsymbol{\beta}$
- We want to choose parameters $\boldsymbol{\beta} = [\beta_0, \beta_1, ..., \beta_r]^T$ such that the joint probability of $P(\boldsymbol{Y} = \mathbf{y})$ for all observations is maximized

Model Estimation

- We need to estimate $\boldsymbol{\beta}$
- We want to choose parameters $\boldsymbol{\beta} = [\beta_0, \beta_1, ..., \beta_r]^T$ such that the joint probability of $P(\boldsymbol{Y} = \mathbf{y})$ for all observations is maximized
- Example
  - Obs 1: $\mathbf{x_1}, y_1 = 1$; $p(y_1 = 1) = p_1 = \dfrac{1}{1 + \exp(-\boldsymbol{\beta}^T \mathbf{x_1})}$
  - Obs 2: $\mathbf{x_2}, y_2 = 0$; $p(y_2 = 0) = p_2 = 1 - \left(\dfrac{1}{1 + \exp(-\boldsymbol{\beta}^T \mathbf{x_2})} \right)$
  - $L(\boldsymbol{\beta}) = p_1 \times p_2$

Model Estimation

- We need to estimate $\boldsymbol{\beta}$
- We want to choose parameters $\boldsymbol{\beta} = [\beta_0, \beta_1, ..., \beta_r]^T$ such that the joint probability of $P(\boldsymbol{Y} = \mathbf{y})$ for all observations is maximized
- Example
  - Obs 1: $\mathbf{x_1}, y_1 = 1$; $p(y_1 = 1) = p_1 = \dfrac{1}{1 + \exp(-\boldsymbol{\beta}^T \mathbf{x_1})}$
  - Obs 2: $\mathbf{x_2}, y_2 = 0$; $p(y_2 = 0) = p_2 = 1 - \left(\dfrac{1}{1 + \exp(-\boldsymbol{\beta}^T \mathbf{x_2})} \right)$
  - $L(\boldsymbol{\beta}) = p_1 \times p_2$
- The function of parameters $L(\boldsymbol{\beta})$ that we want to maximize is called the *Likelihood Function*

Maximum Likelihood

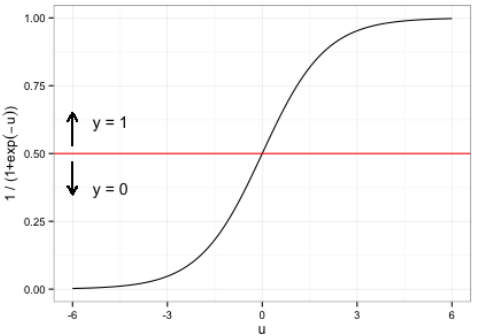
- Typically to maximize the log likelihood we differentiate the function with respect to the parameters and set it equal to zero
- In case of the logistic regression there is no closed form solution due to the non-linearity introduced by exponentials inside the sum
- We need to use numerical methods for finding the parameter values that maximize the log likelihood

Prediction

For a test observation or point i, once we obtain the probability of $y_i = 1$, we assign a prediction value such that

$$\begin{cases} y_i = 1 & \text{if } \ p(y_i) > 0.5 \\ y_i = 0 & \text{otherwise} \end{cases}$$

Logistic Regression: Decision Boundary



Evaluating Classifiers

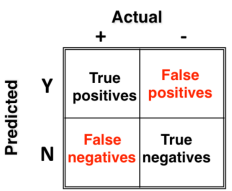
- Fill a *confusion matrix* (also known as *success matrix*) contrasting the actual classification with the predicted on the training set

		Actual Class	
		Dog	Cat
Predicted Class	Dog	88	12
	Cat	10	90

Evaluating Classifiers

- Fill a *confusion matrix* (also known as *success matrix*) contrasting the actual classification with the predicted on the training set

		Actual Class	
		Dog	Cat
Predicted Class	Dog	88	12
	Cat	10	90



From https://www.svds.com/the-basics-of-classifier-evaluation-part-1/

Evaluating Classifiers

- Accuracy

$$\frac{\#\text{Correct predictions}}{\#\text{Total predictions}}$$

Evaluating Classifiers

- Accuracy

$$\frac{\#\text{Correct predictions}}{\#\text{Total predictions}}$$

- true positive rate $= \frac{TP}{TP+FN}$
- False positive rate $= \frac{FP}{FP+TN}$

Evaluating Classifiers

- Accuracy

$$\frac{\#\text{Correct predictions}}{\#\text{Total predictions}}$$

- true positive rate $= \frac{TP}{TP+FN}$
- False positive rate $= \frac{FP}{FP+TN}$
- Precision=Positive predictive value $= \frac{TP}{TP+FP}$
- Recall = Sensitivity = true positive rate $= \frac{TP}{TP+FN}$

Evaluating Classifiers

- Accuracy

$$\frac{\#\text{Correct predictions}}{\#\text{Total predictions}}$$

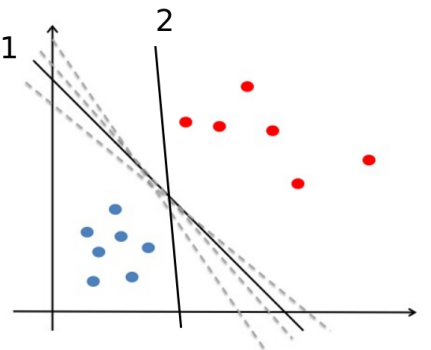
- true positive rate $= \frac{TP}{TP+FN}$
- False positive rate $= \frac{FP}{FP+TN}$
- Precision=Positive predictive value $= \frac{TP}{TP+FP}$
- Recall = Sensitivity = true positive rate $= \frac{TP}{TP+FN}$
- F1 is the harmonic mean of precision and recall:

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

Playtime!

- Open the "5. Classification.ipynb" notebook
- Do Part 1
- Estimated duration: 30 min

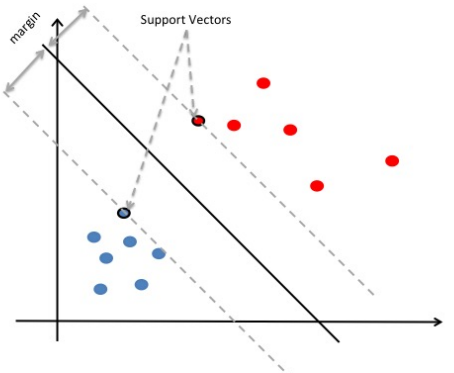
- Find a line that separates the observations belonging to different classes.
- Would you choose line 1 or line 2? Why?



- **note**: for mathematical convenience, y will now be {-1, 1} (instead of {0, 1} from before)

Support Vector Machines - Support Vectors

Goal in SVM: maximize the separation margin.



Support Vector Machines - Formalization

A hyperplane is defined by parameters $\beta_0, \boldsymbol{\beta}$ such that

$$\beta_0 + \boldsymbol{\beta}^T \mathbf{x} \begin{cases} = 0 & \text{if } \mathbf{x} \text{ lies on the plane} \\ < 0 & \text{if } \mathbf{x} \text{ on one side} \\ > 0 & \text{if } \mathbf{x} \text{ on the other side} \end{cases}$$

Support Vector Machines - Formalization

A hyperplane is defined by parameters $\beta_0, \boldsymbol{\beta}$ such that

$$\beta_0 + \boldsymbol{\beta}^T \mathbf{x} \begin{cases} = 0 & \text{if } \mathbf{x} \text{ lies on the plane} \\ < 0 & \text{if } \mathbf{x} \text{ on one side} \\ > 0 & \text{if } \mathbf{x} \text{ on the other side} \end{cases}$$

Rule: $\text{sign}(\beta_0 + \boldsymbol{\beta}^T \mathbf{x})$ tells us the class.

Support Vector Machines - Formalization

A hyperplane is defined by parameters $\beta_0, \boldsymbol{\beta}$ such that

$$\beta_0 + \boldsymbol{\beta}^T \mathbf{x} \begin{cases} = 0 & \text{if } \mathbf{x} \text{ lies on the plane} \\ < 0 & \text{if } \mathbf{x} \text{ on one side} \\ > 0 & \text{if } \mathbf{x} \text{ on the other side} \end{cases}$$

Rule: $\text{sign}(\beta_0 + \boldsymbol{\beta}^T \mathbf{x})$ tells us the class.

The *margin-maximizer* hyperplane $\beta_0, \boldsymbol{\beta}$ comes from

$$\max_{\beta_0, \boldsymbol{\beta}} M \text{ s.t.}$$
$$y_j(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_j) \geq M, \forall j$$
$$||\boldsymbol{\beta}||^2 = 1$$

Support Vector Machines - Formalization

A hyperplane is defined by parameters $\beta_0, \boldsymbol{\beta}$ such that

$$\beta_0 + \boldsymbol{\beta}^T \mathbf{x} \begin{cases} = 0 & \text{if } \mathbf{x} \text{ lies on the plane} \\ < 0 & \text{if } \mathbf{x} \text{ on one side} \\ > 0 & \text{if } \mathbf{x} \text{ on the other side} \end{cases}$$

Rule: $\text{sign}(\beta_0 + \boldsymbol{\beta}^T \mathbf{x})$ tells us the class.

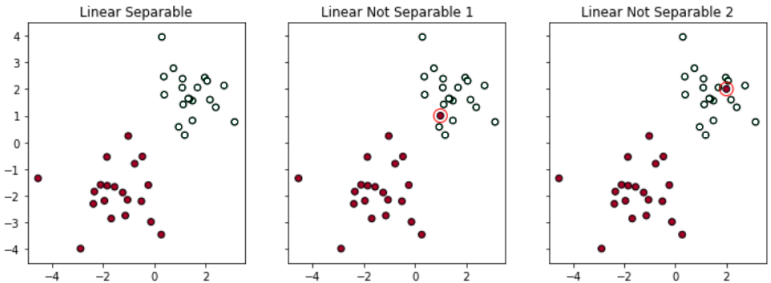
The *margin-maximizer* hyperplane $\beta_0, \boldsymbol{\beta}$ comes from

$$\max_{\beta_0, \boldsymbol{\beta}} M \text{ s.t.}$$
$$y_j(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_j) \geq M, \forall j$$
$$||\boldsymbol{\beta}||^2 = 1$$

After calculation, we find that $\boldsymbol{\beta} = \sum_{j \in \mathcal{S}} \alpha_j y_j \mathbf{x}_j$.
The plane only depends on the support vectors $\mathcal{S}$ (subset of the training set).
β_0 can be obtained by solving $\alpha_j \left[y_j(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}_j) - 1 \right] = 0$ for any of the support vectors $\mathbf{x}_j, j \in \mathcal{S}$

Support Vector Machines - Hard vs Soft margin

What if the data is not linearly separable?



- Reformulate optimization problem to "tolerate" a few dots getting misclassified
- Tolerance parameter C controls the amount of misclassifications that are allowed

Support Vector Machines - Hard vs Soft margin

Tolerance parameter C tries to balance the trade-off between finding a line that maximizes the margin and minimizes the misclassification

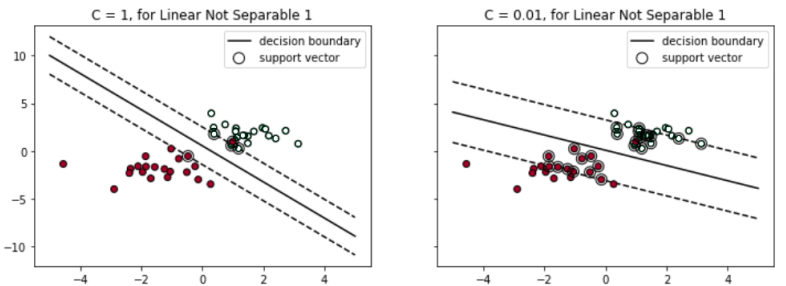
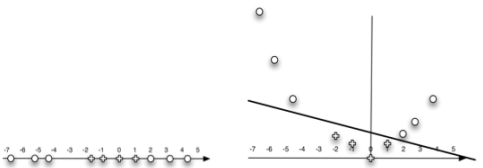


Image credit: `https://medium.com/bite-sized-machine-learning/support-vector-machine-explained-soft-margin-kernel-tricks-3728dfb92cee`

Support Vector Machines - Kernel trick

What if the data is not linearly separable?



- We can find a mapping $\boldsymbol{\phi}(x)$ into a higher dimension where all objects are linearly separable, in this case $\boldsymbol{\phi}(x) = (x, x^2)$

Support Vector Machines - Kernel functions

- Recall the equation of the separating hyperplane in the support vector classifier:

$$f(\mathbf{x}_i) = \beta_0 + \sum_{j \in \mathcal{S}} \alpha_j y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

where $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \sum_{k=1}^{p} x_{ik} x_{jk}$

Support Vector Machines - Kernel functions

- Recall the equation of the separating hyperplane in the support vector classifier:

$$f(\mathbf{x}_i) = \beta_0 + \sum_{j \in \mathcal{S}} \alpha_j y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

where $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \sum_{k=1}^{p} x_{ik} x_{jk}$

- When we enlarge the feature space, we have

$$f(\mathbf{x}_i) = \beta_0 + \sum_{j \in \mathcal{S}} \alpha_j y_j \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$$

Support Vector Machines - Kernel functions

- Recall the equation of the separating hyperplane in the support vector classifier:

$$f(\mathbf{x}_i) = \beta_0 + \sum_{j \in \mathcal{S}} \alpha_j y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

  where $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \sum_{k=1}^{p} x_{ik} x_{jk}$

- When we enlarge the feature space, we have

$$f(\mathbf{x}_i) = \beta_0 + \sum_{j \in \mathcal{S}} \alpha_j y_j \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$$

- The kernel function K computes these inner products in the transformed space

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$$

Support Vector Machines - Kernel functions

- Recall the equation of the separating hyperplane in the support vector classifier:

$$f(\mathbf{x}_i) = \beta_0 + \sum_{j \in \mathcal{S}} \alpha_j y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

where $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \sum_{k=1}^{p} x_{ik} x_{jk}$

- When we enlarge the feature space, we have

$$f(\mathbf{x}_i) = \beta_0 + \sum_{j \in \mathcal{S}} \alpha_j y_j \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$$

- The kernel function K computes these inner products in the transformed space

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$$

$$f(\mathbf{x}_i) = \beta_0 + \sum_{j \in \mathcal{S}} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

Support Vector Machines - Kernel functions

- Recall the equation of the separating hyperplane in the support vector classifier:

$$f(\mathbf{x}_i) = \beta_0 + \sum_{j \in \mathcal{S}} \alpha_j y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

where $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \sum_{k=1}^{p} x_{ik} x_{jk}$

- When we enlarge the feature space, we have

$$f(\mathbf{x}_i) = \beta_0 + \sum_{j \in \mathcal{S}} \alpha_j y_j \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$$

- The kernel function K computes these inner products in the transformed space

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$$

$$f(\mathbf{x}_i) = \beta_0 + \sum_{j \in \mathcal{S}} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

- If we have K we do not need to know or compute $\boldsymbol{\phi}(\mathbf{x})$ at all!

Support Vector Machines - Kernel functions (contd.)

- A simple example: consider the case with two features X_1 and X_2, and two observations $\mathbf{x}_i = (x_{i1}, x_{i2})$ and $\mathbf{x}_j = (x_{j1}, x_{j2})$

Support Vector Machines - Kernel functions (contd.)

- A simple example: consider the case with two features X_1 and X_2, and two observations $\mathbf{x}_i = (x_{i1}, x_{i2})$ and $\mathbf{x}_j = (x_{j1}, x_{j2})$

$$K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \langle \mathbf{x}_i, \mathbf{x}_j \rangle)^2$$

$$= (1 + x_{i1}x_{j1} + x_{i2}x_{j2})^2$$

$$= \left(1 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + (x_{i1}x_{j1})^2 + (x_{i2}x_{j2})^2 + 2x_{i1}x_{j1}x_{i2}x_{j2}\right)$$

Support Vector Machines - Kernel functions (contd.)

- A simple example: consider the case with two features X_1 and X_2, and two observations $\mathbf{x}_i = (x_{i1}, x_{i2})$ and $\mathbf{x}_j = (x_{j1}, x_{j2})$

$$K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \langle \mathbf{x}_i, \mathbf{x}_j \rangle)^2$$

$$= (1 + x_{i1}x_{j1} + x_{i2}x_{j2})^2$$

$$= \left(1 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + (x_{i1}x_{j1})^2 + (x_{i2}x_{j2})^2 + 2x_{i1}x_{j1}x_{i2}x_{j2}\right)$$

- If we were to define our enlarged feature space as

$$\boldsymbol{\phi}(\mathbf{x}_i) = (1, \sqrt{2}x_{i1}, \sqrt{2}x_{i2}, x_{i1}^2, x_{i2}^2, \sqrt{2}x_{i1}x_{i2})$$

we see that $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$!

Support Vector Machines - Kernel functions (contd.)

- *Linear* kernel: original support vector classifier

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

Support Vector Machines - Kernel functions (contd.)

- *Linear* kernel: original support vector classifier

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

- *Polynomial* kernel of degree d: Generates new features through polynomial combinations of existing features (degree d)

$$K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \langle \mathbf{x}_i, \mathbf{x}_j \rangle)^d$$

Support Vector Machines - Kernel functions (contd.)

- *Linear* kernel: original support vector classifier

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

- *Polynomial* kernel of degree d: Generates new features through polynomial combinations of existing features (degree d)

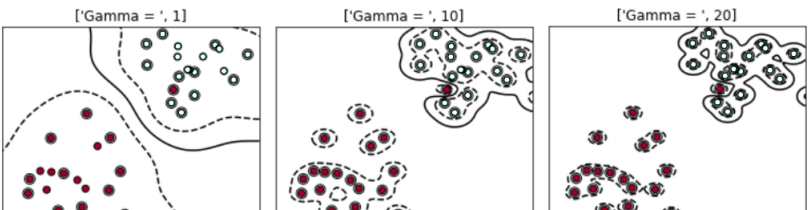
$$K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \langle \mathbf{x}_i, \mathbf{x}_j \rangle)^d$$

- *Gaussian Radial basis function* kernel: value depends on the distance between the two points point

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2) = \exp(-\gamma \sum_{k=1}^{p} (x_{ik} - x_{jk})^2)$$

Support Vector Machines - RBF kernel

- Parameter γ (gamma) controls how much nearby points influence each other
- In practice, controls how "wiggling" the decision boundary is



Support Vector Machines - Conclusion

- SVMs learn the function of class separation as in logistic regression.
- The objective function for finding the best separation of classes is based on maximizing the separation margin.
- SVMs do not model the distribution of the data rather only find the support vectors.

Playtime!

- Open the "5. Classification.ipynb" notebook
- Do Part 2
- Estimated duration: 15 min