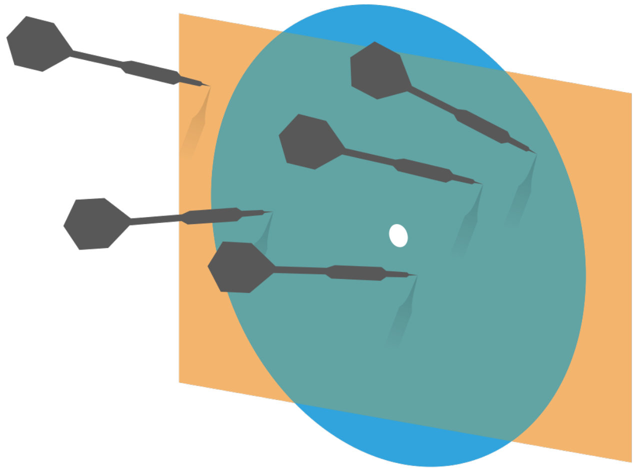
Probability and Statistics for Computer Science



"…many problems are naturally classification problems"---Prof. Forsyth

Credit: wikipedia

Hongye Liu, Teaching Assistant Prof, CS361, UIUC, 11.12.2020

Last time

- Decision tree (II)
- Random forest
- Support Vector Machine (I)

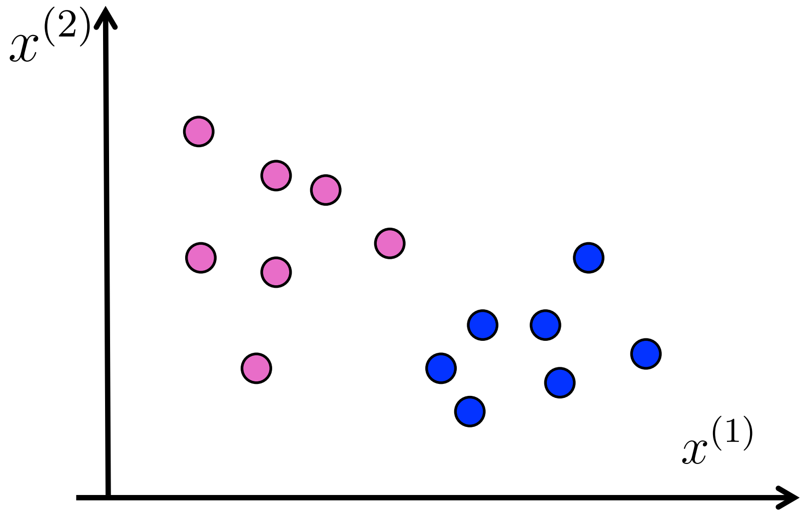
Objectives

Motivation for Studying Support Vector Machine

- When solving a classification problem, it is good to try several techniques.
- Criteria to consider in choosing the classifier include
  - Accuracy ✓
  - Training speed
  - Classification speed ✓
  - Performance with small training set
  - Interpretability ✓

SVM problem formulation

- At first we assume a binary classification problem
- The training set consists of N items
  - Feature vectors x_i of dimension d
  - Corresponding class labels $y_i \in \{\pm 1\}$
- We can picture the training data as a d-dimensional scatter plot with colored labels



Decision boundary of SVM

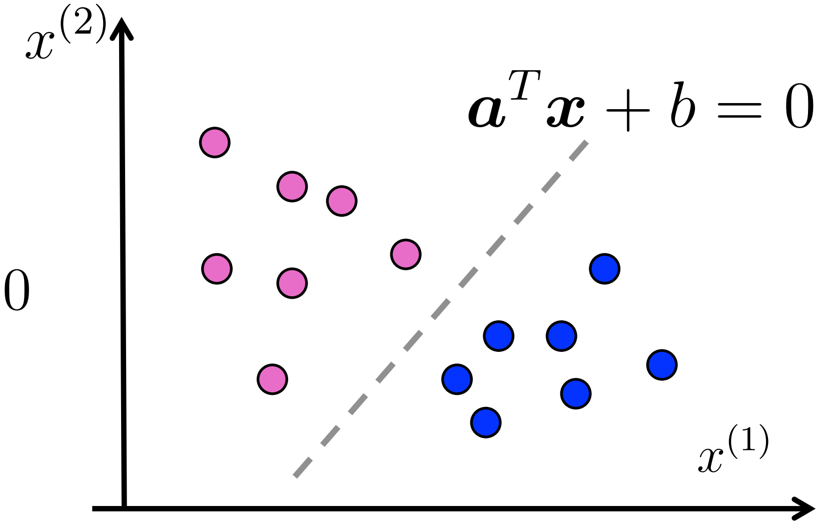
- SVM uses a hyperplane as its **decision boundary**

- The decision boundary is:

$$a_1 x^{(1)} + a_2 x^{(2)} + \dots + a_d x^{(d)} + b = 0$$

- In vector notation, the hyperplane can be written as:

$$\boldsymbol{a}^T \boldsymbol{x} + b = 0$$

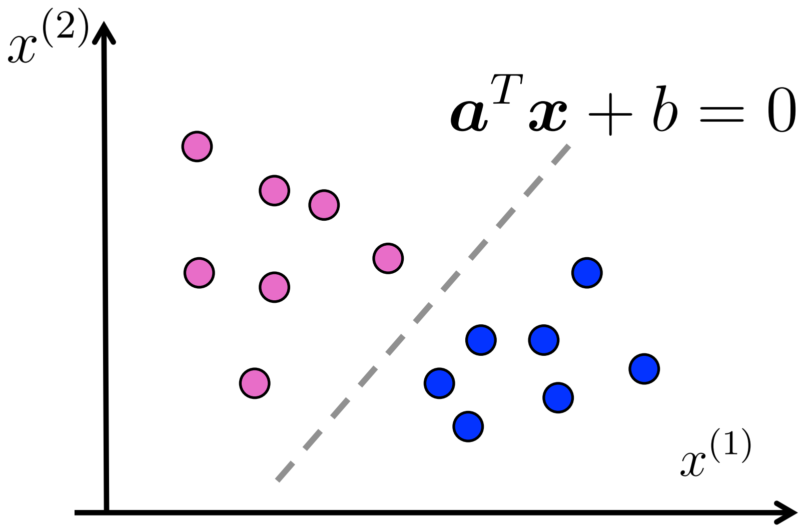


Classification function of SVM

- SVM assigns a class label to a feature vector according to the following rule:

  +1 if $\boldsymbol{a}^T \boldsymbol{x}_i + b \geq 0$

  -1 if $\boldsymbol{a}^T \boldsymbol{x}_i + b < 0$

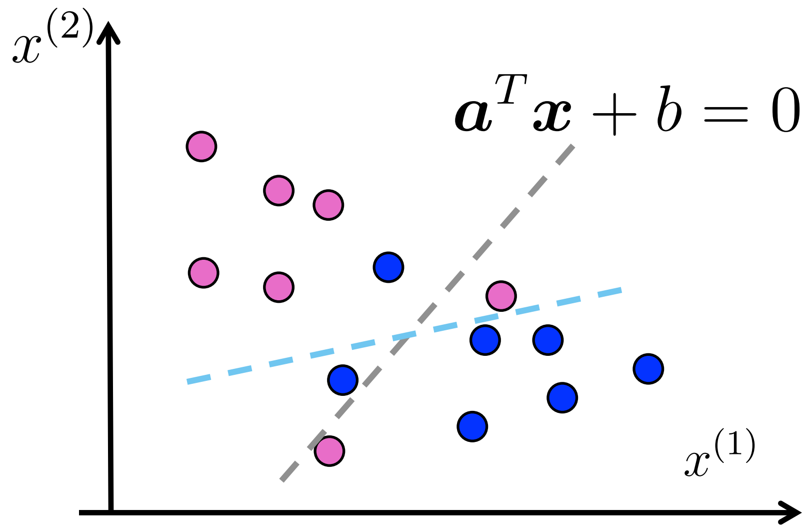


- In other words, the classification function is: $sign(\boldsymbol{a}^T \boldsymbol{x}_i + b)$

- Note that
  - If $|\boldsymbol{a}^T \boldsymbol{x}_i + b|$ is small, then $\boldsymbol{x}_i$ was close to the decision boundary
  - If $|\boldsymbol{a}^T \boldsymbol{x}_i + b|$ is large, then $\boldsymbol{x}_i$ was far from the decision boundary

What if there is no clean cut boundary?

- Some boundaries are better than others for the training data
- Some boundaries are likely more robust for run-time data
- We need to a quantitative measure to decide about the boundary
- The **loss function** can help decide if one boundary is better than others



Loss function 1

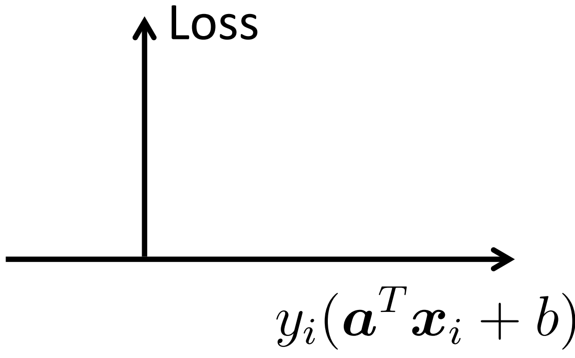
- For any given feature vector $\boldsymbol{x}_i$ with class label $y_i \in \{\pm 1\}$, we want
  - Zero loss if $\boldsymbol{x}_i$ is classified correctly $sign(\boldsymbol{a}^T \boldsymbol{x}_i + b) = y_i$
  - Positive loss if $\boldsymbol{x}_i$ is misclassified $sign(\boldsymbol{a}^T \boldsymbol{x}_i + b) \neq y_i$
  - If $\boldsymbol{x}_i$ is misclassified, more loss is assigned if it's further away from the boundary

- This loss function 1 meets the criteria above:

$$max(0, -y_i(\boldsymbol{a}^T \boldsymbol{x}_i + b))$$

- Training error cost

$$S(\boldsymbol{a}, b) = \frac{1}{N} \sum_{i=1}^{N} max(0, -y_i(\boldsymbol{a}^T \boldsymbol{x}_i + b))$$



Q. What's the value of this function ?

$$max(0, -y_i(\boldsymbol{a}^T\boldsymbol{x}_i + b)) \quad \text{if} \quad sign(\boldsymbol{a}^T\boldsymbol{x}_i + b) = y_i$$

[✓ A. 0.]
B. others.

Q. What’s the value of this function ?

$$max(0, -y_i(\boldsymbol{a}^T\boldsymbol{x}_i + b)) \quad \text{if} \quad sign(\boldsymbol{a}^T\boldsymbol{x}_i + b) \neq y_i$$

A. 0.

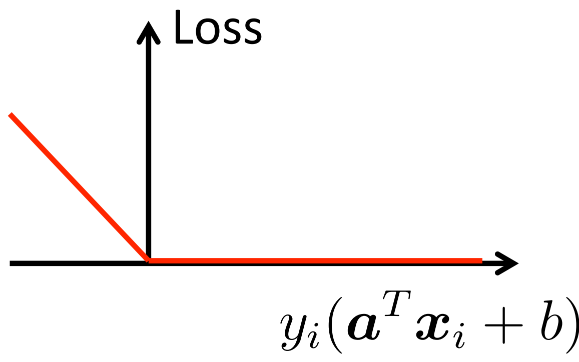
[✓ B. A value greater than or equal to 0.]

Loss function 1

- For any given feature vector $\boldsymbol{x}_i$ with class label $y_i \in \{\pm 1\}$, we want
  - Zero loss if $\boldsymbol{x}_i$ is classified correctly $sign(\boldsymbol{a}^T \boldsymbol{x}_i + b) = y_i$
  - Positive loss if $\boldsymbol{x}_i$ is misclassified $sign(\boldsymbol{a}^T \boldsymbol{x}_i + b) \neq y_i$
  - If $\boldsymbol{x}_i$ is misclassified, more loss is assigned if it's further away from the boundary

- This loss function 1 meets the criteria above:

$$max(0, -y_i(\boldsymbol{a}^T \boldsymbol{x}_i + b))$$



- Training error cost

$$S(\boldsymbol{a}, b) = \frac{1}{N} \sum_{i=1}^{N} max(0, -y_i(\boldsymbol{a}^T \boldsymbol{x}_i + b))$$

The problem with loss function 1

- Loss function1 does not distinguish between the following decision boundaries if they both classify $\boldsymbol{x}_i$ correctly.
  - One passes the two classes closely
  - One that passes with a wider margin

- But leaving a larger margin gives robustness for run-time data- **the large margin principle**

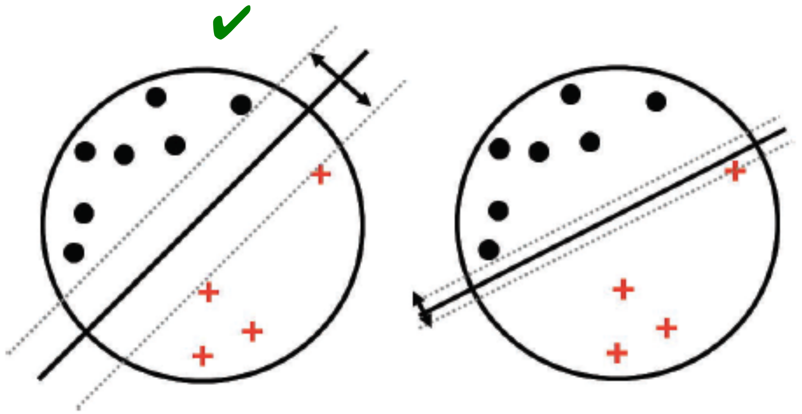


Figure 14.11 Illustration of the large margin principle. Left: a separating hyper-plane with large margin. Right: a separating hyper-plane with small margin.

Credit: Kelvin Murphy

Loss function 2: the hinge loss

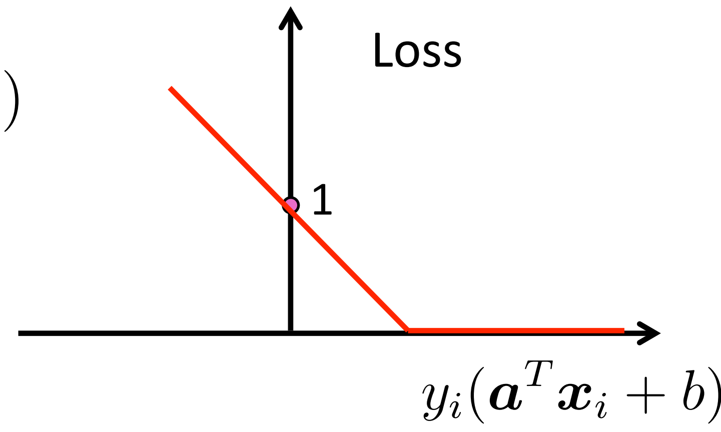
- We want to impose a small positive loss if $\boldsymbol{x}_i$ is correctly classified but close to the boundary

- The **hinge loss** function meets the criteria above:

$$max(0, 1 - y_i(\boldsymbol{a}^T\boldsymbol{x}_i + b))$$

- Training error cost

$$S(\boldsymbol{a}, b) = \frac{1}{N}\sum_{i=1}^{N} max(0, 1 - y_i(\boldsymbol{a}^T\boldsymbol{x}_i + b))$$



The problem with loss function 2

- Loss function 2 favors decision boundaries that have large $\|\boldsymbol{a}\|$ because increasing $\|\boldsymbol{a}\|$ can zero out the loss for a correctly classified $\boldsymbol{x}_i$ near the boundary.

- But large $\|\boldsymbol{a}\|$ makes the classification function $sign(\boldsymbol{a}^T\boldsymbol{x}_i + b)$ extremely sensitive to small changes in $\boldsymbol{x}_i$ and make it less robust to run-time data.

- So small $\|\boldsymbol{a}\|$ is better.

Hinge loss with regularization penalty

- We add a penalty on the square magnitude $\|\boldsymbol{a}\|^2 = \boldsymbol{a}^T\boldsymbol{a}$

- Training error cost

$$S(\boldsymbol{a}, b) = \left[\frac{1}{N}\sum_{i=1}^{N} max(0, 1 - y_i(\boldsymbol{a}^T\boldsymbol{x}_i + b))\right] + \lambda(\frac{\boldsymbol{a}^T\boldsymbol{a}}{2})$$

- The **regularization parameter** λ trade off between these two objectives

Q. What does the penalty discourage?

$$S(\boldsymbol{a}, b) = \left[\frac{1}{N}\sum_{i=1}^{N} max(0, 1 - y_i(\boldsymbol{a}^T\boldsymbol{x}_i + b))\right] + \boxed{\lambda(\frac{\boldsymbol{a}^T\boldsymbol{a}}{2})}$$

A. Too big a magnitude of the vector **a**

B. Too many data points in the training set

How to compute the decision boundary?

Convex set and convex function

- If a set is convex, any line connecting two points in the set is completely included in the set

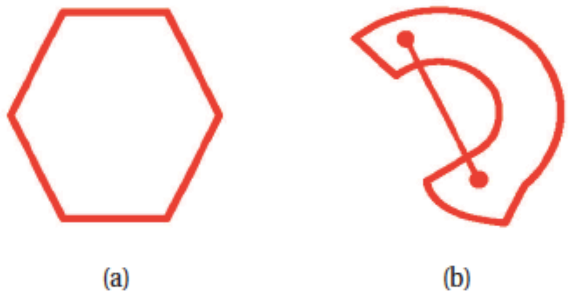
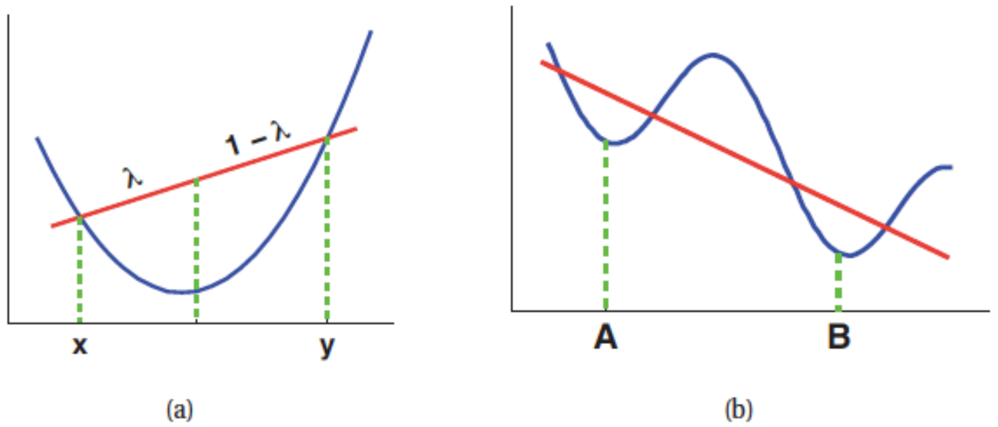


Figure 7.4 (a) Illustration of a convex set. (b) Illustration of a nonconvex set.

- A convex function: the area above the curve is convex

$$f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda) f(y)$$



Credit: Dr. Kelvin Murphy

Q. Is this curve a convex curve?

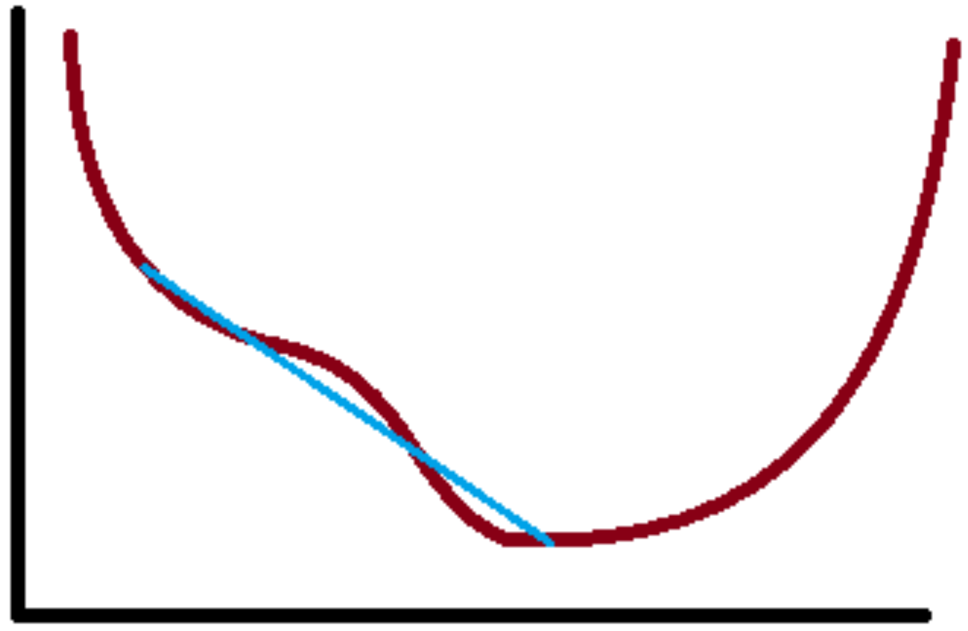
A. YES

B. NO

Q. Is this curve a convex curve?

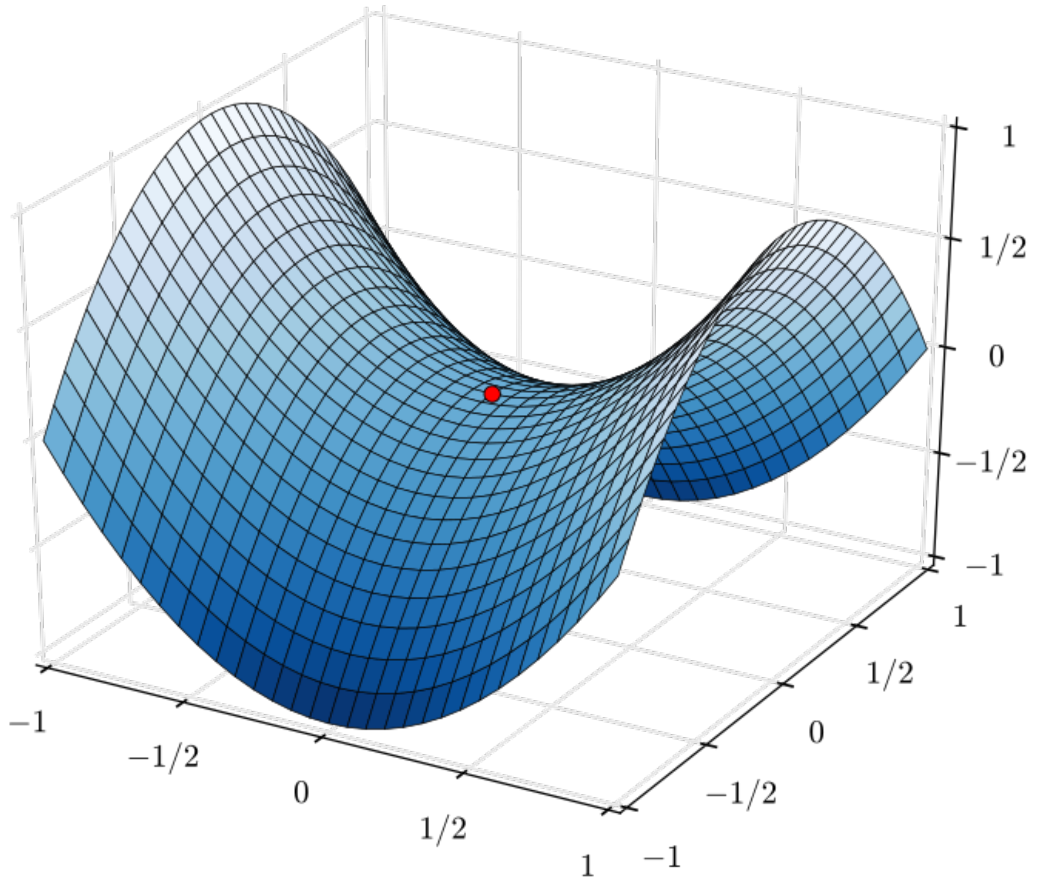
A. YES

B. NO



Q. Is this surface convex?

A. YES
B. NO

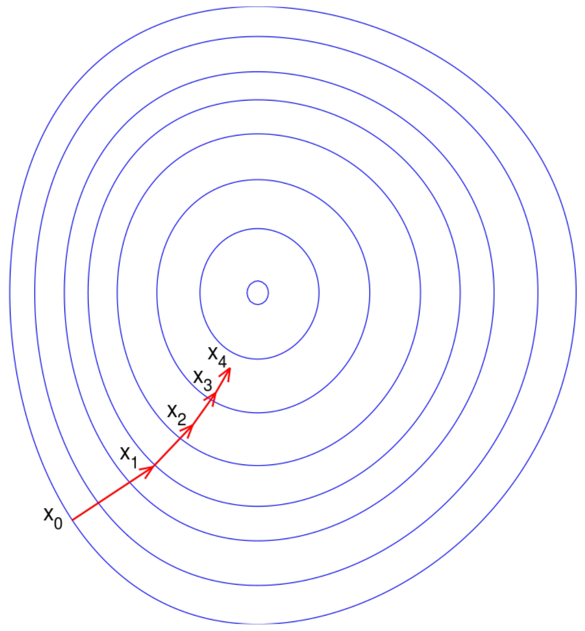


Source: wikipedia

Iterative minimization by gradient descent

- For a function such as

- A convex surface

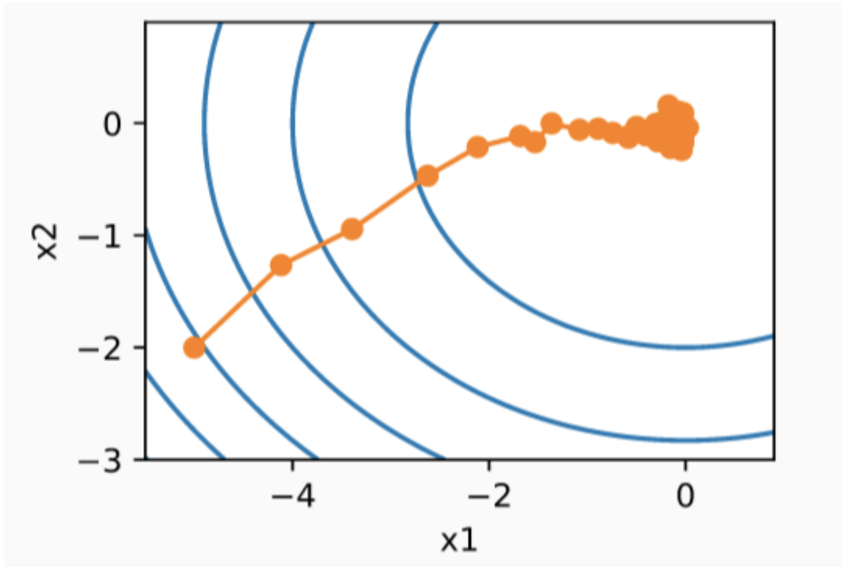


Source: wikipedia

Gradient Descent

Stochastic gradient descent

$\boldsymbol{x}_k \in \{\boldsymbol{x}_i\}$



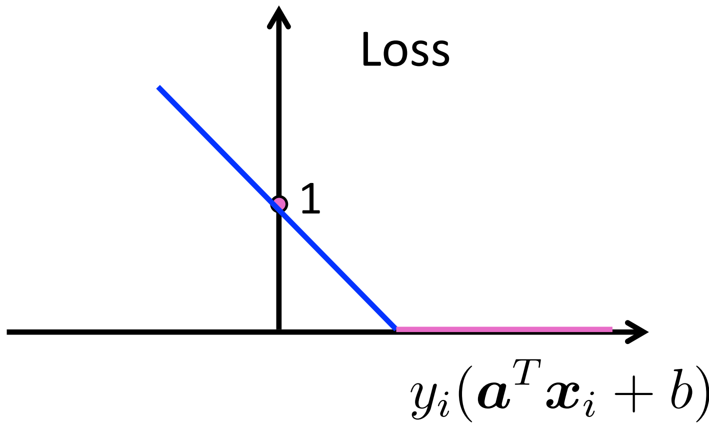
The difference btw GD and SGD

Update parameters of the hyperplane during the stochastic gradient descent

- Since $S_k(\boldsymbol{a}, b) = max(0, 1 - y_k(\boldsymbol{a}^T\boldsymbol{x}_k + b))$ and $S_0(\boldsymbol{a}, b) = \lambda(\frac{\boldsymbol{a}^T\boldsymbol{a}}{2})$

  We have the following updating equations:

If $y_k(\boldsymbol{a}^T\boldsymbol{x}_k + b) \geq 1$	**If** $y_k(\boldsymbol{a}^T\boldsymbol{x}_k + b) < 1$
$\boldsymbol{a} \leftarrow \boldsymbol{a} - \eta(\lambda\boldsymbol{a})$	$\boldsymbol{a} \leftarrow \boldsymbol{a} - \eta(\lambda\boldsymbol{a} - y_k\boldsymbol{x}_k)$
$b \leftarrow b$	$b \leftarrow b - \eta(-y_k)$



Training procedure-minimizing the cost function

- The training error cost $S(\boldsymbol{a}, b)$ is a function of decision boundary parameters $(\boldsymbol{a}, b)$, so it can help us find the best decision boundary.
- Fix λ and set some initial values for $(\boldsymbol{a}, b)$
- Search iteratively for $(\boldsymbol{a}, b)$
- Repeat the previous steps for several values of λ and choose the one that gives the decision boundary with best accuracy on a validation data set.

Validation/testing of SVM model

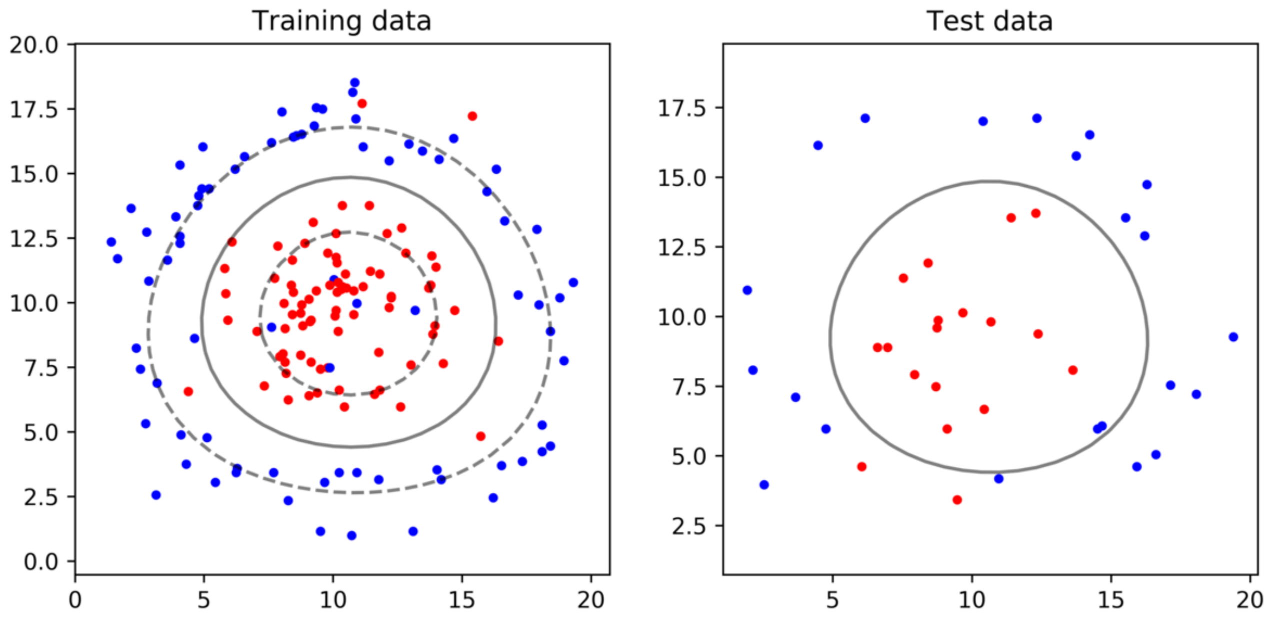
- Split the labeled data into **training**, **validation** and **test** sets.
- For each choice of **λ**, run stochastic gradient descent to find the best decision boundary parameters (**a**, b) using the training set.
- Choose the best **λ** based on accuracy on the validation set.
- Finally evaluate the SVM's accuracy on the **test** set.
- This process avoids overfitting the data.

Extension to multiclass classification

- All vs. all
  - Train a separate binary classifier for each pair of classes.
  - To classify, run all classifiers and see which class it will be labeled most with.
  - Computational complexity is quadratic to the number of classes.
- One vs. all
  - Train a separate binary classifier for each class against all else.
  - To classify, run all classifiers and see which label gets the highest score
  - Computational complexity scales linearly.

What if the data is inseparable linearly?

- There is a chance the data is inseparable
- Use the non-linear **SVM with kernels!**
- Decision boundary is curved



Naïve Bayes classifier

- Training
  - Use the training data $\{(\mathbf{x}_i, y_i)\}$ to estimate a probability model $P(y|\boldsymbol{x})$
  - Assume that the features of $\{\boldsymbol{x}\}$ are conditionally independent given the class label y

$$P(\boldsymbol{x}|y) = \prod_{j=1}^{d} P(\boldsymbol{x}^{(j)}|y)$$

- Classification
  - Assign the label $\underset{y}{argmax}\ P(y|\boldsymbol{x})$ to a feature vector $\boldsymbol{x}$

Naïve Bayes Model

- MAP estimator of class variable y given the data $\boldsymbol{x}$

$$\underset{y}{argmax}\ P(y|\boldsymbol{x})$$

$$=$$

Naïve Bayes Model

- MAP estimator of class variable y given the data $\boldsymbol{x}$

$$\begin{aligned} &\underset{y}{argmax}\ P(y|\boldsymbol{x}) \\ &= \underset{y}{argmax}\ \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})} \end{aligned}$$

Naïve Bayes Model

- MAP estimator of class variable y given the data $\boldsymbol{x}$

$$\begin{aligned} &\underset{y}{argmax}\ P(y|\boldsymbol{x}) \\ &= \underset{y}{argmax}\ \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})} \\ &= \underset{y}{argmax}\ P(\boldsymbol{x}|y)P(y) \end{aligned}$$

Because P(x) doesn't depend on y

Naïve Bayes Model

- MAP estimator of class variable y given the data $\boldsymbol{x}$

$$
\begin{aligned}
&\underset{y}{argmax}\ P(y|\boldsymbol{x}) \\
&= \underset{y}{argmax}\ \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})} \\
&= \underset{y}{argmax}\ P(\boldsymbol{x}|y)P(y) \\
&= \underset{y}{argmax}\ \left[\prod_{j=1}^{d} P(\boldsymbol{x}^{(j)}|y)\right] P(y)
\end{aligned}
$$

"Naïve" assumption of conditional independence of features

Naïve Bayes Model

- MAP estimator of class variable y given the data $\boldsymbol{x}$

$$\underset{y}{argmax}\ P(y|\boldsymbol{x})$$

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

$$= \underset{y}{argmax}\ P(\boldsymbol{x}|y)P(y)$$

$$= \underset{y}{argmax}\ \left[\prod_{j=1}^{d} P(\boldsymbol{x}^{(j)}|y)\right] P(y)$$

"Naïve" assumption of conditional independence of features

$$= \underset{y}{argmax}\ \left[\sum_{j=1}^{d} logP(\boldsymbol{x}^{(j)}|y) + log\ P(y)\right]$$

Modeling the prior and the likelihoods

- Model the prior based on the frequency of y in the training set
  - For a binary classifier, this model is a Bernoulli random variable

- Model each likelihood $P(\boldsymbol{x}^{(j)}|y)$ by:
  - Selecting an appropriate family of distributions
    - Normal for real-valued numerical data
    - Poisson for counts in fixed intervals
    - Etc.
  - Fitting the parameters of the distribution using MLE

An example of Naive Bayes training

Training data

$x^{(1)}$	$x^{(2)}$	y
3.5	10	1
1.0	8	1
0.0	10	-1
-3.0	14	-1

Modeling $P(\boldsymbol{x}^{(1)}|y)$ as normal

$$P(\boldsymbol{x}^{(1)}|y=1)$$

$$\mu_{MLE} = \frac{3.5+1.0}{2} = 2.25$$

$$\sigma_{MLE} = 1.25$$

$$P(\boldsymbol{x}^{(1)}|y=-1)$$

$$\mu_{MLE} = -1.5$$

$$\sigma_{MLE} = 1.5$$

Modeling $P(\boldsymbol{x}^{(2)}|y)$ as Poisson

$$P(\boldsymbol{x}^{(2)}|y=1)$$

$$\lambda_{MLE} = \frac{10+8}{2} = 9$$

$$P(\boldsymbol{x}^{(2)}|y=-1)$$

$$\lambda_{MLE} = 12$$

Modeling $P(y)$ as Bernoulli

$$P(y=1) = \frac{2}{4} = 0.5$$

$$P(y=-1) = 0.5$$

Classification example:

For a new feature vector x = [x1,x2,...], ie x = [3,9] in the example

$$\underset{y}{argmax} \left[\sum_{j=1}^{d} log P(\boldsymbol{x}^{(j)}|y) + log\ P(y) \right]$$

Classification example:

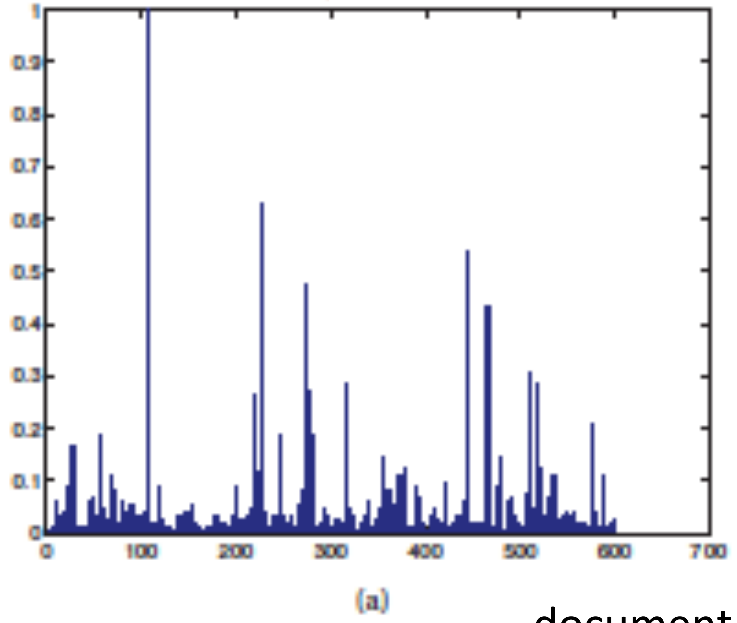
For a new feature vector x = [x1,x2,...], ie x = [3,9] in the example

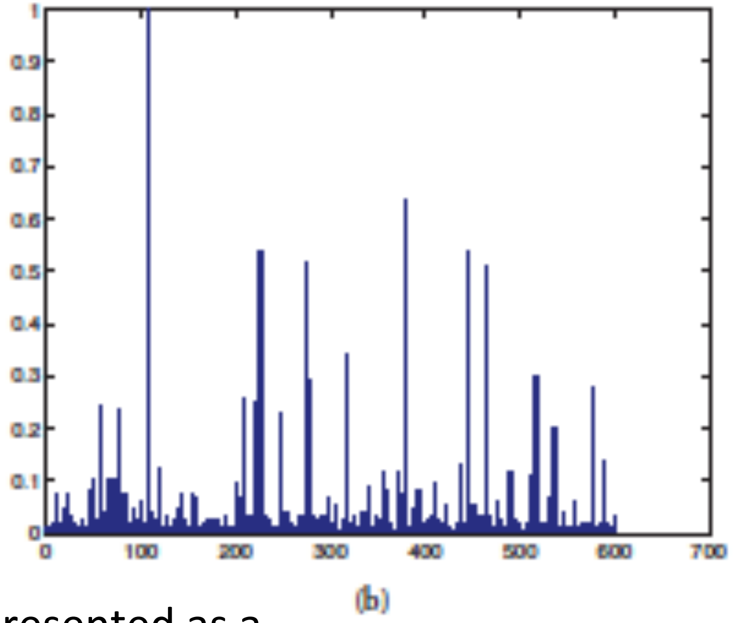
$$\underset{y}{argmax} \left[\sum_{j=1}^{d} logP(\boldsymbol{x}^{(j)}|y) + log\ P(y) \right]$$

$$g(y) = \left\{ \right.$$

Example of Naïve Bayesian Model

"Bag of words" Naive Bayesian models for document class





X-windows

document (represented as a bag-of-words bit vector), each column is a word

MS-windows

What about the decision boundary?

- Not explicit as in the case of decision tree
- This method is parametric, generative
  - The model was specified with parameters to generate label for test data

Pros and Cons of Naïve Bayesian Classifier

- Pros:
  - Simple approach
  - Good accuracy
  - Good for high dimensional data
- Cons:
  - The assumption of conditional independence of features
  - No explicit decision boundary
  - Sometimes has numerical issues

Assignments

- Finish Chapter 11 of the textbook
- Next time: Linear regression

Additional References

- Robert V. Hogg, Elliot A. Tanis and Dale L. Zimmerman. “Probability and Statistical Inference”
- Kelvin Murphy, “Machine learning, A Probabilistic perspective”

See you next time

See You!