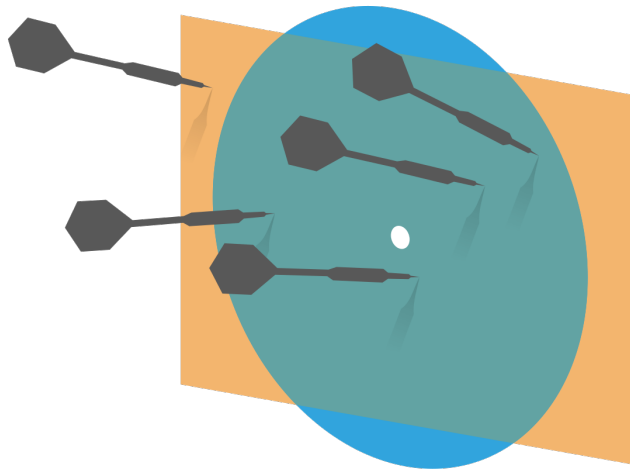


Probability and Statistics for Computer Science



“...many problems are naturally
classification problems” ---Prof.
Forsyth

Credit: wikipedia

Last time

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

Content

- ✱ Support vector machine (II)
 - ✱ Stochastic Gradient Descent
- ✱ Naïve Bayesian Classifier

Considerations in choosing a classifier

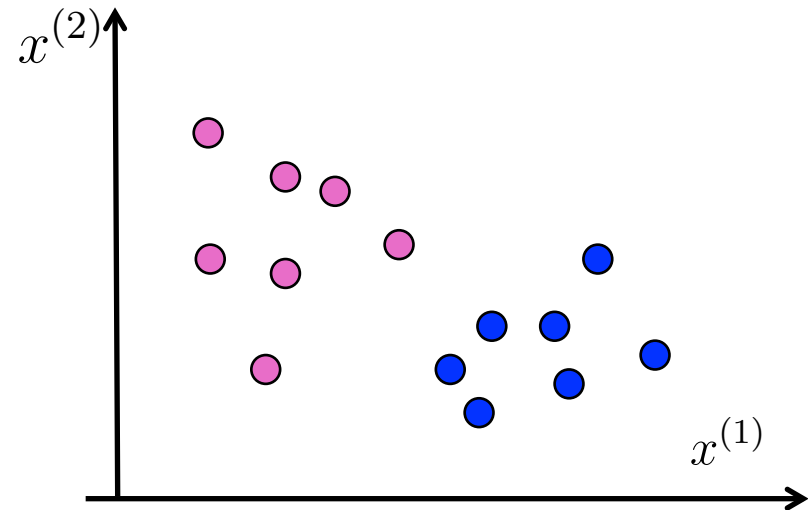
- ✱ When solving a classification problem, it is good to try several techniques.
- ✱ Criteria to consider in choosing the classifier include
 - ✱ Accuracy
 - ✱ Training speed
 - ✱ Classification speed
 - ✱ Performance with small training set
 - ✱ Interpretability

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

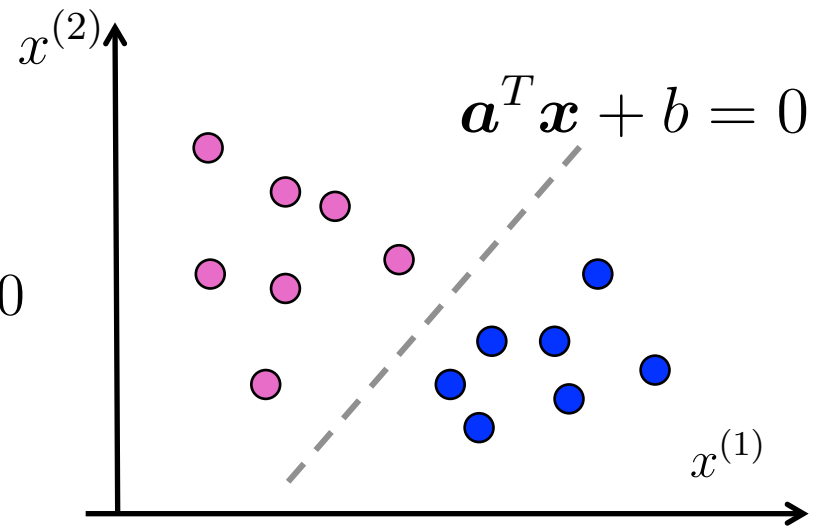
- ✱ SVM uses a hyperplane as its **decision boundary**

- ✱ The decision boundary is:

$$a_1x^{(1)} + a_2x^{(2)} + \dots + a_dx^{(d)} + b = 0$$

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

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



Classification function of SVM

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

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

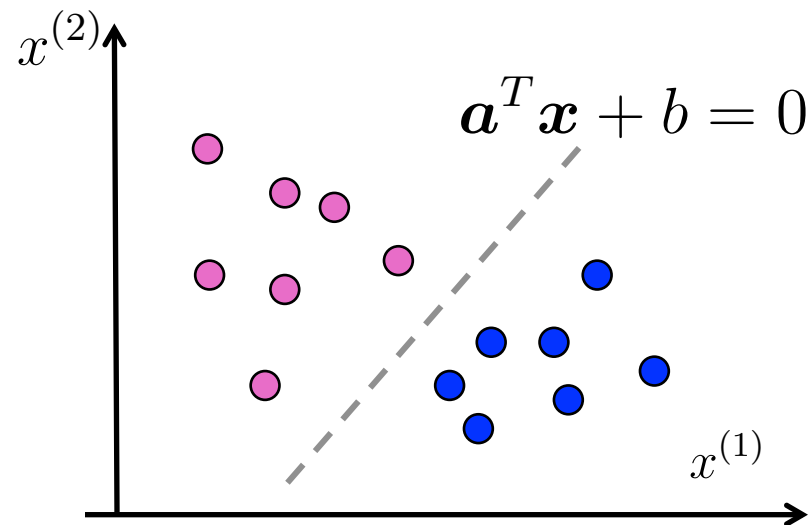
$$-1 \text{ if } \mathbf{a}^T \mathbf{x}_i + b < 0$$

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

- ✱ Note that

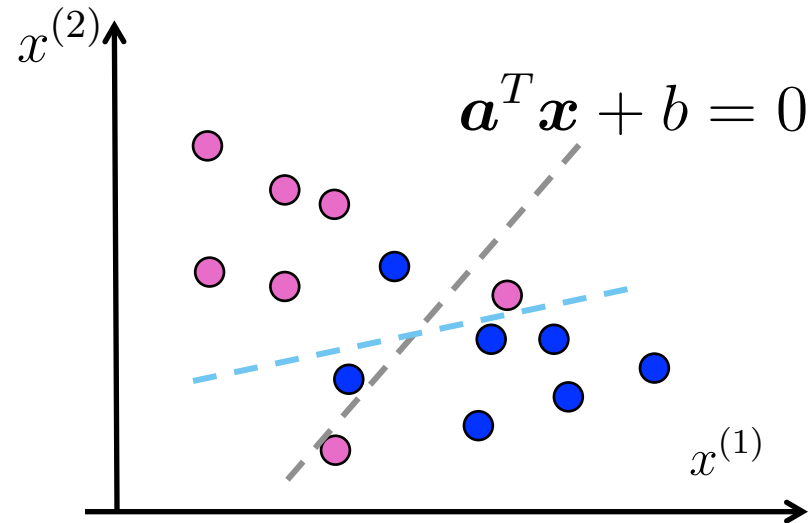
- ✱ If $|\mathbf{a}^T \mathbf{x}_i + b|$ is small, then $\mathbf{x}_i$ was close to the decision boundary

- ✱ If $|\mathbf{a}^T \mathbf{x}_i + b|$ is large, then $\mathbf{x}_i$ 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 a quantitative measure to decide about the boundary
- ✱ The **loss function** can help decide if one boundary is better than others



Loss function 1

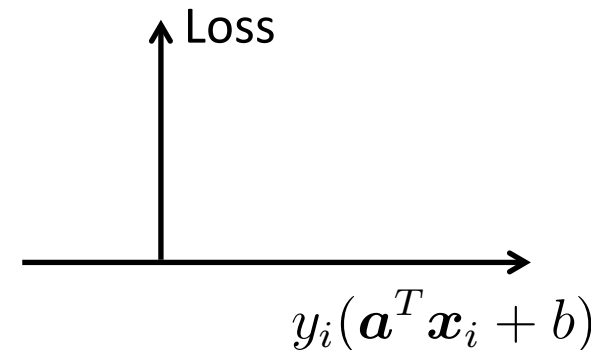
- ✱ For any given feature vector $\mathbf{x}_i$ with class label $y_i \in \{\pm 1\}$, we want
 - ✱ Zero loss if $\mathbf{x}_i$ is classified correctly $\text{sign}(\mathbf{a}^T \mathbf{x}_i + b) = y_i$
 - ✱ Positive loss if $\mathbf{x}_i$ is misclassified $\text{sign}(\mathbf{a}^T \mathbf{x}_i + b) \neq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

- ✱ This loss function 1 meets the criteria above:

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

- ✱ Training error cost

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



Q. What's the value of this function ?

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

A. 0.

B. others.

Q. What's the value of this function ?

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

A. 0.

B. A value greater than or equal to 0.

Loss function 1

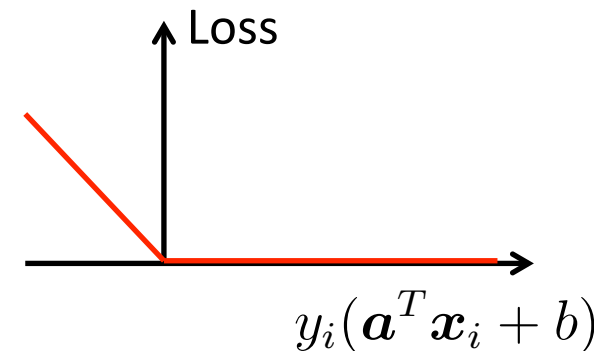
- ✱ For any given feature vector $\mathbf{x}_i$ with class label $y_i \in \{\pm 1\}$, we want
 - ✱ Zero loss if $\mathbf{x}_i$ is classified correctly $\text{sign}(\mathbf{a}^T \mathbf{x}_i + b) = y_i$
 - ✱ Positive loss if $\mathbf{x}_i$ is misclassified $\text{sign}(\mathbf{a}^T \mathbf{x}_i + b) \neq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

- ✱ This loss function 1 meets the criteria above:

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

- ✱ Training error cost

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



The problem with loss function 1

- ✱ Loss function 1 does not distinguish between the following decision boundaries if they both classify $\mathbf{x}_i$ 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**

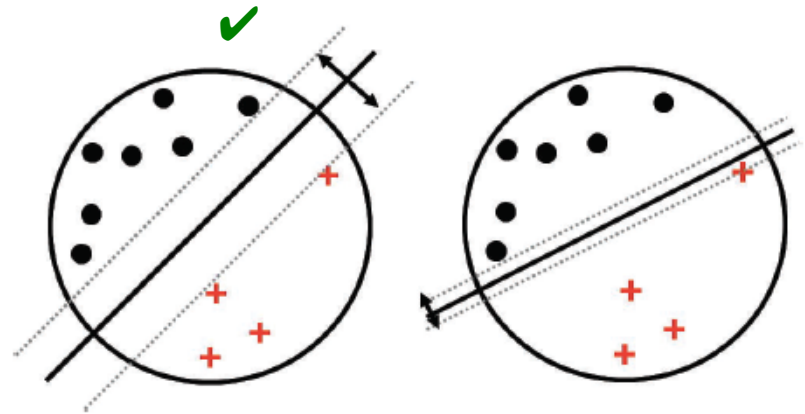


Figure 14.11 Illustration of the large margin principle. Left: a separating hyper-plane with large margin. Right: a separating hyper-plane with small margin.

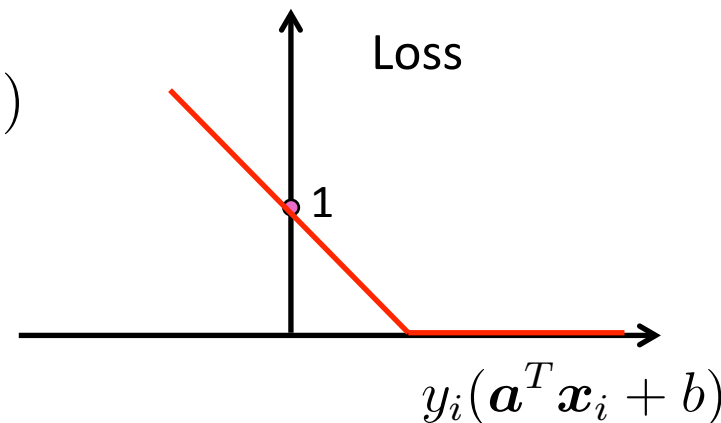
Loss function 2: the hinge loss

- ✱ We want to impose a small positive loss if $\mathbf{x}_i$ is correctly classified but close to the boundary
- ✱ The **hinge loss** function meets the criteria above:

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

- ✱ Training error cost

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



The problem with loss function 2

- ✱ Loss function 2 favors decision boundaries that have large $\|\mathbf{a}\|$ because increasing $\|\mathbf{a}\|$ can zero out the loss for a correctly classified $\mathbf{x}_i$ near the boundary.
- ✱ But large $\|\mathbf{a}\|$ makes the classification function $\text{sign}(\mathbf{a}^T \mathbf{x}_i + b)$ extremely sensitive to small changes in $\mathbf{x}_i$ and make it less robust to run-time data.
- ✱ So small $\|\mathbf{a}\|$ is better.

Hinge loss with regularization penalty

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

- ✱ Training error cost

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

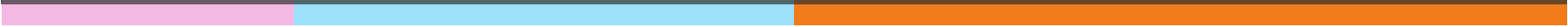
- ✱ The **regularization parameter** λ trade off between these two objectives

Q. What does the penalty discourage?

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

- A. Too big a magnitude of the vector $\mathbf{a}$
- 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



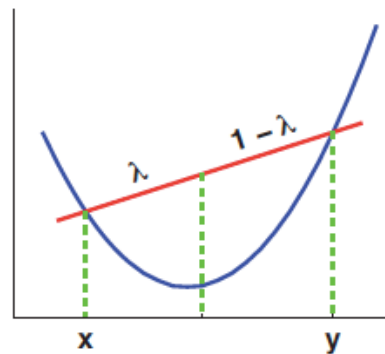
(a)



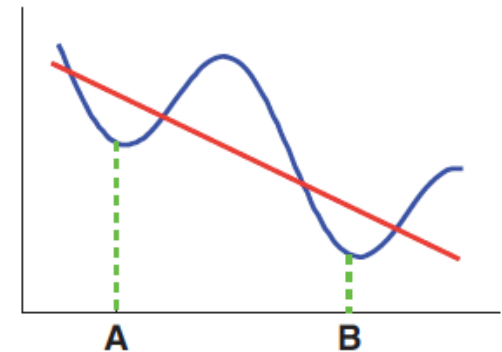
(b)

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



(a)



(b)

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

Q. Is this curve a convex curve?

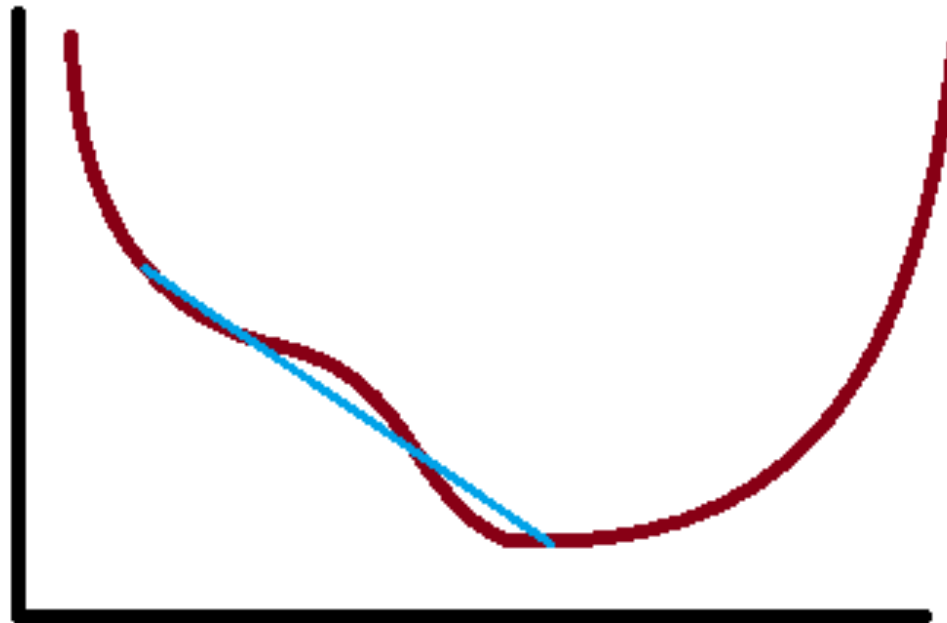
A.YES

B.NO

Q. Is this curve a convex curve?

A.YES

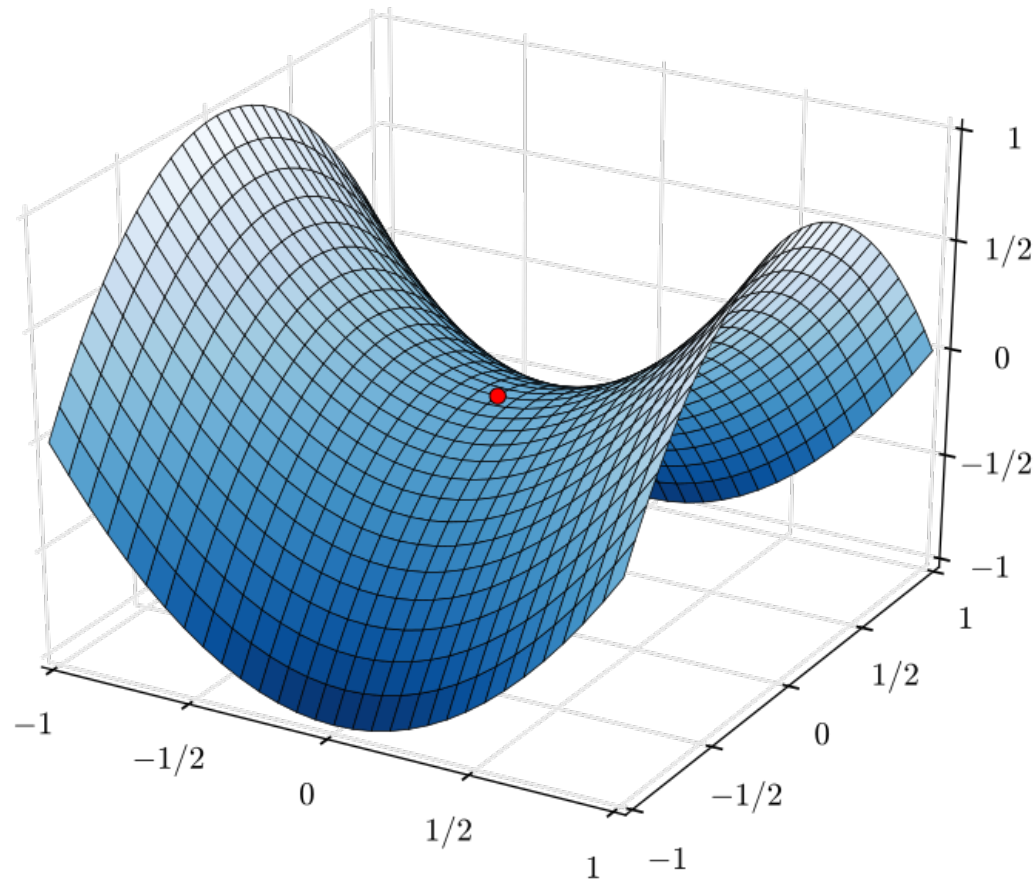
B.NO



Q. Is this surface convex?

A.YES

B.NO



Source: wikipedia

Training procedure-minimizing the cost function

- ✱ The training error cost $S(\mathbf{a}, b)$ is a function of decision boundary parameters $(\mathbf{a}, b)$, so it can help us find the best decision boundary.
- ✱ Fix λ and set some initial values for $(\mathbf{a}, b)$
- ✱ Search iteratively for $(\mathbf{a}, b)$
- ✱ 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.

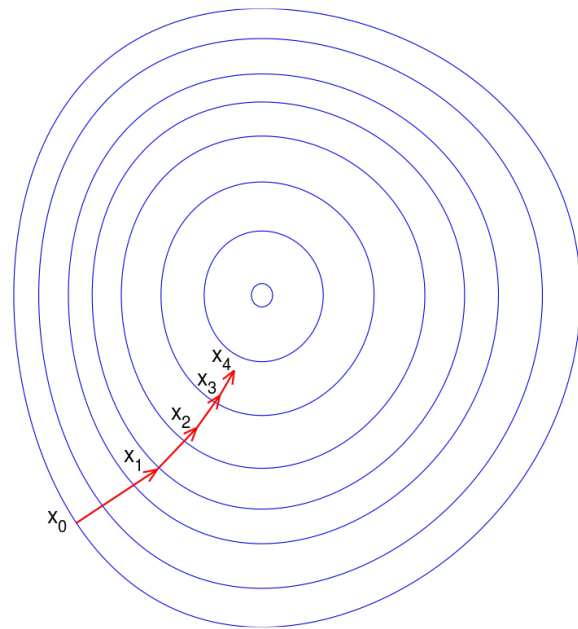
Iterative minimization by gradient descent

✱ For simplicity, we write

$$S(\mathbf{a}, b) = \left[\frac{1}{N} \sum_{i=1}^N \max(0, 1 - y_i(\mathbf{a}^T \mathbf{x}_i + b)) \right] + \lambda \left(\frac{\mathbf{a}^T \mathbf{a}}{2} \right) = \left[\frac{1}{N} \sum_{i=1}^N S_i(\mathbf{a}, b) \right] + S_0(\mathbf{a}, b)$$

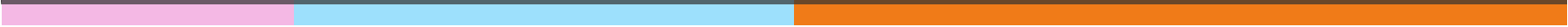
✱ The direction of steepest descent is

$$-\nabla S(\mathbf{a}, b) = - \left[\frac{1}{N} \sum_{i=1}^N \nabla S_i(\mathbf{a}, b) \right] - \nabla S_0(\mathbf{a}, b)$$



Source: wikipedia

First order approximation

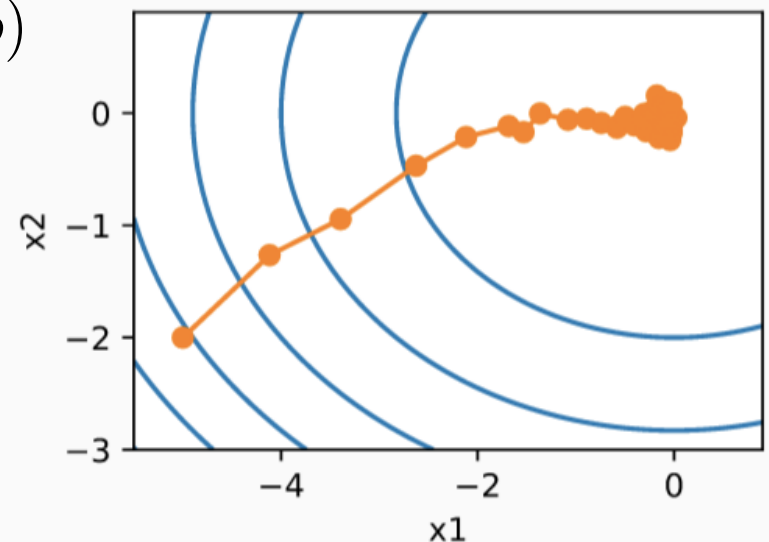


Stochastic gradient descent

- ✱ The exact $\nabla S(\mathbf{a}, b)$ is too expensive for large N, and some functions' gradient is hard to compute with formulas. We need to calculate numerically. Please note the $\max()$ causes difficulty in our case.
- ✱ Strategy: Approximate with **random sampling of a vector** $\mathbf{x}_k \in \{\mathbf{x}_i\}$

$$-\nabla S(\mathbf{a}, b) \approx -\nabla S_k(\mathbf{a}, b) - \nabla S_0(\mathbf{a}, b)$$

$$\nabla S_k(\mathbf{a}, b) = \begin{bmatrix} \frac{\partial S_k}{\partial a_1} \\ \vdots \\ \frac{\partial S_k}{\partial b} \end{bmatrix}$$



Update parameters of the hyperplane during the stochastic gradient descent

- ✱ Since $S_k(\mathbf{a}, b) = \max(0, 1 - y_k(\mathbf{a}^T \mathbf{x}_k + b))$ and $S_0(\mathbf{a}, b) = \lambda(\frac{\mathbf{a}^T \mathbf{a}}{2})$
We have the following updating equations:

if $y_k(\mathbf{a}^T \mathbf{x}_k + b) \geq 1$

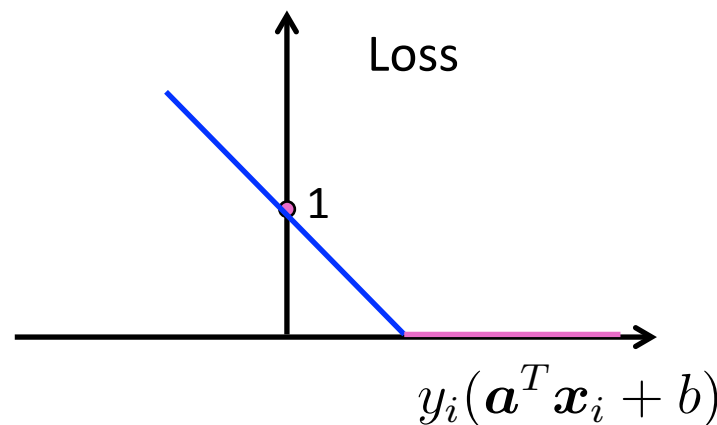
$$\mathbf{a} \leftarrow \mathbf{a} - \eta(\lambda \mathbf{a})$$

$$b \leftarrow b$$

if $y_k(\mathbf{a}^T \mathbf{x}_k + b) < 1$

$$\mathbf{a} \leftarrow \mathbf{a} - \eta(\lambda \mathbf{a} - y_k \mathbf{x}_k)$$

$$b \leftarrow b - \eta(-y_k)$$



Iteration scheme during the stochastic gradient descent

- ✱ The number of steps is called **epoch**, which is usually set at the size **N** of the training set.

- ✱ In the e th epoch, it is common to choose a **steplength** η (also called learning rate) which changes from bigger to smaller.

$$\eta = \frac{m}{e + n}$$

- ✱ Where m and n are constants selected by experiments.

The difference btw GD and SGD



Validation/testing of SVM model

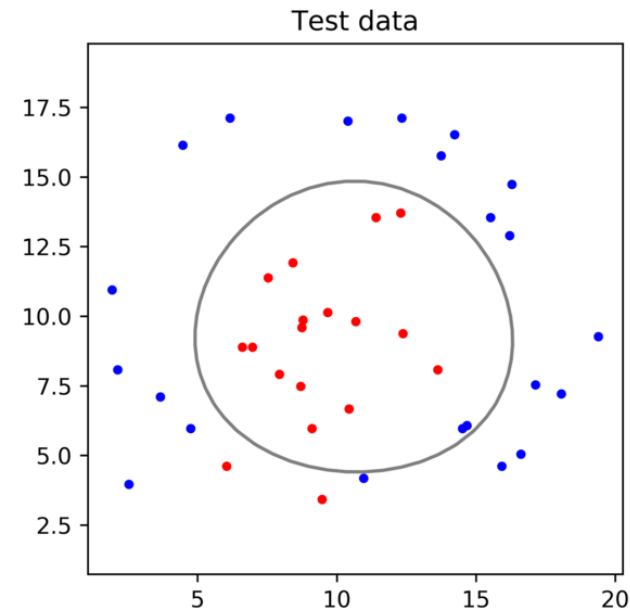
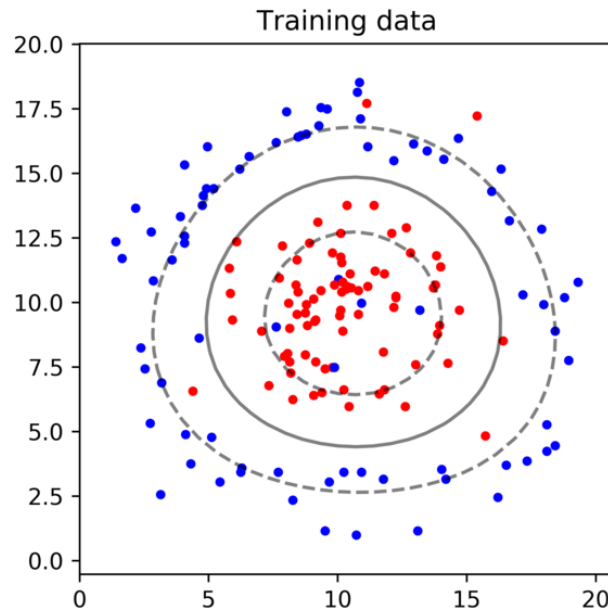
- ✱ Split the labeled data into **training**, **validation** and **test** sets.
- ✱ For each choice of λ , run stochastic gradient descent to find the best decision boundary parameters ($\mathbf{a}$, b) using the training set.
- ✱ Choose the best λ based on accuracy on the validation set.
- ✱ Finally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

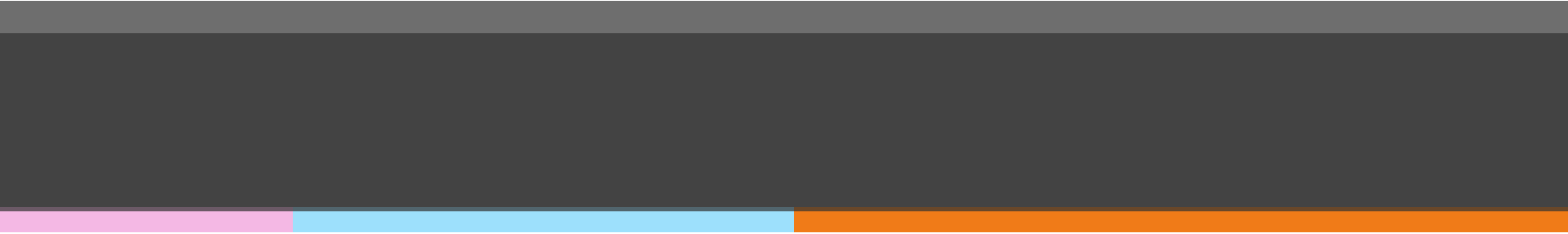
Extension to multiclass classification

- ✱ 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 $P(y|\mathbf{x})$
- ✱ Assume that the features of $\{\mathbf{x}\}$ are conditionally independent given the class label y

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

✱ Classification

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

Naïve Bayes Model

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

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

=

Naïve Bayes Model

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

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

Naïve Bayes Model

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

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

Because $P(\mathbf{x})$ doesn't depend on y

Naïve Bayes Model

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

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

“Naïve” assumption
of conditional
independence of
features



Naïve Bayes Model

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

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

“Naïve” assumption of conditional independence of features



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 $P(\mathbf{x}^{(j)}|y)$ 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

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

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

$$P(\mathbf{x}^{(1)}|y = 1)$$
$$\mu_{MLE} = \frac{3.5 + 1.0}{2} = 2.25$$
$$\sigma_{MLE} = 1.25$$

$$P(\mathbf{x}^{(1)}|y = -1)$$
$$\mu_{MLE} = -1.5$$
$$\sigma_{MLE} = 1.5$$

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

$$P(\mathbf{x}^{(2)}|y = 1)$$
$$\lambda_{MLE} = \frac{10 + 8}{2} = 9$$

$$P(\mathbf{x}^{(2)}|y = -1)$$
$$\lambda_{MLE} = 12$$

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 $\mathbf{x} = [x_1, x_2, \dots]$, ie $\mathbf{x} = [3, 9]$ in the example

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

Classification example:

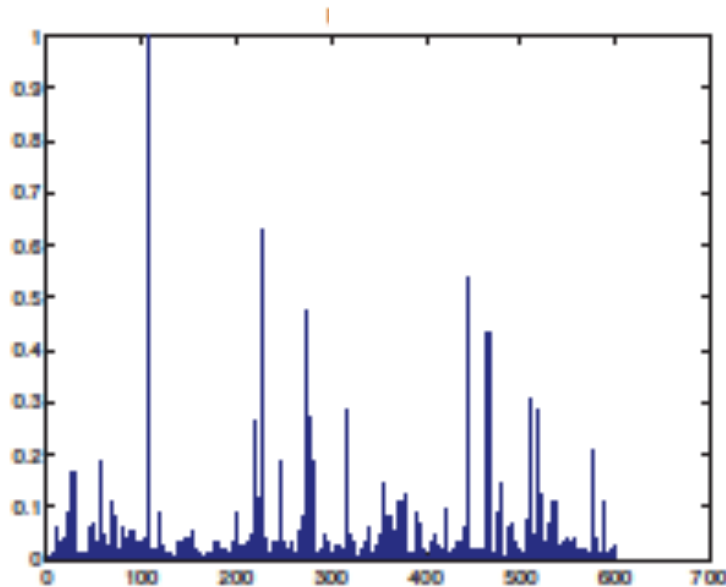
For a new feature vector $x = [x_1, x_2, \dots]$, ie $x = [3, 9]$ in the example

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

$$g(y) = \begin{cases} \log \frac{e^{-\frac{(3-2.25)^2}{2 \times 1.25^2}}}{\sqrt{2\pi} \times 1.25} + \log \frac{e^{-9} 9^9}{9!} + \log \frac{1}{2} & ; \quad y = 1 \\ \log \frac{e^{-\frac{(3-1.5)^2}{2 \times 1.5^2}}}{\sqrt{2\pi} \times 1.5} + \log \frac{e^{-12} 12^9}{9!} + \log \frac{1}{2} & ; \quad y = -1 \end{cases}$$

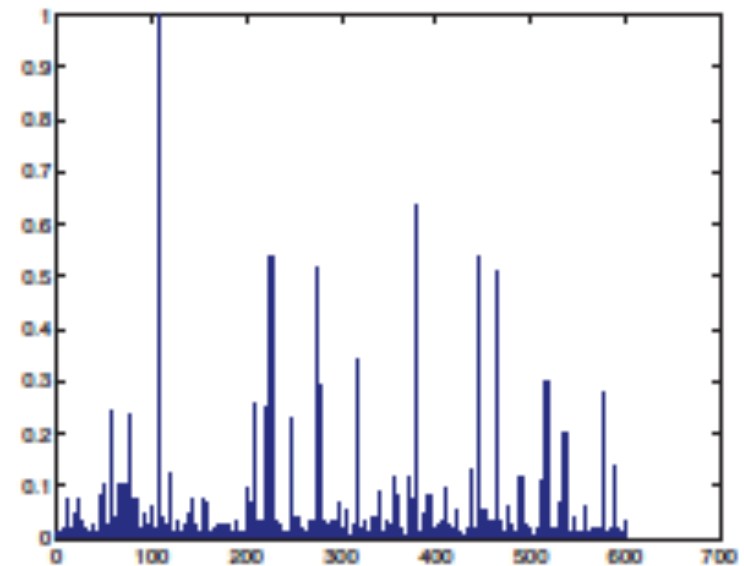
Example of Naïve Bayesian Model

“Bag of words” Naive Bayesian models for document class



(a)

X-windows



(b)

MS-windows

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

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!*

