COMPSCI 589
Lecture 5: Support Vector Machines, Basis Expansion and Kernels

Benjamin M. Marlin

College of Information and Computer Sciences
University of Massachusetts Amherst

Slides by Benjamin M. Marlin (marlin@cs.umass.edu).
Created with support from National Science Foundation Award# IIS-1350522.

Outline

Overview

- To date we've seen one example of a discriminative linear classifier.

Overview

- To date we've seen one example of a discriminative linear classifier.
- Today we'll introduce a second example, support vector machines.

Overview

- To date we've seen one example of a discriminative linear classifier.
- Today we'll introduce a second example, support vector machines.
- We'll then address the question of how to increase the capacity of linear classifiers so they can produce non-linear classification boundaries.

Outline

Support Vector Machines

- A binary support vector machine is a discriminative classifier that takes labels in the set $\{-1, 1\}$.

Support Vector Machines

- A binary support vector machine is a discriminative classifier that takes labels in the set $\{-1, 1\}$.
- The decision function has the form:

$$f_{SVM}(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$$

Support Vector Machines

- A binary support vector machine is a discriminative classifier that takes labels in the set $\{-1, 1\}$.
- The decision function has the form:

$$f_{SVM}(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$$

- It's easy to show that the decision boundary for logistic regression can be written in exactly the same way.

Support Vector Machines

- A binary support vector machine is a discriminative classifier that takes labels in the set $\{-1, 1\}$.
- The decision function has the form:

$$f_{SVM}(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$$

- It's easy to show that the decision boundary for logistic regression can be written in exactly the same way.
- **Question:** If logistic regression and SVMs have the same form for their decision boundaries, how do the differ?

Logistic Loss

- In the case of logistic regression with ℓ_2 regularization, we select the model parameters by maximizing the function:

$$C\sum_{i=1}^{n}\log P(Y=y_i|\mathbf{X}=\mathbf{x}_i)-||\mathbf{w}||_2^2$$

Logistic Loss

- In the case of logistic regression with ℓ_2 regularization, we select the model parameters by maximizing the function:

$$C \sum_{i=1}^{n} \log P(Y = y_i | \mathbf{X} = \mathbf{x}_i) - ||\mathbf{w}||_2^2$$

- Under the assumption that the labels take the values $\{-1, 1\}$, it can be shown that this is equivalent to minimizing the function:

$$C \sum_{i=1}^{n} \log(1 + \exp(-y_i \cdot g(\mathbf{x}))) + ||\mathbf{w}||_2^2$$

where $L_{log}(y_i, g(\mathbf{x}_i)) = \log(1 + \exp(-y_i \cdot g(\mathbf{x})))$ is the *logistic loss function* and $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$.

Hinge Loss

- In the case of SVMs with ℓ_2 regularization, we select the model parameters by minimizing the function:

$$C\sum_{i=1}^{n}\max(0, 1 - y_i \cdot g(\mathbf{x}_i)) + ||\mathbf{w}||_2^2$$

Hinge Loss

- In the case of SVMs with ℓ_2 regularization, we select the model parameters by minimizing the function:

$$C\sum_{i=1}^{n}\max(0,1-y_i\cdot g(\mathbf{x}_i))+||\mathbf{w}||_2^2$$

- The function $L_h(y_i, g(\mathbf{x}_i)) = \max(0, 1 - y_i \cdot g(\mathbf{x}_i))$ is called the *hinge loss*.

Zero-One Loss

- Both the logistic loss and the hinge loss are convex upper bounds on the zero-one loss:

$$L_{01}(y_i, g(\mathbf{x}_i)) = \mathbb{I}[y_i \neq \text{sign}(g(\mathbf{x}_i))]$$

Zero-One Loss

- Both the logistic loss and the hinge loss are convex upper bounds on the zero-one loss:

$$L_{01}(y_i, g(\mathbf{x}_i)) = \mathbb{I}[y_i \neq \text{sign}(g(\mathbf{x}_i))]$$

- The average zero-one loss over a data set is exactly the classification error rate.

Zero-One Loss

- Both the logistic loss and the hinge loss are convex upper bounds on the zero-one loss:

$$L_{01}(y_i, g(\mathbf{x}_i)) = \mathbb{I}[y_i \neq \text{sign}(g(\mathbf{x}_i))]$$

- The average zero-one loss over a data set is exactly the classification error rate.
- This is the loss function we'd like to minimize, but this generally isn't computationally feasible, thus the need for surrogate loss functions.

Zero-One Loss

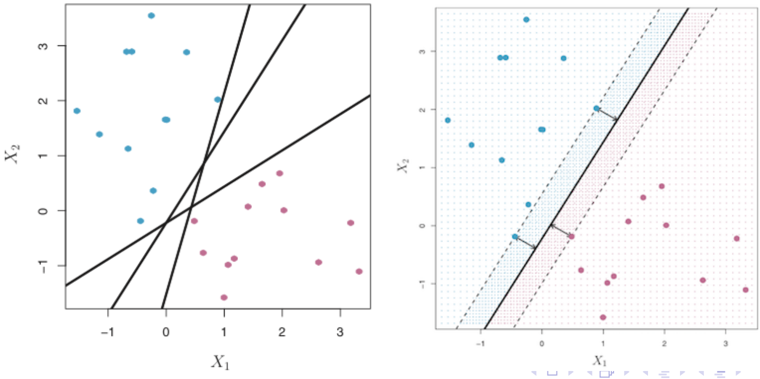
- Both the logistic loss and the hinge loss are convex upper bounds on the zero-one loss:

$$L_{01}(y_i, g(\mathbf{x}_i)) = \mathbb{I}[y_i \neq \text{sign}(g(\mathbf{x}_i))]$$

- The average zero-one loss over a data set is exactly the classification error rate.
- This is the loss function we'd like to minimize, but this generally isn't computationally feasible, thus the need for surrogate loss functions.
- Hinge loss has some advantages over logistic loss, as we'll see.

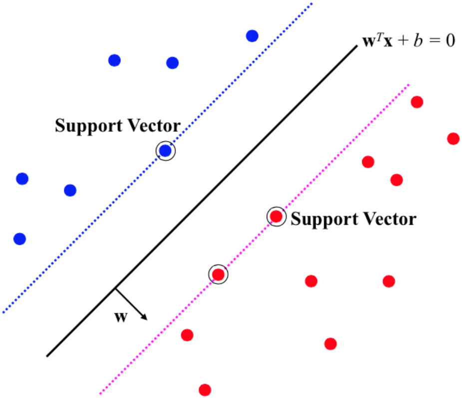
Maximum Margin Property

Part of popularity of SVMs stems from the fact that the hinge loss results in the *maximum margin* decision boundary when the training cases are linearly separable.



Support Vector Property

In the linearly separable case, some data points will always fall exactly on the margins. These points are called *support vectors*.



SVMs vs Logistic Regression

- SVMs and Logistic regression are both discriminative linear classifiers with identical capacity and space complexity.

SVMs vs Logistic Regression

- SVMs and Logistic regression are both discriminative linear classifiers with identical capacity and space complexity.
- SVMs and Logistic regression have very similar convex loss functions and identical regularizers.

SVMs vs Logistic Regression

- SVMs and Logistic regression are both discriminative linear classifiers with identical capacity and space complexity.
- SVMs and Logistic regression have very similar convex loss functions and identical regularizers.
- The hinge loss is not differentiale, unlike the logistic loss, so SVMs require more advanced optimization methods (sub-gradient descent or quadratic programming), but there are extremely good algorithms and implementations available.

SVMs vs Logistic Regression

- SVMs and Logistic regression are both discriminative linear classifiers with identical capacity and space complexity.
- SVMs and Logistic regression have very similar convex loss functions and identical regularizers.
- The hinge loss is not differentiale, unlike the logistic loss, so SVMs require more advanced optimization methods (sub-gradient descent or quadratic programming), but there are extremely good algorithms and implementations available.
- The maximum-margin property of SVMs can yield better generalization than logistic regression when data is scarce.

SVMs vs Logistic Regression

- It is somewhat more difficult to form a multi-class SVM than a multi-class logistic regression model, and many implementations use "hacks" like one-vs-all.

SVMs vs Logistic Regression

- It is somewhat more difficult to form a multi-class SVM than a multi-class logistic regression model, and many implementations use “hacks” like one-vs-all.
- Unlike logistic regression, SVMs do not produce probabilistic outputs, but again, there are hacks that can estimate probabilities.

Outline

The Problem with Linear Models

- The problem with linear classifiers is that their decision boundaries are by definition linear in the input feature space.

The Problem with Linear Models

- The problem with linear classifiers is that their decision boundaries are by definition linear in the input feature space.
- This means that they can have very high bias on complex data.

The Problem with Linear Models

- The problem with linear classifiers is that their decision boundaries are by definition linear in the input feature space.
- This means that they can have very high bias on complex data.
- **Question:** How can we relax the constraint of linear decision boundaries while retaining the nice properties of linear classifiers?

Basis Function Expansion

- One very simple solution is to apply a set of functions $\phi_1,...,\phi_K$ to the raw feature vector $\mathbf{x}$ to map it in to a new feature space:

$$\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$$

Basis Function Expansion

- One very simple solution is to apply a set of functions $\phi_1,...,\phi_K$ to the raw feature vector $\mathbf{x}$ to map it in to a new feature space:

$$\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$$

- This is called a *basis function expansion* since $K > D$ in general. This requires that we know the functions $\phi_1,...,\phi_K$ that we want to apply in advance.

Basis Function Expansion

- One very simple solution is to apply a set of functions ϕ_1,...,ϕ_K to the raw feature vector $\mathbf{x}$ to map it in to a new feature space:

$$\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_K(\mathbf{x})]$$

- This is called a *basis function expansion* since $K > D$ in general. This requires that we know the functions ϕ_1,...,ϕ_K that we want to apply in advance.

- We then define a linear classifier (SVM or Logistic Regression) in this new feature space:

$$\mathbf{w}^T\phi(\mathbf{x}) + b$$

Basis Function Expansion Examples

- **Degree 2 Polynomial Basis:** We include all single features x_d, their squares x_d^2, and all products of two distinct features $x_d x_{d'}$.

Basis Function Expansion Examples

- **Degree 2 Polynomial Basis:** We include all single features x_d, their squares x_d^2, and all products of two distinct features $x_d x_{d'}$.
- **Degree B Polynomial Basis:** We include all single features x_d, and all unique products of between 2 and B features.

Basis Function Expansion Examples

- **Degree 2 Polynomial Basis:** We include all single features x_d, their squares x_d^2, and all products of two distinct features $x_d x_{d'}$.
- **Degree B Polynomial Basis:** We include all single features x_d, and all unique products of between 2 and B features.
- The problem is that the space complexity of representing the expanded set of features is essentially $O(D^B)$.

Basis Function Expansion Examples

- **Degree 2 Polynomial Basis:** We include all single features x_d, their squares x_d^2, and all products of two distinct features $x_d x_{d'}$.
- **Degree B Polynomial Basis:** We include all single features x_d, and all unique products of between 2 and B features.
- The problem is that the space complexity of representing the expanded set of features is essentially $O(D^B)$.
- Next we'll see how this problem can be solved.

Outline

Representer Theorem

- One of the interesting properties of SVMs is that the optimal weight vectors can always be expressed as a weighted linear combination of the data vectors:

$$\mathbf{w} = \sum_{j=1}^{N} \alpha_j \mathbf{x}_j$$

Representer Theorem

- One of the interesting properties of SVMs is that the optimal weight vectors can always be expressed as a weighted linear combination of the data vectors:

$$\mathbf{w} = \sum_{j=1}^{N} \alpha_j \mathbf{x}_j$$

- This result is called the *representer theorem*.

Dependence on Inner Products

- Plugging this result back in to the SVM objective we find that the objective only depends on the data through inner products: $\mathbf{x}_j^T\mathbf{x}_i$:

$$g(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i + b = \sum_{j=1}^{N} \alpha_j \mathbf{x}_j^T\mathbf{x}_i + b$$

$$||\mathbf{w}||_2^2 = \mathbf{w}^T\mathbf{w} = \sum_{j=1}^{N}\sum_{i=1}^{N} \alpha_j\alpha_i \mathbf{x}_j^T\mathbf{x}_i$$

Basis Expansion and Representer Theorem

- Under an arbitrary basis expansion $\phi(\mathbf{x})$ this result becomes:

$$g(\phi(\mathbf{x}_i)) = \sum_{j=1}^{N} \alpha_j \phi(\mathbf{x}_j)^T \phi(\mathbf{x}_i) + b$$

$$||\mathbf{w}||_2^2 = \sum_{j=1}^{N} \sum_{i=1}^{N} \alpha_j \alpha_i \phi(\mathbf{x}_j)^T \phi(\mathbf{x}_i)$$

- It can be shown in the linearly separable case that the α_i parameters for data cases that are not support vectors are always 0.

The Kernel Trick

- Amazingly, for many useful basis function expansions $\phi(\mathbf{x})$, it is possible to find a function $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T\phi(\mathbf{x}')$ that can compute the inner product under the basis expansion **without ever explicitly performing the basis function expansion**!

The Kernel Trick

- Amazingly, for many useful basis function expansions $\phi(\mathbf{x})$, it is possible to find a function $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T\phi(\mathbf{x}')$ that can compute the inner product under the basis expansion **without ever explicitly performing the basis function expansion**!
- Such functions are called *kernel functions* and this is known as the "Kernel Trick".

The Kernel Trick

- Amazingly, for many useful basis function expansions $\phi(\mathbf{x})$, it is possible to find a function $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$ that can compute the inner product under the basis expansion **without ever explicitly performing the basis function expansion**!
- Such functions are called *kernel functions* and this is known as the “Kernel Trick”.
- Importantly, you can also directly learn the parameters α_i and b using the kernel trick, without constructing the basis expansion.

The Kernel Trick

- Amazingly, for many useful basis function expansions $\phi(\mathbf{x})$, it is possible to find a function $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T\phi(\mathbf{x}')$ that can compute the inner product under the basis expansion **without ever explicitly performing the basis function expansion**!
- Such functions are called *kernel functions* and this is known as the “Kernel Trick”.
- Importantly, you can also directly learn the parameters α_i and b using the kernel trick, without constructing the basis expansion.
- Interestingly, there exist kernels for which the basis function expansion implied by the kernel isn’t even finite dimensional!

Examples of Kernel Functions

- **Degree B Polynomial Kernel:** $\mathcal{K}_P(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + 1)^B$

Examples of Kernel Functions

- **Degree B Polynomial Kernel:** $\mathcal{K}_P(\mathbf{x},\mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + 1)^B$
- **Gaussian/RBF Kernel:** $\mathcal{K}_G(\mathbf{x},\mathbf{x}') = \exp(-\gamma||\mathbf{x}-\mathbf{x}'||_2^2)$

Examples of Kernel Functions

- **Degree B Polynomial Kernel:** $\mathcal{K}_P(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + 1)^B$
- **Gaussian/RBF Kernel:** $\mathcal{K}_G(\mathbf{x}, \mathbf{x}') = \exp(-\gamma||\mathbf{x} - \mathbf{x}'||_2^2)$
- Many more domain-specific kernels for strings, histograms, probability distributions, and other complex structured objects.

Trade-Offs: Basis Expansion and Kernels

- Linear classifiers are fast and space efficient, but can have high bias.

Trade-Offs: Basis Expansion and Kernels

- Linear classifiers are fast and space efficient, but can have high bias.
- Basis expansion requires more space ($O(NK)$ for data and $O(K)$ for parameters), but yields non-linear classifiers that have lower bias.

Trade-Offs: Basis Expansion and Kernels

- Linear classifiers are fast and space efficient, but can have high bias.
- Basis expansion requires more space ($O(NK)$ for data and $O(K)$ for parameters), but yields non-linear classifiers that have lower bias.
- Kernel SVMs actually require $O(N^2)$ space for storing all the kernel values during training and have $O(N)$ parameters. This can still be much lower than $O(NK)$ for large sets of basis functions. Kernels also yield non-linear classifiers that have lower bias.

Trade-Offs: Basis Expansion and Kernels

- Linear classifiers are fast and space efficient, but can have high bias.
- Basis expansion requires more space ($O(NK)$ for data and $O(K)$ for parameters), but yields non-linear classifiers that have lower bias.
- Kernel SVMs actually require $O(N^2)$ space for storing all the kernel values during training and have $O(N)$ parameters. This can still be much lower than $O(NK)$ for large sets of basis functions. Kernels also yield non-linear classifiers that have lower bias.
- Kernel SVMs often have at least two parameters (C and a kernel hyperparameter). These need to be set jointly, which can be computationally expensive.

Trade-Offs: Basis Expansion and Kernels

- Gaussian kernel SVMs also have infinite capacity, and a very closely related to weighted KNN.

Trade-Offs: Basis Expansion and Kernels

- Gaussian kernel SVMs also have infinite capacity, and a very closely related to weighted KNN.
- Importantly, everything we said about SVMs and kernels is also true for logistic regression. Applying the kernel trick to logistic regression yields a model called “kernel logistic regression” or KLR.

Trade-Offs: Basis Expansion and Kernels

- Gaussian kernel SVMs also have infinite capacity, and a very closely related to weighted KNN.
- Importantly, everything we said about SVMs and kernels is also true for logistic regression. Applying the kernel trick to logistic regression yields a model called "kernel logistic regression" or KLR.
- KLR can exploit infinite dimensional feature spaces, can be learned with smooth optimization methods, supports probabilistic outputs, and has an easy multi-class generalization, but lacks the margin maximization property.