E0 270 Machine Learning **Lecture 6 (Jan 29, 2015)**

Kernel Methods

Lecturer: Shivani Agarwal

Disclaimer: These notes are a *brief* summary of the topics covered in the lecture. They are not a substitute for the full lecture.

Outline

- Non-linear models via basis functions
- Closer look at the SVM dual: kernel functions, kernel SVM
- RKHSs and Representer Theorem
- Kernel logistic regression
- Kernel ridge regression

1 Non-linear Models via Basis Functions

Let $\mathcal{X} = \mathbb{R}^d$. We have seen methods for learning linear models of the form $h(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$ for binary classification (such as logistic regression and SVMs) and $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ for regression (such as linear least squares regression and SVR); we have also seen methods for learning linear models for multiclass classification, of the form $h(\mathbf{x}) \in \text{argmax}_{y \in [r]} \mathbf{w}_y^\top \mathbf{x} + b_y$ (such as multiclass logistic regression). What if we want to learn a *non-linear* model? What would be a simple way to achieve this using the methods we have seen so far?

One way to achieve this is to map instances $\mathbf{x} \in \mathbb{R}^d$ to some new feature vectors $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^n$ via some non-linear feature mapping $\phi : \mathbb{R}^d \to \mathbb{R}^n$, and then to a linear model in this transformed space. For example, if one maps instances $\mathbf{x} \in \mathbb{R}^d$ to $n = (1 + 2d + \binom{d}{2})$-dimensional feature vectors

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_d \\ x_1 x_2 \\ \vdots \\ x_{d-1} x_d \\ x_1^2 \\ \vdots \\ x_d^2 \end{pmatrix},$$

then learning a linear model in the transformed space is equivalent to learning a quadratic model in the original instance space. In general, one can choose any *basis functions* $\phi_1, \ldots \phi_n : \mathcal{X} \to \mathbb{R}$, and learn a linear model over these: $\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) + b$, where $\mathbf{w} \in \mathbb{R}^n$ (in fact, one can do this for $\mathcal{X} \neq \mathbb{R}^d$ as well). For example, in least squares regression applied to a training sample $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)) \in (\mathbb{R}^d \times \mathbb{R})^m$, one would simply replace the matrix $\mathbf{X} \in \mathbb{R}^{m \times d}$ with the *design matrix* $\boldsymbol{\Phi} \in \mathbb{R}^{m \times n}$, where $\boldsymbol{\Phi}_{ij} = \phi_j(\mathbf{x}_i)$. What is a potential difficulty in doing this?

If n is large (e.g. as would be the case if the feature mapping $\boldsymbol{\phi}$ corresponded to a high-degree polynomial), then the above approach can be computationally expensive. In this lecture we look at a technique that allows one to implement the above idea efficiently for many algorithms. We start by taking a closer look at the SVM dual which we derived in the last lecture.

2 Closer Look at the SVM Dual: Kernel Functions, Kernel SVM

Recall the form of the dual we derived for the (soft-margin) linear SVM:

$$\max_{\boldsymbol{\alpha}} \quad -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j (\mathbf{x}_i^\top \mathbf{x}_j) + \sum_{i=1}^{m}\alpha_i \tag{1}$$

subject to

$$\sum_{i=1}^{m}\alpha_i y_i = 0 \tag{2}$$

$$0 \leq \alpha_i \leq C\,, \quad i = 1,\ldots,m. \tag{3}$$

If we implement this on feature vectors $\boldsymbol{\phi}(\mathbf{x}_i) \in \mathbb{R}^n$ in place of $\mathbf{x}_i \in \mathbb{R}^d$, we get the following optimization problem:

$$\max_{\boldsymbol{\alpha}} \quad -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \big(\boldsymbol{\phi}(\mathbf{x}_i)^\top \boldsymbol{\phi}(\mathbf{x}_j)\big) + \sum_{i=1}^{m}\alpha_i \tag{4}$$

subject to

$$\sum_{i=1}^{m}\alpha_i y_i = 0 \tag{5}$$

$$0 \leq \alpha_i \leq C\,, \quad i = 1,\ldots,m. \tag{6}$$

This involves computing dot products between vectors $\boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j)$ in $\mathbb{R}^n$. Similarly, using the learned model to make predictions on a new test point $\mathbf{x} \in \mathbb{R}^d$ also involves computing dot products between vectors in $\mathbb{R}^n$:

$$h(\mathbf{x}) = \operatorname{sign}\left(\sum_{i\in \mathrm{SV}} \widehat{\alpha}_i y_i \boldsymbol{\phi}(\mathbf{x}_i)^\top \boldsymbol{\phi}(\mathbf{x}) + \widehat{b}\right).$$

For example, as we saw above, one can learn a quadratic classifier in $\mathcal{X} = \mathbb{R}^2$ by learning a linear classifier in $\boldsymbol{\phi}(\mathbb{R}^2) \subset \mathbb{R}^6$, where

$$\boldsymbol{\phi}\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ x_2^2 \end{pmatrix};$$

clearly, a straightforward approach to learning an SVM classifier in this space (and applying it to a new test point) will involve computing dot products in $\mathbb{R}^6$ (more generally, when learning a degree-q polynomial in $\mathbb{R}^d$, such a straightforward approach will involve computing dot products in $\mathbb{R}^n$ for $n = O(d^q)$).

Now, consider replacing dot products $\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$ in the above example with $K(\mathbf{x}, \mathbf{x}')$, where $\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^2$,

$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^2\,.$$

It can be verified (exercise!) that $K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}_K(\mathbf{x})^\top \boldsymbol{\phi}_K(\mathbf{x}')$, where

$$\boldsymbol{\phi}_K\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \sqrt{2}x_1 x_2 \\ x_1^2 \\ x_2^2 \end{pmatrix}.$$

Thus, using $K(\mathbf{x}, \mathbf{x}')$ above instead of $\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$ implicitly computes dot products in $\mathbb{R}^6$, with computation of dot products required only in $\mathbb{R}^2$!

In fact, one can use any symmetric, positive semi-definite *kernel* function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (also called a *Mercer* kernel function) in the SVM algorithm directly, even if the feature space implemented by the kernel function cannot be described explicitly. Any such kernel function yields a convex dual problem; if K is positive *definite*, then K also corresponds to inner products in some inner product space V (i.e. $K(x, x') = \langle \phi(x), \phi(x') \rangle$ for some $\phi : \mathcal{X} \to V$).

For Euclidean instance spaces $\mathcal{X} = \mathbb{R}^d$, examples of commonly used kernel functions include the *polynomial kernel* $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^q$,which results in learning a degree-q polynomial threshold classifier, and the *Gaussian kernel*, also known as the *radial basis function (RBF) kernel*, $K(\mathbf{x}, \mathbf{x}') = \exp\left(\frac{-\|\mathbf{x}-\mathbf{x}'\|_2^2}{2\sigma^2}\right)$ (where $\sigma > 0$ is a parameter of the kernel), which effectivey implements dot products in an infinite-dimensional inner product space; in both cases, evaluating the kernel $K(\mathbf{x}, \mathbf{x}')$ at any two points $\mathbf{x}, \mathbf{x}'$ requires only $O(d)$ computation time. Kernel functions can also be used for non-vectorial data ($\mathcal{X} \neq \mathbb{R}^d$); for example, kernel functions are often used to implicitly embed instance spaces containing strings, trees etc into an inner product space, and to implicitly learn a linear classifier in this space. Intuitively, it is helpful to think of kernel functions as capturing some sort of 'similarity' between pairs of instances in $\mathcal{X}$.

To summarize, given a training sample $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \{\pm 1\})^m$, in order to learn a kernel SVM classifier using a kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, one simply solves the kernel SVM dual given by

$$\max_{\boldsymbol{\alpha}} \quad -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i, x_j) + \sum_{i=1}^{m} \alpha_i \tag{7}$$

subject to

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{8}$$

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m, \tag{9}$$

and then predicts the label of a new instance $x \in \mathcal{X}$ according to

$$h(x) = \operatorname{sign}\left(\sum_{i \in \mathrm{SV}} \widehat{\alpha}_i y_i K(x_i, x) + \widehat{b} \right),$$

where

$$\widehat{b} = \frac{1}{|\mathrm{SV}_1|} \sum_{i \in \mathrm{SV}_1} \left(y_i - \sum_{j \in \mathrm{SV}} \widehat{\alpha}_j y_j K(x_i, x_j) \right).$$

3 RKHSs and Representer Theorem

Let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric positive definite kernel function. Let

$$\mathcal{F}_K^0 = \left\{ f : \mathcal{X} \to \mathbb{R} \;\middle|\; f(x) = \sum_{i=1}^{r} \alpha_i K(x_i, x) \text{ for some } r \in \mathbb{Z}_+, \alpha_i \in \mathbb{R}, x_i \in \mathcal{X} \right\}.$$

For $f, g \in \mathcal{F}_K^0$ with $f(x) = \sum_{i=1}^{r} \alpha_i K(x_i, x)$ and $g(x) = \sum_{j=1}^{s} \beta_j K(x'_j, x)$, define

$$\langle f, g \rangle_K = \sum_{i=1}^{r}\sum_{j=1}^{s} \alpha_i \beta_i K(x_i, x'_j) \tag{10}$$

$$\|f\|_K = \sqrt{\langle f, f \rangle_K}\,. \tag{11}$$

Let $\mathcal{F}_K$ be the completion of $\mathcal{F}_K^0$ under the metric induced by the above norm. Then $\mathcal{F}_K$ is called the *reproducing kernel Hibert space* (RKHS) associated with K.

Note that the SVM classifier learned using kernel K is of the form

$$h(x) = \text{sign}(f(x) + b)\,,$$

where $f(x) = \sum_{i \in \text{SV}} \widehat{\alpha}_i y_i K(x_i, x)$, i.e. where $f \in \mathcal{F}_K$.

In fact, consider the following optimization problem:

$$\min_{f \in \mathcal{F}_K, b \in \mathbb{R}} \quad \frac{1}{m} \sum_{i=1}^{m} \big(1 - y_i(f(x_i) + b)\big)_+ + \lambda \|f\|_K^2\,.$$

It turns out that the above SVM solution (with $C = \frac{1}{2\lambda m}$) is a solution to this problem, i.e. kernel SVM solution minimizes the RKHS-norm regularized hinge loss over all functions over the form $f(x) + b$ for $f \in \mathcal{F}_K, b \in \mathbb{R}$.

More generally, we have the following result:

Theorem 1 (Representer Theorem)**.** *Let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel function. Let $\mathcal{Y} \subseteq \mathbb{R}$. Let $S = ((x_1, y_1)), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$. Let $L : \mathbb{R}^m \times \mathcal{Y}^m \to \mathbb{R}$. Let $\Omega : \mathbb{R}_+ \to \mathbb{R}_+$ be a monotonically increasing function. Then for $\lambda > 0$, there is a solution to the optimization problem*

$$\min_{f \in \mathcal{F}_K, b \in \mathbb{R}} \quad L\Big(\big(f(x_1) + b, \ldots, f(x_m) + b\big), (y_1, \ldots, y_m)\Big) + \lambda\, \Omega(\|f\|_K^2)$$

of the form

$$\widehat{f}(x) = \sum_{i=1}^{m} \widehat{\alpha}_i K(x_i, x) \quad \textit{for some } \widehat{\alpha}_1, \ldots, \widehat{\alpha}_m \in \mathbb{R}.$$

If Ω is strictly increasing, then all solutions have this form.

The above result tells us that even if $\mathcal{F}_K$ is an infinite-dimensional space, any optimization problem resulting from minimizing a loss over a finite training sample regularized by some increasing function of the RKHS-norm is effectively a finite-dimensional optimization problem, and moreover, the solution to this problem can be written as a kernel expansion over the training points. In particular, minimizing any other loss over $\mathcal{F}_K$ (regularized by the RKHS-norm) will also yield a solution of this form!

Exercise. Show that linear functions $f : \mathbb{R}^d \to \mathbb{R}$ of the form $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ form an RKHS with linear kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ given by $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$ and with $\|f\|_K^2 = \|\mathbf{w}\|_2^2$.

4 Kernel Logistic Regression

Given a training sample $S \in (\mathcal{X} \times \{\pm 1\})^m$ and kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, the kernel logistic regression classifier is given by the solution to the following optimization problem:

$$\min_{f \in \mathcal{F}_K, b \in \mathbb{R}} \quad \frac{1}{m} \sum_{i=1}^{m} \ln\left(1 + e^{-y_i(f(x_i)+b)}\right) + \lambda \|f\|_K^2\,.$$

Since we know from the Representer Theorem that the solution has the form $\widehat{f}(x) = \sum_{i=1}^m \widehat{\alpha}_i K(x_i, x)$, we can write the above as an optimization problem over $\boldsymbol{\alpha}, b$:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^m, b \in \mathbb{R}} \quad \frac{1}{m} \sum_{i=1}^{m} \ln\left(1 + e^{-y_i(\sum_{j=1}^m \alpha_j K(x_j, x_i)+b)}\right) + \lambda \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j K(x_i, x_j)\,.$$

This is of a similar form as in standard logistic regression, with m basis functions $\phi_j(x) = K(x_j, x)$ for $j \in [m]$ (and $\mathbf{w} \equiv \boldsymbol{\alpha}$)! In particular, define $\mathbf{K} \in \mathbb{R}^{m \times m}$ as $K_{ij} = K(x_1, x_j)$ (this is often called the *gram matrix*), and let $\mathbf{k}_i$ denote the i-th column of this matrix. Then we can write the above as simply

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^m, b \in \mathbb{R}} \quad \frac{1}{m} \sum_{i=1}^{m} \ln\left(1 + e^{-y_i(\boldsymbol{\alpha}^\top \mathbf{k}_i + b)}\right) + \lambda \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}\,,$$

which is similar to the form for standard linear logistic regression (with feature vectors $\mathbf{k}_i$) – except for the regularizer being $\boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha}$ rather than $\|\boldsymbol{\alpha}\|_2^2$ – and can be solved similarly as before, using similar numerical optimization methods.

We note that unlike SVMs, here in general, the solution has $\widehat{\alpha}_i \neq 0 \; \forall i \in [m]$. A variant of logistic regression called the *import vector machine* (IVM) adopts a greedy approach to find a subset IV $\subseteq [m]$ such that the function

$$\widehat{f}'(x) + \widehat{b} = \sum_{i \in \mathrm{IV}} \widehat{\alpha}_i K(x_i, x) + \widehat{b}$$

gives good performance. Compared to SVMs, IVMs can provide more natural class probability estimates, as well as more natural extensions to multiclass classification.

5 Kernel Ridge Regression

Given a training sample $S \in (\mathcal{X} \times \mathbb{R})^m$ and kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, consider first a kernel ridge regression formulation for learning a function $f \in \mathcal{F}_K$:

$$\min_{f \in \mathcal{F}_K} \quad \frac{1}{m} \sum_{i=1}^m \big(y_i - f(x_i)\big)^2 + \lambda \|f\|_K^2 \,.$$

Again, since we know from the Representer Theorem that the solution has the form $\widehat{f}(x) = \sum_{i=1}^m \widehat{\alpha}_i K(x_i, x)$, we can write the above as an optimization problem over $\boldsymbol{\alpha}$:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^m} \quad \frac{1}{m} \sum_{i=1}^m \left(y_i - \sum_{j=1}^m \alpha_j K(x_j, x_i)\right)^2 + \lambda \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j K(x_i, x_j) \,,$$

or in matrix notation,

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^m} \quad \frac{1}{m} \sum_{i=1}^m \big(y_i - \boldsymbol{\alpha}^\top \mathbf{k}_i\big)^2 + \lambda \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha} \,.$$

Again, this is of the same form as standard linear ridge regression, with feature vectors $\mathbf{k}_i$ and with regularizer $\boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha}$ rather than $\|\boldsymbol{\alpha}\|_2^2$. If K is positive definite, in which case the gram matrix $\mathbf{K}$ is invertible, then setting the gradient of the objective above w.r.t. $\boldsymbol{\alpha}$ to zero can be seen to yield

$$\widehat{\boldsymbol{\alpha}} = \big(\mathbf{K} + \lambda m \mathbf{I}_m\big)^{-1} \mathbf{y} \,,$$

where as before $\mathbf{I}_m$ is the $m \times m$ identity matrix and $\mathbf{y} = (y_1, \ldots, y_m)^\top \in \mathbb{R}^m$.

Exercise. Show that if $\mathcal{X} = \mathbb{R}^d$ and one wants to explicitly include a bias term b in the linear ridge regression solution which is not included in the regularization, then defining

$$\widetilde{\mathbf{X}} = \begin{bmatrix} \mathbf{x}_1^\top & 1 \\ \vdots & \vdots \\ \mathbf{x}_m^\top & 1 \end{bmatrix} , \quad \widetilde{\mathbf{w}} = \begin{pmatrix} \mathbf{w} \\ b \end{pmatrix} , \quad \mathbf{L} = \begin{bmatrix} \mathbf{I}_d & 0 \\ 0 & 0 \end{bmatrix} ,$$

one gets the solution

$$\widehat{\widetilde{\mathbf{w}}} = (\widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}} + \lambda m \mathbf{L})^{-1} \widetilde{\mathbf{X}}^\top \mathbf{y} \,.$$

How would you extend this to learning a function of the form $f(x) + b$ for $f \in \mathcal{F}_K, b \in \mathbb{R}$ in the kernel ridge regression setting?