

## 4.2 Kernel SVM

### *Machine Learning 1: Foundations*

Marius Kloft (TUK)

1 Kernel Methods

2 Kernel SVM

# Recap

Kernel methods:

- ▶ Use classifiers of the form

$$f(\mathbf{x}) = \text{sign}(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b).$$

# Recap

Kernel methods:

- ▶ Use classifiers of the form

$$f(\mathbf{x}) = \text{sign}(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b).$$

Kernel trick

- 1 Avoid computing  $\phi(\mathbf{x})$  explicitly—instead formulate learning machine in a way that it accesses the data only through inner products  $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$
- 2 Replace all occurrences of  $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$  by  $k(\mathbf{x}_i, \mathbf{x}_j)$ .

# Recap

Kernel methods:

- ▶ Use classifiers of the form

$$f(\mathbf{x}) = \text{sign}(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b).$$

Kernel trick

- 1 Avoid computing  $\phi(\mathbf{x})$  explicitly—instead formulate learning machine in a way that it accesses the data only through inner products  $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$
- 2 Replace all occurrences of  $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$  by  $k(\mathbf{x}_i, \mathbf{x}_j)$ .

Next:

Example of kernel trick applied to SVM.

# Example: SVM

Recall:

## Unconstrained linear soft-margin SVM

$$\min_{b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b))$$

# Example: SVM

Recall:

## Unconstrained linear soft-margin SVM

$$\min_{b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \max\left(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\right)$$

## Proposition (representer theorem for SVM)

The optimal solution  $\mathbf{w}^*$  of the above SVM satisfies:

$$\exists \alpha : \mathbf{w}^* = \sum_{i=1}^n \alpha_i \mathbf{x}_i,$$

where  $X := (\mathbf{x}_1, \dots, \mathbf{x}_n)$ .

# Example: SVM

Recall:

## Unconstrained linear soft-margin SVM

$$\min_{b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \max\left(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\right)$$

## Proposition (representer theorem for SVM)

The optimal solution  $\mathbf{w}^*$  of the above SVM satisfies:

$$\exists \alpha : \mathbf{w}^* = \sum_{i=1}^n \alpha_i \mathbf{x}_i,$$

where  $X := (\mathbf{x}_1, \dots, \mathbf{x}_n)$ .

We omit the proof for now, but the theorem is very intuitive:

- It says that  $\mathbf{w}^* \in \text{span}(\mathbf{x}_1, \dots, \mathbf{x}_n)$ .



# Example: SVM

Recall:

## Unconstrained linear soft-margin SVM

$$\min_{b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \max\left(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\right)$$

## Proposition (representer theorem for SVM)

The optimal solution  $\mathbf{w}^*$  of the above SVM satisfies:

$$\exists \alpha : \mathbf{w}^* = \sum_{i=1}^n \alpha_i \mathbf{x}_i,$$

where  $X := (\mathbf{x}_1, \dots, \mathbf{x}_n)$ .

We omit the proof for now, but the theorem is very intuitive:

- It says that  $\mathbf{w}^* \in \text{span}(\mathbf{x}_1, \dots, \mathbf{x}_n)$ .

We now plug  $\mathbf{w} = \sum_{i=1}^n \alpha_i \mathbf{x}_i$  into the SVM ...

# Kernel SVM

...and obtain:

$$\min_{b \in \mathbb{R}, \alpha \in \mathbb{R}^n} \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j \mathbf{x}_i^\top \mathbf{x}_j + C \sum_{i=1}^n \max \left( 0, 1 - y_i \left( \sum_{j=1}^n \alpha_j \mathbf{x}_i^\top \mathbf{x}_j + b \right) \right).$$

# Kernel SVM

...and obtain:

$$\min_{b \in \mathbb{R}, \alpha \in \mathbb{R}^n} \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j \boxed{\mathbf{x}_i^\top \mathbf{x}_j} + C \sum_{i=1}^n \max \left( 0, 1 - y_i \left( \sum_{j=1}^n \alpha_j \boxed{\mathbf{x}_i^\top \mathbf{x}_j} + b \right) \right).$$

We then replace all occurrences of  $\mathbf{x}_i^\top \mathbf{x}_j$  by  $k(\mathbf{x}_i, \mathbf{x}_j)$ :

## Kernel SVM

$$\min_{b \in \mathbb{R}, \alpha \in \mathbb{R}^n} \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) + C \sum_{i=1}^n \max \left( 0, 1 - y_i \left( \sum_{j=1}^n \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) + b \right) \right)$$

# Kernel SVM

...and obtain:

$$\min_{b \in \mathbb{R}, \alpha \in \mathbb{R}^n} \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j \mathbf{x}_i^\top \mathbf{x}_j + C \sum_{i=1}^n \max \left( 0, 1 - y_i \left( \sum_{j=1}^n \alpha_j \mathbf{x}_i^\top \mathbf{x}_j + b \right) \right).$$

We then replace all occurrences of  $\mathbf{x}_i^\top \mathbf{x}_j$  by  $k(\mathbf{x}_i, \mathbf{x}_j)$ :

## Kernel SVM

$$\min_{b \in \mathbb{R}, \alpha \in \mathbb{R}^n} \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) + C \sum_{i=1}^n \max \left( 0, 1 - y_i \left( \sum_{j=1}^n \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) + b \right) \right)$$

We predict the label of a new point  $\mathbf{x}$  using:

$$f(\mathbf{x}) = \text{sign} \left( \mathbf{w}^\top \phi(\mathbf{x}) + b \right) = \text{sign} \left( \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b \right).$$

# Kernel SVM

...and obtain:

$$\min_{b \in \mathbb{R}, \alpha \in \mathbb{R}^n} \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j \mathbf{x}_i^\top \mathbf{x}_j + C \sum_{i=1}^n \max \left( 0, 1 - y_i \left( \sum_{j=1}^n \alpha_j \mathbf{x}_i^\top \mathbf{x}_j + b \right) \right).$$

We then replace all occurrences of  $\mathbf{x}_i^\top \mathbf{x}_j$  by  $k(\mathbf{x}_i, \mathbf{x}_j)$ :

## Kernel SVM

$$\min_{b \in \mathbb{R}, \alpha \in \mathbb{R}^n} \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) + C \sum_{i=1}^n \max \left( 0, 1 - y_i \left( \sum_{j=1}^n \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) + b \right) \right)$$

We predict the label of a new point  $\mathbf{x}$  using:

$$f(\mathbf{x}) = \text{sign} \left( \mathbf{w}^\top \phi(\mathbf{x}) + b \right) = \text{sign} \left( \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b \right).$$

How to solve the kernel SVM?

# Kernel SVM Optimization

Could apply CVXOPT as Kernel SVM is convex (but too slow!)

Instead we apply a SGD sort of algorithm as follows:

## Doubly SGD algorithm for kernel SVM

- 1: initialize  $(b, \alpha)$  (e.g., randomly)
- 2: **for**  $t = 1 : T$  **do**
- 3:     Randomly select  $B$  many data points
- 4:     Denote their indexes by  $J \subset \{1, \dots, n\}$  (i.e.,  $|J| = B$ )
- 5:      $(b, \alpha) := (b, \alpha) - \lambda_t \nabla_{\alpha_J} \left( \frac{n^2}{2B^2} \sum_{i,j \in J} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \right.$   
       $\left. + \frac{Cn}{B} \sum_{i \in J} \max \left( 0, 1 - y_i \left( \frac{n}{B} \sum_{j \in J} \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) + b \right) \right) \right)$
- 6: **end for**

FYI: an analogue derivation and algorithm can be stated for LR.

---

For instance, we can choose  $B = 100$ .

# Can We Kernelize Also Other Linear Learning Machines?

---

The general representer theorem holds in particular also for the SVM. Thus the proof here serves also as a proof for the proposition shown on Slide 3.

# Can We Kernelize Also Other Linear Learning Machines?

Yes:

## Theorem (general representer theorem)

Let  $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$  be a kernel over a Hilbert space  $\mathcal{H}$ . Let  $L : \mathbb{R}^n \rightarrow \mathbb{R}$  be any function. Then any solution

$$\mathbf{w}^* \in \arg \min_{\mathbf{w} \in \mathcal{H}} \frac{1}{2} \|\mathbf{w}\|^2 + L(\langle \mathbf{w}, \phi(\mathbf{x}_1) \rangle, \dots, \langle \mathbf{w}, \phi(\mathbf{x}_n) \rangle)$$

satisfies: there exist  $\alpha_1, \dots, \alpha_n$  such that  $\mathbf{w}^* = \sum_{i=1}^n \alpha_i \phi(\mathbf{x}_i)$ .

---

The general representer theorem holds in particular also for the SVM. Thus the proof here serves also as a proof for the proposition shown on Slide 3.



# Proof

## Proof:

Assume the contrary:

$\exists \mathbf{w}_{\parallel} \in \text{span}(\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_2)), \mathbf{w}_{\perp} \notin \text{span}(\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_2)) :$   
 $\mathbf{w}^* = \mathbf{w}_{\parallel} + \mathbf{w}_{\perp}$  with  $\|\mathbf{w}_{\perp}\| > 0$ .

Then:

$$\langle \mathbf{w}^*, \phi(\mathbf{x}_i) \rangle = (\mathbf{w}_{\parallel} + \mathbf{w}_{\perp})^T \phi(\mathbf{x}_i) = \mathbf{w}_{\parallel}^T \phi(\mathbf{x}_i) + \underbrace{\mathbf{w}_{\perp}^T \phi(\mathbf{x}_i)}_{=0} = \mathbf{w}_{\parallel}^T \phi(\mathbf{x}_i)$$

Also:

$$\|\mathbf{w}^*\|^2 = \|\mathbf{w}_{\parallel}\|^2 + \underbrace{\|\mathbf{w}_{\perp}\|^2}_{>0}$$

which contradicts the optimality of  $\mathbf{w}^*$ .  $\square$

# Conclusion

Kernel methods:

- ▶ Use linear classifiers on the data mapped into a high-dimensional space

# Conclusion

Kernel methods:

- ▶ Use linear classifiers on the data mapped into a high-dimensional space

Kernel trick:

- ▶ But never compute explicitly in that space
- ▶ requires formulating the learning machine in terms of inner products
  - ▶ which are then replaced by the kernel

# Conclusion

Kernel methods:

- ▶ Use linear classifiers on the data mapped into a high-dimensional space

Kernel trick:

- ▶ But never compute explicitly in that space
- ▶ requires formulating the learning machine in terms of inner products
  - ▶ which are then replaced by the kernel

Representer theorem:

- ▶ this works for many linear learning machines
  - ▶ Example: SVM

# Reading

- ▶ Klaus-Robert Müller et al.: An Introduction to Kernel-based Learning Algorithms. IEEE Transactions on Neural Networks, 12(2), 2001.

<http://axon.cs.byu.edu/~martinez/classes/678/Papers/MullerKernel.pdf>

# Refs I



K.-R. Müller, S. Mika, G. Rätsch, S. Tsuda, and B Schölkopf, An introduction to kernel-based learning algorithms, *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 181–202, 2001.

## Further Information (non-mandatory class content)

For the Gaussian RBF kernel, we have a more efficient way of training kernel machines (such as the kernel SVM)...

It is based on the following Theorem:



For the Gaussian RBF kernel, we have a more efficient way of training kernel machines (such as the kernel SVM)...

It is based on the following Theorem:

## Theorem (Bochner's Theorem)

The Gaussian RBF kernel  $k$  with bandwidth  $\sigma^2 > 0$  satisfies the following identity:

$$k(\mathbf{x}, \tilde{\mathbf{x}}) = 2\mathbb{E}_{\substack{\mathbf{w} \sim N(0, I_d) \\ b \sim U(0, 2\pi)}} \left[ \cos\left(\frac{\mathbf{w}^\top \mathbf{x}}{\sigma} + b\right) \cos\left(\frac{\mathbf{w}^\top \tilde{\mathbf{x}}}{\sigma} + b\right) \right]$$

Remark: we employ here the following notation.

- ▶  $N(0, I_d)$  is the  $d$ -dimensional standard normal distribution.
- ▶  $U(0, 2\pi)$  is a uniform distribution on the interval  $[0, 2\pi]$ .

## Random features computation algorithm

- 1: **for**  $i = 1 : m$  **do**
- 2:     sample  $\mathbf{w}_i \sim N(0, I_d)$
- 3:     sample  $b_i \sim U(0, 2\pi)$
- 4: **end for**
- 5: Define

$$\hat{\phi}_m : \mathbb{R}^d \rightarrow \mathbb{R}^m$$

$$\mathbf{x} \mapsto \sqrt{\frac{2}{n}} \left( \cos \left( \frac{\mathbf{w}_1^\top \mathbf{x}}{\sigma} + b_1 \right), \dots, \cos \left( \frac{\mathbf{w}_m^\top \mathbf{x}}{\sigma} + b_m \right) \right)^\top$$

- 6: **return**  $\hat{\phi}_m$

## Corollary

Let  $\hat{\phi}_m$  be the output of the above algorithm and  $k$  the Gaussian RBF kernel with bandwidth  $\sigma^2 > 0$ . Then:

$$\forall \mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^d : \left\langle \hat{\phi}_m(\mathbf{x}), \hat{\phi}_m(\tilde{\mathbf{x}}) \right\rangle \xrightarrow{m \rightarrow \infty} k(\mathbf{x}, \tilde{\mathbf{x}})$$

This results in a random-feature SVM training algorithm:

- ▶ Compute the map  $\phi_m$  using the algorithm from the previous slide
- ▶ Apply the map to all training points, resulting in a dataset  $\hat{\phi}_m(X) := \{\hat{\phi}_m(\mathbf{x}_1), \dots, \hat{\phi}_m(\mathbf{x}_n)\}$
- ▶ Train a standard linear SVM solver on  $\hat{\phi}_m(X)$  and  $\mathbf{y}$  (e.g., LINLINEAR or Vowpal Wabbit)

This results in a random-feature SVM training algorithm:

- ▶ Compute the map  $\phi_m$  using the algorithm from the previous slide
- ▶ Apply the map to all training points, resulting in a dataset  $\hat{\phi}_m(X) := \{\hat{\phi}_m(\mathbf{x}_1), \dots, \hat{\phi}_m(\mathbf{x}_n)\}$
- ▶ Train a standard linear SVM solver on  $\hat{\phi}_m(X)$  and  $\mathbf{y}$  (e.g., LINLINEAR or Vowpal Wabbit)

This means we can use a super-fast linear SVM solver to train a non-linear learning machine (Gaussian-RBF-kernel SVM)!

Works more generally for all linear learning machines that can be kernelized (e.g., logistic regression).