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

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

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

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

Example: SVM

Recall:

Unconstrained linear soft-margin SVM

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

Proposition (representer theorem for SVM)

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

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

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

Example: SVM

Recall:

Unconstrained linear soft-margin SVM

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

Proposition (representer theorem for SVM)

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

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

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

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

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

Example: SVM

Recall:

Unconstrained linear soft-margin SVM

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

Proposition (representer theorem for SVM)

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

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

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

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

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

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

Kernel SVM

...and obtain:

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

Kernel SVM

...and obtain:

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

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

Kernel SVM

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

Kernel SVM

...and obtain:

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

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

Kernel SVM

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

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

$$f(\mathbf{x}) = \operatorname{sign}\left(\mathbf{w}^\top \phi(\mathbf{x}) + b\right) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i y_i \overset{k(x_i, x)}{\cancel{\mathbf{x}_i^\top \mathbf{x}}} + b\right).$$

Kernel SVM

...and obtain:

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

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

Kernel SVM

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

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

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

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 $(b, \boldsymbol{\alpha})$ (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: $$(b, \boldsymbol{\alpha}) := (b, \boldsymbol{\alpha}) - \lambda_t \nabla_{\boldsymbol{\alpha}_J} \left(\frac{n^2}{2B^2} \sum_{i,j \in J} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) + \frac{Cn}{B} \sum_{i \in J} \max\left(0, 1 - y_i \left(\frac{n}{B} \sum_{j \in J} \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) + b \right)\right)\right)$$
6: **end for**

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

LIBSVM

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

$$\mathbf{w}^* \in \operatorname*{arg\,min}_{\mathbf{w} \in \mathcal{H}} \overbrace{\frac{1}{2}\|\mathbf{w}\|^2 + L(\langle \mathbf{w}, \phi(\mathbf{x}_1)\rangle, \ldots, \langle \mathbf{w}, \phi(\mathbf{x}_n)\rangle)}^{=: g(w)}$$

satisfies: there exist $\alpha_1, \ldots, \alpha_n$ 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

Assume the contrary, that is, ~~there~~ exists a solution w^* $= \underbrace{w_\parallel}_{\in \operatorname{span}(x_1,\dots,x_n)\ (**)} + \underbrace{w_\perp}_{(*)}$ with $w_\perp \neq 0$

$\Rightarrow \|w_\perp\| > 0.$

$$g(w^*) = \frac{1}{2}\underbrace{\|w^*\|^2}_{= \|w_\parallel\|^2 + \underbrace{\|w_\perp\|^2}_{>0}} + L\big(\underbrace{\langle w^*, \phi(x_1)\rangle}, \dots, \underbrace{\langle w^*, \phi(x_n)\rangle}\big)$$

$$= \langle w_\parallel + w_\perp, \phi(x_i)\rangle \overset{(**)}{=} \langle w_\parallel, \phi(x_i)\rangle + \underbrace{\langle w_\perp, \phi(x_i)\rangle}_{} = \langle w_\parallel, \phi(x_i)\rangle \quad =0$$

$$> g(w_\parallel) \quad \lightning$$

$\square$

(*) $\forall\, x \in \operatorname{span}(x_1, \dots, x_n): \langle w_\perp, x \rangle = 0$

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 explicitely 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 explicitely 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.

Appendix

Further Information
(non-mandatory class content)

Random Features

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:

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

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

$$\hat{\phi}_m : \begin{array}{ccc} \mathbb{R}^d & \to & \mathbb{R}^m \\ \mathbf{x} & \mapsto & \sqrt{\frac{2}{n}} \left(\cos\left(\frac{\mathbf{w}_1^\top \mathbf{x}}{\sigma} + b_1 \right), \ldots, \cos\left(\frac{\mathbf{w}_m^\top \mathbf{x}}{\sigma} + b_m \right) \right)^\top \end{array}$$

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:

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

Random Features 3/3

This results in a random-feature SVM training algorithm:

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

Random Features 3/3

This results in a random-feature SVM training algorithm:

- Compute the map ϕ_m using the algorithm from the previous slide
- Apply the map to all training points, resulting in a dataset $\hat{\phi}_m(X) := \left\{\hat{\phi}_m(\mathbf{x}_1), \ldots, \hat{\phi}_m(\mathbf{x}_n)\right\}$
- 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).