Deep-er Kernels

John Shawe-Taylor

Department of Computer Science
University College London

ROKS Meeting, July 2013

Joint work as referenced plus Dimitrios Athanasakis, Delmiro Fernandez-Reyes

Background

- Deep learning has (re-)emerged as having important research and commercial value
- Deep belief networks and related approaches have led this charge
- Kernels are sometimes refered to as 'shallow'
- Aim of this talk is to:
    - Discuss what we mean by deep learning
    - Describe a number of ways in which kernel learning has been made 'deeper'

Background

- Deep learning has (re-)emerged as having important research and commercial value
- Deep belief networks and related approaches have led this charge
- Kernels are sometimes refered to as 'shallow'
- Aim of this talk is to:
  - Discuss what we mean by deep learning
  - Describe a number of ways in which kernel learning has been made 'deeper'

Background

- Deep learning has (re-)emerged as having important research and commercial value
- Deep belief networks and related approaches have led this charge
- Kernels are sometimes refered to as 'shallow'
- Aim of this talk is to:
  - Discuss what we mean by deep learning
  - Describe a number of ways in which kernel learning has been made 'deeper'

Background

- Deep learning has (re-)emerged as having important research and commercial value
- Deep belief networks and related approaches have led this charge
- Kernels are sometimes refered to as ‘shallow’
- Aim of this talk is to:
  - Discuss what we mean by deep learning
  - Describe a number of ways in which kernel learning has been made ‘deeper’

Why Shallow Learning?

- Kernels learn non-linear functions in the input space so would appear to be as flexible as deep learning systems
- However, they actually implement linear functions in the kernel defined feature space:

$$\mathbf{x} \longmapsto_{\text{fixed}} \phi(\mathbf{x}) \longmapsto_{\text{learned}} \langle \mathbf{w}, \phi(\mathbf{x}) \rangle$$

  so that the learning (of $\mathbf{w}$) only occurs in one 'layer'.
- This is contrasted with deep learning where parameters are spread across several layers typically with non-linear transfer functions
  - Learning of the deeper layers is often unsupervised with the final classifier trained with the earlier layers fixed
  - Hence, we are effectively pre-learning a representation – this would be analogous to learning the kernel

Why Shallow Learning?

- Kernels learn non-linear functions in the input space so would appear to be as flexible as deep learning systems
- However, they actually implement linear functions in the kernel defined feature space:

  $$\mathbf{x} \longmapsto_{\text{fixed}} \phi(\mathbf{x}) \longmapsto_{\text{learned}} \langle \mathbf{w}, \phi(\mathbf{x}) \rangle$$

  so that the learning (of $\mathbf{w}$) only occurs in one 'layer'.
- This is contrasted with deep learning where parameters are spread across several layers typically with non-linear transfer functions
  - Learning of the deeper layers is often unsupervised with the final classifier trained with the earlier layers fixed
  - Hence, we are effectively pre-learning a representation – this would be analogous to learning the kernel

Why Shallow Learning?

- Kernels learn non-linear functions in the input space so would appear to be as flexible as deep learning systems
- However, they actually implement linear functions in the kernel defined feature space:

  $$\mathbf{x} \longmapsto_{\text{fixed}} \phi(\mathbf{x}) \longmapsto_{\text{learned}} \langle \mathbf{w}, \phi(\mathbf{x}) \rangle$$

  so that the learning (of $\mathbf{w}$) only occurs in one 'layer'.
- This is contrasted with deep learning where parameters are spread across several layers typically with non-linear transfer functions
  - Learning of the deeper layers is often unsupervised with the final classifier trained with the earlier layers fixed
  - Hence, we are effectively pre-learning a representation – this would be analogous to learning the kernel

What happens in practice?

- In practice we typically do perform some learning of the kernel:
  - fix some hyper-parameters via some heuristic (e.g. width σ of a Gaussian kernel)
  - use cross-validation to adapt the hyperparameter to optimise performance of the task (classification, regression, etc)
- In some respects this undermines the more principled approach espoused by kernel methods based on generalisation bounds:
  - standard generalisation bounds no longer apply if we choose the feature space based on the training data
  - even test set bounds will be invalidated if we include the testing data in the representation learning phase
- Often more sophisticated representations encode 'deep' prior knowledge, but are 'learned' by trial and error
  - for example the histograms of patch cluster presence used in an object detection system

What happens in practice?

- In practice we typically do perform some learning of the kernel:
  - fix some hyper-parameters via some heuristic (e.g. width σ of a Gaussian kernel)
  - use cross-validation to adapt the hyperparameter to optimise performance of the task (classification, regression, etc)
- In some respects this undermines the more principled approach espoused by kernel methods based on generalisation bounds:
  - standard generalisation bounds no longer apply if we choose the feature space based on the training data
  - even test set bounds will be invalidated if we include the testing data in the representation learning phase
- Often more sophisticated representations encode 'deep' prior knowledge, but are 'learned' by trial and error
  - for example the histograms of patch cluster presence used in an object detection system

What happens in practice?

- In practice we typically do perform some learning of the kernel:
  - fix some hyper-parameters via some heuristic (e.g. width σ of a Gaussian kernel)
  - use cross-validation to adapt the hyperparameter to optimise performance of the task (classification, regression, etc)
- In some respects this undermines the more principled approach espoused by kernel methods based on generalisation bounds:
  - standard generalisation bounds no longer apply if we choose the feature space based on the training data
  - even test set bounds will be invalidated if we include the testing data in the representation learning phase
- Often more sophisticated representations encode 'deep' prior knowledge, but are 'learned' by trial and error
  - for example the histograms of patch cluster presence used in an object detection system

Aim of this talk

- Present a number of promising directions that tick (some of) the following boxes:
  - Learn a (kernel) representation possibly tuned to the main learning task
  - Provide any analysis of the resulting system that supports its design and bounds its performance
  - Provide empirical evidence that supports the approach on real world data
- the different contributions may appear disjointed but I hope a convincing and coherent story will emerge:
  - deep-er learning of kernels is alive and kicking!

Aim of this talk

- Present a number of promising directions that tick (some of) the following boxes:
    - Learn a (kernel) representation possibly tuned to the main learning task
    - Provide any analysis of the resulting system that supports its design and bounds its performance
    - Provide empirical evidence that supports the approach on real world data
- the different contributions may appear disjointed but I hope a convincing and coherent story will emerge:
    - deep-er learning of kernels is alive and kicking!

Aim of this talk

- Present a number of promising directions that tick (some of) the following boxes:
  - Learn a (kernel) representation possibly tuned to the main learning task
  - Provide any analysis of the resulting system that supports its design and bounds its performance
  - Provide empirical evidence that supports the approach on real world data
- the different contributions may appear disjointed but I hope a convincing and coherent story will emerge:
  - deep-er learning of kernels is alive and kicking!

Aim of this talk

- Present a number of promising directions that tick (some of) the following boxes:
  - Learn a (kernel) representation possibly tuned to the main learning task
  - Provide any analysis of the resulting system that supports its design and bounds its performance
  - Provide empirical evidence that supports the approach on real world data
- the different contributions may appear disjointed but I hope a convincing and coherent story will emerge:
  - deep-er learning of kernels is alive and kicking!

Aim of this talk

- Present a number of promising directions that tick (some of) the following boxes:
  - Learn a (kernel) representation possibly tuned to the main learning task
  - Provide any analysis of the resulting system that supports its design and bounds its performance
  - Provide empirical evidence that supports the approach on real world data
- the different contributions may appear disjointed but I hope a convincing and coherent story will emerge:
  - deep-er learning of kernels is alive and kicking!

Aim of this talk

- Present a number of promising directions that tick (some of) the following boxes:
  - Learn a (kernel) representation possibly tuned to the main learning task
  - Provide any analysis of the resulting system that supports its design and bounds its performance
  - Provide empirical evidence that supports the approach on real world data
- the different contributions may appear disjointed but I hope a convincing and coherent story will emerge:
  - deep-er learning of kernels is alive and kicking!

Matching pursuit

- Matching pursuit greedily chooses training examples that determine directions in feature space that are well-suited to some task and then deflates
- Analysis combining sparse reconstruction with generalisation error bounds gives first bounds on performance in learnt subspace
- Allows different criteria for selection to be implemented in one framework, eg sparse PCA, classification, regression, canonical correlation analysis, etc. and all come with bounds

  ⋆ Hussain, Z., Shawe-Taylor, J., Hardoon, D.R. and Dhanjal, C (2011) Design and Generalization Analysis of Orthogonal Matching Pursuit Algorithms, IEEE Trans on Information Theory, 57, 5326–5341.

Matching pursuit

- Matching pursuit greedily chooses training examples that determine directions in feature space that are well-suited to some task and then deflates
- Analysis combining sparse reconstruction with generalisation error bounds gives first bounds on performance in learnt subspace
- Allows different criteria for selection to be implemented in one framework, eg sparse PCA, classification, regression, canonical correlation analysis, etc. and all come with bounds

⋆ Hussain, Z., Shawe-Taylor, J., Hardoon, D.R. and Dhanjal, C (2011) Design and Generalization Analysis of Orthogonal Matching Pursuit Algorithms, IEEE Trans on Information Theory, 57, 5326–5341.

Matching pursuit

- Matching pursuit greedily chooses training examples that determine directions in feature space that are well-suited to some task and then deflates
- Analysis combining sparse reconstruction with generalisation error bounds gives first bounds on performance in learnt subspace
- Allows different criteria for selection to be implemented in one framework, eg sparse PCA, classification, regression, canonical correlation analysis, etc. and all come with bounds

  ⋆ Hussain, Z., Shawe-Taylor, J., Hardoon, D.R. and Dhanjal, C (2011) Design and Generalization Analysis of Orthogonal Matching Pursuit Algorithms, IEEE Trans on Information Theory, 57, 5326–5341.

Matching pursuit bound plot

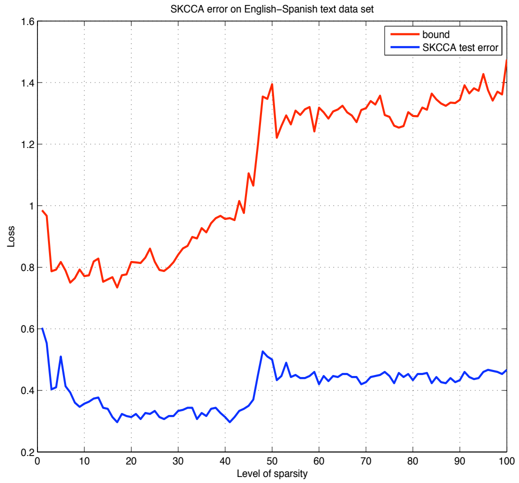


Figure : Bound plot for sparse KCCA using 1-dimension.

Kernels from Probabilistic Models

- If we consider learning a representation as pre-processing stage, it is natural to consider modelling the data with a probabilistic model
- There are then two main methods of defining kernels from probabilistic models:
  - Averaging over a model class - i.e. each model gives one feature:
  $$\kappa(x, z) = \sum_{m \in M} P(x|m)P(z|m)P_M(m)$$
  also known as the marginalisation kernel.
  - Fisher kernels for cases where the model is determined by a real parameter vector
- Give example of Fisher kernel

Kernels from Probabilistic Models

- If we consider learning a representation as pre-processing stage, it is natural to consider modelling the data with a probabilistic model
- There are then two main methods of defining kernels from probabilistic models:
  - Averaging over a model class - i.e. each model gives one feature:
    $$\kappa(x, z) = \sum_{m \in M} P(x|m)P(z|m)P_M(m)$$
    also known as the marginalisation kernel.
  - Fisher kernels for cases where the model is determined by a real parameter vector
- Give example of Fisher kernel

Kernels from Probabilistic Models

- If we consider learning a representation as pre-processing stage, it is natural to consider modelling the data with a probabilistic model
- There are then two main methods of defining kernels from probabilistic models:
  - Averaging over a model class - i.e. each model gives one feature:
  $$\kappa(x, z) = \sum_{m \in M} P(x|m)P(z|m)P_M(m)$$
  also known as the marginalisation kernel.
  - Fisher kernels for cases where the model is determined by a real parameter vector
- Give example of Fisher kernel

Fisher kernels

- We assume the model is parametrised according to some parameters: consider the simple example of a 1-dim Gaussian distribution parametrised by μ and σ:

$$M = \left\{ P(x|\theta) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right) : \theta = (\mu, \sigma) \in \mathbb{R}^2 \right\}.$$

- The Fisher score vector is the derivative of the log likelihood of an input x wrt the parameters:

$$\log \mathcal{L}_{(\mu,\sigma)}(x) = -\frac{(x-\mu)^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma).$$

Fisher kernels

- We assume the model is parametrised according to some parameters: consider the simple example of a 1-dim Gaussian distribution parametrised by μ and σ:

$$M = \left\{ P(x|\theta) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right) : \theta = (\mu, \sigma) \in \mathbb{R}^2 \right\}.$$

- The Fisher score vector is the derivative of the log likelihood of an input x wrt the parameters:

$$\log \mathcal{L}_{(\mu,\sigma)}(x) = -\frac{(x-\mu)^2}{2\sigma^2} - \frac{1}{2} \log(2\pi\sigma).$$

Fisher kernels

- Hence the score vector is given by:

$$\mathbf{g}\left(\theta^0, x\right) = \left(\frac{(x-\mu_0)}{\sigma_0^2}, \frac{(x-\mu_0)^2}{\sigma_0^3} - \frac{1}{2\sigma_0}\right).$$

- Taking $\mu_0 = 0$ and $\sigma_0 = 1$ the feature embedding is given by:

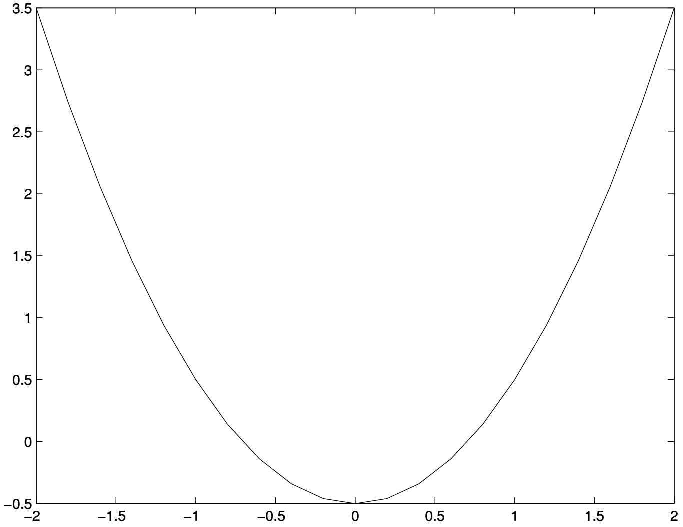
Fisher kernels

- Hence the score vector is given by:

$$\mathbf{g}\left(\theta^0, x\right) = \left(\frac{(x-\mu_0)}{\sigma_0^2}, \frac{(x-\mu_0)^2}{\sigma_0^3} - \frac{1}{2\sigma_0}\right).$$

- Taking $\mu_0 = 0$ and $\sigma_0 = 1$ the feature embedding is given by:

Fisher kernels



String kernels as Fisher kernels

- We can consider a Markov model of generating text conditioned on the previous n-characters
- Taking the uniform distribution model gives the class of string kernels - but these can now be learned based on a corpus
- can extend to probabilistic Finite State Automata learned from the corpus
- results competitive with tfidf BoWs on Reuters, with some improvements in average precision

⋆ C. Saunders, J. Shawe-Taylor and A. Vinokourov (2003) String Kernels, Fisher Kernels and Finite State Automata, NIPS 15.

String kernels as Fisher kernels

- We can consider a Markov model of generating text conditioned on the previous n-characters
- Taking the uniform distribution model gives the class of string kernels - but these can now be learned based on a corpus
- can extend to probabilistic Finite State Automata learned from the corpus
- results competitive with tfidf BoWs on Reuters, with some improvements in average precision

$\star$ C. Saunders, J. Shawe-Taylor and A. Vinokourov (2003) String Kernels, Fisher Kernels and Finite State Automata, NIPS 15.

String kernels as Fisher kernels

- We can consider a Markov model of generating text conditioned on the previous n-characters
- Taking the uniform distribution model gives the class of string kernels - but these can now be learned based on a corpus
- can extend to probabilistic Finite State Automata learned from the corpus
- results competitive with tfidf BoWs on Reuters, with some improvements in average precision

$\star$ C. Saunders, J. Shawe-Taylor and A. Vinokourov (2003) String Kernels, Fisher Kernels and Finite State Automata, NIPS 15.

String kernels as Fisher kernels

- We can consider a Markov model of generating text conditioned on the previous n-characters
- Taking the uniform distribution model gives the class of string kernels - but these can now be learned based on a corpus
- can extend to probabilistic Finite State Automata learned from the corpus
- results competitive with tfidf BoWs on Reuters, with some improvements in average precision

  ⋆ C. Saunders, J. Shawe-Taylor and A. Vinokourov (2003) String Kernels, Fisher Kernels and Finite State Automata, NIPS 15.

Multiple kernel learning

- MKL puts a 1-norm constraint on a linear combination of kernels:

$$\left\{ \kappa(\mathbf{x}, \mathbf{x}') = \sum_{t=1}^{N} z_t \kappa_t(\mathbf{x}, \mathbf{x}') : z_t \geq 0, \sum_{t=1}^{N} z_t = 1 \right\}$$

  and trains an SVM while optimizing z_t – a convex problem

- obtain corresponding bound (using convex hull bound for Rademacher complexity):

$$P(y \neq \operatorname{sgn}(g(\mathbf{x}))) \leq \frac{1}{m\gamma} \sum_{i=1}^{m} \xi_i + \frac{1}{\gamma} \hat{R}_m \left(\bigcup_{t=1}^{N} \mathcal{F}_t \right) + 3 \sqrt{\frac{\ln(2/\delta)}{2m}}$$

  where $\mathcal{F}_t = \{ \mathbf{x} \to \langle \mathbf{w}, \phi_{\mathbf{t}}(\mathbf{x}) \rangle : \|\mathbf{w}\| \leq 1 \}$.

Multiple kernel learning

- MKL puts a 1-norm constraint on a linear combination of kernels:

$$\left\{\kappa(\mathbf{x},\mathbf{x}') = \sum_{t=1}^{N} z_t \kappa_t(\mathbf{x},\mathbf{x}') : z_t \geq 0, \sum_{t=1}^{N} z_t = 1\right\}$$

  and trains an SVM while optimizing z_t – a convex problem
- obtain corresponding bound (using convex hull bound for Rademacher complexity):

$$\begin{aligned} &P(y \neq \operatorname{sgn}(g(\mathbf{x}))) \\ &\leq \frac{1}{m\gamma}\sum_{i=1}^{m}\xi_i + \frac{1}{\gamma}\hat{R}_m\left(\bigcup_{t=1}^{N}\mathcal{F}_t\right) + 3\sqrt{\frac{\ln(2/\delta)}{2m}} \end{aligned}$$

  where $\mathcal{F}_t = \{\mathbf{x} \to \langle \mathbf{w}, \phi_{\mathbf{t}}(\mathbf{x})\rangle : \|\mathbf{w}\| \leq 1\}$.

Rademacher complexity

The Rademacher complexity provides a way of measuring the complexity of a function class $\mathcal{F}$ by testing how well on average it can align with random noise:

$$\hat{R}_m(\mathcal{F}) = \mathbb{E}_\sigma \left[\sup_{f \in \mathcal{F}} \frac{2}{m} \sum_{i=1}^{m} \sigma_i f(\mathbf{x}_i) \right].$$

is known as the Rademacher complexity of the function class $\mathcal{F}$.

Bounding MKL

- Need a bound on $\hat{R}_m\left(\mathcal{F} = \bigcup_{t=1}^{N} \mathcal{F}_t\right)$
- McDiarmid gives with probability $1 - \delta_0$ of a random selection of σ^*:

$$\hat{R}_m(\mathcal{F}) \leq \frac{2}{m} \sup_{f \in \mathcal{F}} \sum_{i=1}^{m} \sigma_i^* f(\mathbf{x}_i) + 4\sqrt{\frac{\ln(1/\delta_t)}{2m}}$$

and $$\frac{2}{m} \sup_{f \in \mathcal{F}_t} \sum_{i=1}^{m} \sigma_i^* f(\mathbf{x}_i) \leq \hat{R}_m(\mathcal{F}_t) + 4\sqrt{\frac{\ln(1/\delta_t)}{2m}}$$

with probability $1 - \delta_t$

Bounding MKL

- Need a bound on $$\hat{R}_m\left(\mathcal{F} = \bigcup_{t=1}^{N} \mathcal{F}_t\right)$$
- McDiarmid gives with probability $1 - \delta_0$ of a random selection of σ^*:

$$\hat{R}_m(\mathcal{F}) \leq \frac{2}{m} \sup_{f \in \mathcal{F}} \sum_{i=1}^{m} \sigma_i^* f(\mathbf{x}_i) + 4\sqrt{\frac{\ln(1/\delta_t)}{2m}}$$

and $$\frac{2}{m} \sup_{f \in \mathcal{F}_t} \sum_{i=1}^{m} \sigma_i^* f(\mathbf{x}_i) \leq \hat{R}_m(\mathcal{F}_t) + 4\sqrt{\frac{\ln(1/\delta_t)}{2m}}$$

with probability $1 - \delta_t$

Bounding MKL

- Hence taking $\delta_t = \delta/2(N+1)$ for $t = 0, \ldots, N$

$$\begin{aligned}
&\hat{R}_m\left(\mathcal{F} = \bigcup_{t=1}^{N} \mathcal{F}_t\right) \\
&\quad \leq \frac{2}{m} \sup_{f \in \mathcal{F}} \sum_{i=1}^{m} \sigma_i^* f(\mathbf{x}_i) + 4\sqrt{\frac{\ln(2(N+1)/\delta)}{2m}} \\
&\quad \leq \frac{2}{m} \max_{1 \leq t \leq N} \sup_{f \in \mathcal{F}_t} \sum_{i=1}^{m} \sigma_i^* f(\mathbf{x}_i) + 4\sqrt{\frac{\ln(2(N+1)/\delta)}{2m}} \\
&\quad \leq \frac{2}{m} \max_{1 \leq t \leq N} \hat{R}_m(\mathcal{F}_t) + 8\sqrt{\frac{\ln(2(N+1)/\delta)}{2m}}
\end{aligned}$$

with probability $1 - \delta/2$.

Bounding MKL

- This gives an overall bound on the generalisation of MKL of

$$P(y \neq \mathrm{sgn}(g(\mathbf{x}))) \leq \frac{1}{m\gamma}\sum_{i=1}^{m}\xi_i + \frac{2}{\gamma m}\max_{1\leq t\leq N}\mathrm{tr}(\mathbf{K}_t) + 8\sqrt{\frac{\ln(2(N+1)/\delta)}{2m}} + 3\sqrt{\frac{\ln(4/\delta)}{2m}}$$

  where $\mathbf{K}_t$ is the t-th kernel matrix.

- Bound gives only a logarithmic (additive) dependence on the number of kernels.

  ⋆ Zakria Hussain and John Shawe-Taylor (2011) Improved Loss Bounds For Multiple Kernel Learning, Proceedings of AISTATS, 370-377.

Bounding MKL

- This gives an overall bound on the generalisation of MKL of

$$P(y \neq \mathrm{sgn}(g(\mathbf{x}))) \leq \frac{1}{m\gamma}\sum_{i=1}^{m}\xi_i + \frac{2}{\gamma m}\max_{1\leq t\leq N}\mathrm{tr}(\mathbf{K}_t) + 8\sqrt{\frac{\ln(2(N+1)/\delta)}{2m}} + 3\sqrt{\frac{\ln(4/\delta)}{2m}}$$

  where $\mathbf{K}_t$ is the t-th kernel matrix.
- Bound gives only a logarithmic (additive) dependence on the number of kernels.

  ⋆ Zakria Hussain and John Shawe-Taylor (2011) Improved Loss Bounds For Multiple Kernel Learning, Proceedings of AISTATS, 370-377.

Experimental results with large-scale MKL

- Vedaldi et al. have applied to the PASCAL Visual Objects Challenge (VOC 2007) data and
    - improvements over the winners of the challenge in 17 out of the 20 categories
    - in more than half of the categories the increase in average precision was over 25%
    - have also scaled effectively to millions of kernels

⋆ A. Vedaldi, V. Gulshan, M. Varma and A. Zisserman. Multiple kernels for object detection. In Proceedings CVPR, Kyoto, Japan, September 2009.

Linear programming boosting

Replacing the 2-norm regularisation of the SVM with a 1-norm gives a linear programme: can solve its dual using an iterative method:

```
1  initialise $u_i = 1/m, i = 1, \dots, m$, $\beta = \infty$, $J = \emptyset$
2  choose $j^\star$ that maximises $f(j) = \sum_{i=1}^m u_i y_i \mathbf{H}_{ij}$
3  if $f(j^\star) \leq \beta$ solve primal restricted to $J$ and exit
4  $J = J \cup \{j^\star\}$
5  Solve dual restricted to set $J$ to give $u_i, \beta$
6  Go to 2
```

- Note that u_i is a distribution on the examples
- Each j added acts like an additional weak learner
- $f(j)$ is simply the weighted classification accuracy
- Hence gives 'boosting' algorithm - with previous weights updated satisfying error bound
- Guaranteed convergence and soft stopping criteria

Linear Programming MKL

- Column generation gives efficient MKL if we can pick the best weak learner in each $\mathcal{F}_t$ efficiently:

$$\begin{aligned}
\sup_{f \in \mathcal{F}_t} \sum_{i=1}^{m} u_i y_i f(\mathbf{x}_i) &= \sup_{\mathbf{w}:\|\mathbf{w}\|\leq 1} \sum_{i=1}^{m} u_i y_i \left\langle \mathbf{w}, \phi_t(\mathbf{x}_i) \right\rangle \\
&= \sup_{\mathbf{w}:\|\mathbf{w}\|\leq 1} \left\langle \mathbf{w}, \sum_{i=1}^{m} u_i y_i \phi_t(\mathbf{x}_i) \right\rangle \\
&= \left\| \sum_{i=1}^{m} u_i y_i \phi_t(\mathbf{x}_i) \right\| \\
&= \sqrt{\mathbf{u}'\mathbf{Y}\mathbf{K}_t\mathbf{Y}\mathbf{u}} =: N_t
\end{aligned}$$

easily computable from the kernel matrices (note that $\mathbf{u}$ is sparse after first iteration and can also be chosen sparse at the start).

MKL Algorithmics

- The optimal weak learner from $\mathcal{F}_t$ is realised by the weight vector that achieves the supremum

$$\mathbf{w} = \frac{\sum_{i=1}^m u_i y_i \phi_t(\mathbf{x}_i)}{\|\sum_{i=1}^m u_i y_i \phi_t(\mathbf{x}_i)\|}$$

  which has dual representation:

$$\alpha_i = \frac{1}{N_t} u_i y_i$$

- Hence, can use the linear programming boosting approach to implement multiple kernel learning.
- More generally can view the $\mathbf{u}$ vector as a signal to refine other representations

MKL Algorithmics

- The optimal weak learner from $\mathcal{F}_t$ is realised by the weight vector that achieves the supremum

$$\mathbf{w} = \frac{\sum_{i=1}^m u_i y_i \phi_t(\mathbf{x}_i)}{\|\sum_{i=1}^m u_i y_i \phi_t(\mathbf{x}_i)\|}$$

which has dual representation:

$$\alpha_i = \frac{1}{N_t} u_i y_i$$

- Hence, can use the linear programming boosting approach to implement multiple kernel learning.
- More generally can view the $\mathbf{u}$ vector as a signal to refine other representations

MKL Algorithmics

- The optimal weak learner from $\mathcal{F}_t$ is realised by the weight vector that achieves the supremum

$$\mathbf{w} = \frac{\sum_{i=1}^m u_i y_i \phi_t(\mathbf{x}_i)}{\|\sum_{i=1}^m u_i y_i \phi_t(\mathbf{x}_i)\|}$$

  which has dual representation:

$$\alpha_i = \frac{1}{N_t} u_i y_i$$

- Hence, can use the linear programming boosting approach to implement multiple kernel learning.
- More generally can view the $\mathbf{u}$ vector as a signal to refine other representations

Learning Fisher kernels

- As an example consider Fisher kernels over a parametrised probabilistic model
- Signal $\mathbf{u}$ can be used to optimise the kernel by adjusting the parameters of the model
- Using HMMs for modelling time series data this approach was applied to forecasting foreign exchange rates.
- Some encouraging results

  ⋆ Sewell, M., Shawe-Taylor, J. (2012). Forecasting foreign exchange rates using kernel methods. Expert Systems with Applications 39(9), 7652-7662

  ⋆ Fletcher, T. and Shawe-Taylor, J. (2012) MKL with Fisher Kernels for High Frequency Currency Prediction, Computational Economics

Learning Fisher kernels

- As an example consider Fisher kernels over a parametrised probabilistic model
- Signal **u** can be used to optimise the kernel by adjusting the parameters of the model
- Using HMMs for modelling time series data this approach was applied to forecasting foreign exchange rates.
- Some encouraging results

  ⋆ Sewell, M., Shawe-Taylor, J. (2012). Forecasting foreign exchange rates using kernel methods. Expert Systems with Applications 39(9), 7652-7662

  ⋆ Fletcher, T. and Shawe-Taylor, J. (2012) MKL with Fisher Kernels for High Frequency Currency Prediction, Computational Economics

Learning Fisher kernels

- As an example consider Fisher kernels over a parametrised probabilistic model
- Signal **u** can be used to optimise the kernel by adjusting the parameters of the model
- Using HMMs for modelling time series data this approach was applied to forecasting foreign exchange rates.
- Some encouraging results

  ⋆ Sewell, M., Shawe-Taylor, J. (2012). Forecasting foreign exchange rates using kernel methods. Expert Systems with Applications 39(9), 7652-7662

  ⋆ Fletcher, T. and Shawe-Taylor, J. (2012) MKL with Fisher Kernels for High Frequency Currency Prediction, Computational Economics

Learning Fisher kernels

- As an example consider Fisher kernels over a parametrised probabilistic model
- Signal **u** can be used to optimise the kernel by adjusting the parameters of the model
- Using HMMs for modelling time series data this approach was applied to forecasting foreign exchange rates.
- Some encouraging results

  ⋆ Sewell, M., Shawe-Taylor, J. (2012). Forecasting foreign exchange rates using kernel methods. Expert Systems with Applications 39(9), 7652-7662

  ⋆ Fletcher, T. and Shawe-Taylor, J. (2012) MKL with Fisher Kernels for High Frequency Currency Prediction, Computational Economics

Non-linear Feature Selection

- There is an interesting result that relates kernel target alignment to maximal covariance with the output

$$\sqrt{\mathrm{E}_{(\mathbf{x},y)\sim P,(\mathbf{x}',y')\sim P}[yy'\kappa(\mathbf{x},\mathbf{x}')]} = \\ = \sup_{\mathbf{w}:\|\mathbf{w}\|\leq 1} \mathrm{E}_{(\mathbf{x},y)\sim P}[y\langle \mathbf{w}, \phi(\mathbf{x})\rangle]$$

- Suggests defining the contribution of a feature as

$$\begin{aligned} c_i &= \mathrm{E}_{S\sim\mathcal{S}_i}\left[\mathrm{E}_{(\mathbf{x},y)\sim P,(\mathbf{x}',y')\sim P}[yy'\kappa_S(\mathbf{x},\mathbf{x}')]\right] - \\ & \quad \mathrm{E}_{S'\sim\mathcal{S}_{\setminus i}}\left[\mathrm{E}_{(\mathbf{x},y)\sim P,(\mathbf{x}',y')\sim P}[yy'\kappa_{S'}(\mathbf{x},\mathbf{x}')]\right], \end{aligned}$$

where $\mathcal{S}_i$ and $\mathcal{S}_{\setminus i}$ are distributions over fixed size sets of features.

Non-linear Feature Selection

- There is an interesting result that relates kernel target alignment to maximal covariance with the output

$$\sqrt{\mathrm{E}_{(\mathbf{x},y)\sim P,(\mathbf{x}',y')\sim P}[yy'\kappa(\mathbf{x},\mathbf{x}')]} = \\ = \sup_{\mathbf{w}:\|\mathbf{w}\|\leq 1} \mathrm{E}_{(\mathbf{x},y)\sim P}[y\langle \mathbf{w}, \phi(\mathbf{x})\rangle]$$

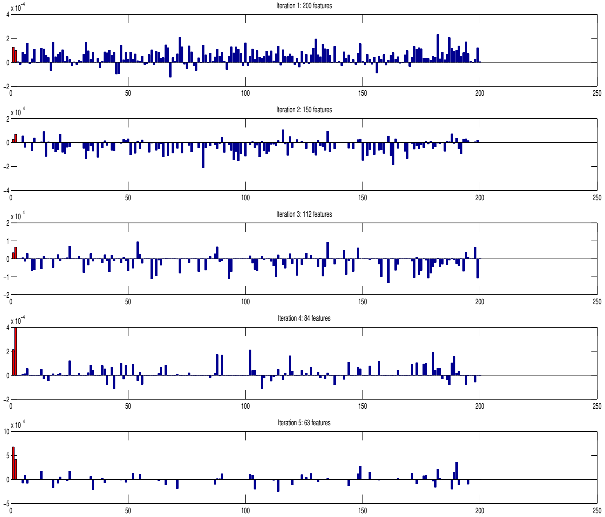
- Suggests defining the contribution of a feature as

$$\begin{aligned} c_i &= \mathrm{E}_{S\sim\mathcal{S}_i}\left[\mathrm{E}_{(\mathbf{x},y)\sim P,(\mathbf{x}',y')\sim P}[yy'\kappa_S(\mathbf{x},\mathbf{x}')]\right] - \\ & \quad \mathrm{E}_{S'\sim\mathcal{S}_{\setminus i}}\left[\mathrm{E}_{(\mathbf{x},y)\sim P,(\mathbf{x}',y')\sim P}[yy'\kappa_{S'}(\mathbf{x},\mathbf{x}')]\right], \end{aligned}$$

where $\mathcal{S}_i$ and $\mathcal{S}_{\setminus i}$ are distributions over fixed size sets of features.

Example

Consider 200-dimensional function that is XOR of the first two features. Take Gaussian kernel - gives results after successive cullings:



Analysis

Some properties:

- Irrelevant features make negative contribution
- Chances of relevant feature being in bottom quarter of the ranked contributions on a sufficiently large random sample is arbitrarily small
- Hence, can cull 25% of bottom ranked features without risking losing good features
- possibility of locking in features that appear in top 25% consisitently

Analysis

Some properties:

- Irrelevant features make negative contribution
- Chances of relevant feature being in bottom quarter of the ranked contributions on a sufficiently large random sample is arbitrarily small
- Hence, can cull 25% of bottom ranked features without risking losing good features
- possibility of locking in features that appear in top 25% consisitently

Analysis

Some properties:

- Irrelevant features make negative contribution
- Chances of relevant feature being in bottom quarter of the ranked contributions on a sufficiently large random sample is arbitrarily small
- Hence, can cull 25% of bottom ranked features without risking losing good features
- possibility of locking in features that appear in top 25% consisitently

Analysis

Some properties:

- Irrelevant features make negative contribution
- Chances of relevant feature being in bottom quarter of the ranked contributions on a sufficiently large random sample is arbitrarily small
- Hence, can cull 25% of bottom ranked features without risking losing good features
- possibility of locking in features that appear in top 25% consisitently

Analysis

Some properties:

- Irrelevant features make negative contribution
- Chances of relevant feature being in bottom quarter of the ranked contributions on a sufficiently large random sample is arbitrarily small
- Hence, can cull 25% of bottom ranked features without risking losing good features
- possibility of locking in features that appear in top 25% consisitently

Results

On artificial data

Dataset	Algorithm	Accuracy	Features	Precision	Recall
Linear Weston	randSel	97.7 ± 2.0	3.0 ± 0.0	91.8 ± 23.1	72.0 ± 16.6
	BaHsic	97.3 ± 3.1	5.0 ± 0.0	91.5 ± 19.4	70.7 ± 14.9
	FoHsic	97.1 ± 3.1	6.0 ± 0.0	95.9 ± 12.0	74.7 ± 17.7
	Corr. Coeff.	92.4 ± 7.8	4.0 ± 0.0	96.1 ± 15.1	76.0 ± 15.5
	Stab. Sel.	97.3 ± 3.1	2.0 ± 0.0	100.0 ± 0.0	40.0 ± 0.0
	RFE	95.3 ± 3.9	5.0 ± 0.0	66.9 ± 33.7	56.0 ± 13.5
Non-Linear Weston	randSel	99.0 ± 1.4	5.0 ± 0.0	100.0 ± 0.0	89.3 ± 12.8
	BaHsic	99.8 ± 0.9	4.0 ± 0.0	100.0 ± 0.0	80.0 ± 7.6
	FoHsic	99.8 ± 0.9	4.0 ± 0.0	100.0 ± 0.0	82.7 ± 7.0
	Corr. Coeff.	56.2 ± 6.8	21.0 ± 0.0	1.7 ± 2.5	18.7 ± 31.6
	Stab. Sel.	50.0 ± 7.1	2.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	RFE	98.9 ± 2.7	5.0 ± 0.0	97.8 ± 5.9	100.0 ± 0.0
XOR	randSel	95.7 ± 3.3	2.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	BaHsic	95.7 ± 3.3	2.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	FoHsic	52.0 ± 6.5	53.0 ± 0.0	9.4 ± 25.3	36.7 ± 44.2
	Corr. Coeff.	58.1 ± 14.9	8.0 ± 0.0	10.4 ± 10.3	50.0 ± 42.3
	Stab. Sel.	49.3 ± 11.1	2.0 ± 0.0	13.3 ± 22.9	13.3 ± 22.9
	RFE	91.8 ± 12.1	2.0 ± 0.0	96.7 ± 12.9	96.7 ± 12.9

Results

On real world omic and microarray data

Dataset	Algorithm	Accuracy	Features	Dataset	Algorithm	Accuracy	Features
TB	randSel	82.9 ± 8.4	64.6 ± 70.3	TB	randSel	82.0 ± 8.6	42.0 ± 47.7
Task 1	BaHsic	81.7 ± 9.0	74.7 ± 101.3	Task 2	BaHsic	81.1 ± 8.9	33.1 ± 40.6
	FoHsic	81.3 ± 9.4	68.0 ± 66.5		FoHsic	80.6 ± 10.8	31.1 ± 35.3
	Corr. Coeff.	82.4 ± 8.8	123.6 ± 85.8		Corr. Coeff.	82.7 ± 9.4	73.4 ± 55.5
	Stab. Sel.	82.9 ± 7.3	121.7 ± 56.4		Stab. Sel.	80.7 ± 8.4	137.3 ± 154.7
	RFE	81.9 ± 8.0	236.2 ± 160.2		RFE	80.2 ± 9.1	82.4 ± 139.9
TB	randSel	86.0 ± 8.1	45.3 ± 33.6	TB	randSel	87.6 ± 4.9	58.5 ± 93.8
Task 3	BaHsic	85.6 ± 9.5	53.3 ± 39.5	Micro	BaHsic	86.1 ± 6.4	61.2 ± 94.7
	FoHsic	85.6 ± 8.8	53.6 ± 44.7	Array	FoHsic	85.2 ± 7.9	52.5 ± 92.9
	Corr. Coeff.	85.4 ± 8.8	132.9 ± 89.7		Corr. Coeff.	84.1 ± 6.6	143.5 ± 114.2
	Stab. Sel.	84.1 ± 9.6	60.0 ± 47.9		Stab. Sel.	87.1 ± 5.9	161.8 ± 136.0
	RFE	83.9 ± 9.2	43.5 ± 71.6		RFE	85.7 ± 6.8	158.0 ± 137.6

Results

Have applied to Deep learning challenge (Black Box Learning Challenge 2013)

- Initial sparse filtering step (Jiquan et al., 2011) – just one preprocessing layer
- performed the culling steps described above
- used the LPBoost MKL method to combine the corresponding kernels created
- Method was third in the final ranking (scored 0.685 vs winning score of 0.702)

Results

Have applied to Deep learning challenge (Black Box Learning Challenge 2013)

- Initial sparse filtering step (Jiquan et al., 2011) – just one preprocessing layer
- performed the culling steps described above
- used the LPBoost MKL method to combine the corresponding kernels created
- Method was third in the final ranking (scored 0.685 vs winning score of 0.702)

Results

Have applied to Deep learning challenge (Black Box Learning Challenge 2013)

- Initial sparse filtering step (Jiquan et al., 2011) – just one preprocessing layer
- performed the culling steps described above
- used the LPBoost MKL method to combine the corresponding kernels created
- Method was third in the final ranking (scored 0.685 vs winning score of 0.702)

Results

Have applied to Deep learning challenge (Black Box Learning Challenge 2013)

- Initial sparse filtering step (Jiquan et al., 2011) – just one preprocessing layer
- performed the culling steps described above
- used the LPBoost MKL method to combine the corresponding kernels created
- Method was third in the final ranking (scored 0.685 vs winning score of 0.702)

Results

Have applied to Deep learning challenge (Black Box Learning Challenge 2013)

- Initial sparse filtering step (Jiquan et al., 2011) – just one preprocessing layer
- performed the culling steps described above
- used the LPBoost MKL method to combine the corresponding kernels created
- Method was third in the final ranking (scored 0.685 vs winning score of 0.702)

Summary and Conclusions

- Learning deep representations is important for analysis of real data
- Many kernel practitioners are using deep learning but typically in a relatively ad-hoc manner
- Attempts to use more principled methods have been rewarded with considerable success
- There is already a range of theoretical results relating to deep-er learning kernel methods that place the approaches on a firm-er footing

Summary and Conclusions

- Learning deep representations is important for analysis of real data
- Many kernel practitioners are using deep learning but typically in a relatively ad-hoc manner
- Attempts to use more principled methods have been rewarded with considerable success
- There is already a range of theoretical results relating to deep-er learning kernel methods that place the approaches on a firm-er footing

Summary and Conclusions

- Learning deep representations is important for analysis of real data
- Many kernel practitioners are using deep learning but typically in a relatively ad-hoc manner
- Attempts to use more principled methods have been rewarded with considerable success
- There is already a range of theoretical results relating to deep-er learning kernel methods that place the approaches on a firm-er footing

Summary and Conclusions

- Learning deep representations is important for analysis of real data
- Many kernel practitioners are using deep learning but typically in a relatively ad-hoc manner
- Attempts to use more principled methods have been rewarded with considerable success
- There is already a range of theoretical results relating to deep-er learning kernel methods that place the approaches on a firm-er footing