Statistics and Clustering with Kernels

Christoph Lampert & Matthew Blaschko

Max-Planck-Institute for Biological Cybernetics
Department Schölkopf: Empirical Inference
Tübingen, Germany

Visual Geometry Group
University of Oxford
United Kingdom

June 20, 2009

Overview

- Kernel Ridge Regression
- Kernel PCA
- Spectral Clustering
- Kernel Covariance and Canonical Correlation Analysis
- Kernel Measures of Independence

Kernel Ridge Regression

- Regularized least squares regression:

$$\min_{w} \sum_{i=1}^{n} \left(y_i - \langle w, x_i \rangle\right)^2 + \lambda \|w\|^2$$

Kernel Ridge Regression

- Regularized least squares regression:

$$\min_{w} \sum_{i=1}^{n} \left(y_i - \langle w, x_i \rangle\right)^2 + \lambda \|w\|^2$$

- Replace w with $\sum_{i=1}^{n} \alpha_i x_i$

$$\min_{\alpha} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{n} \langle x_i, x_j \rangle \right)^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \langle x_i, x_j \rangle$$

- α^* can be solved in closed form solution

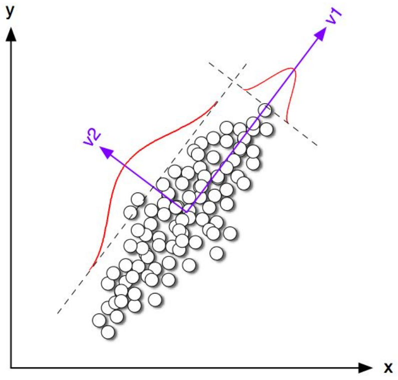
$$\alpha^* = \left(K + \lambda I\right)^{-1} y$$

PCA

Equivalent formulations:

- Minimize squared error between original data and a projection of our data into a lower dimensional subspace
- Maximize variance of projected data

Solutions: Eigenvectors of the empirical covariance matrix



(fig: Tristan Jehan)

PCA continued

- Empirical covariance matrix (biased):

$$\hat{C} = \frac{1}{n} \sum_i (x_i - \mu)(x_i - \mu)^T$$

  where μ is the sample mean.

- $\hat{C}$ is positive (semi-)definite symmetric

- PCA:

$$\max_w \frac{w^T \hat{C} w}{\|w\|^2}$$

Data Centering

- We use the notation X to denote the design matrix where every column of X is a data sample
- We can define a centering matrix

$$H = I - \frac{1}{n} e e^T$$

where e is a vector of all ones

Data Centering

- We use the notation X to denote the design matrix where every column of X is a data sample
- We can define a centering matrix

$$H = I - \frac{1}{n} e e^T$$

  where e is a vector of all ones
- H is idempotent, symmetric, and positive semi-definite (rank $n-1$)

Data Centering

- We use the notation X to denote the design matrix where every column of X is a data sample
- We can define a centering matrix

  $$H = I - \frac{1}{n}ee^T$$

  where e is a vector of all ones
- H is idempotent, symmetric, and positive semi-definite (rank $n-1$)
- The design matrix of *centered* data can be written compactly in matrix form as XH
  - The ith column of XH is equal to $x_i - \mu$, where $\mu = \frac{1}{n}\sum_j x_j$ is the sample mean

Kernel PCA

- PCA:

$$\max_w \frac{w^T \hat{C} w}{\|w\|^2}$$

- Kernel PCA:
  - Replace w by $\sum_i \alpha_i (x_i - \mu)$ - this can be represented compactly in matrix form by $w = XH\alpha$ where X is the design matrix, H is the centering matrix, and α is the coefficient vector.

Kernel PCA

- PCA:

$$\max_w \frac{w^T \hat{C} w}{\|w\|^2}$$

- Kernel PCA:
  - Replace w by $\sum_i \alpha_i (x_i - \mu)$ - this can be represented compactly in matrix form by $w = XH\alpha$ where X is the design matrix, H is the centering matrix, and α is the coefficient vector.
  - Compute $\hat{C}$ in matrix form as $\hat{C} = \frac{1}{n} XHX^T$

Kernel PCA

- PCA:

$$\max_w \frac{w^T \hat{C} w}{\|w\|^2}$$

- Kernel PCA:
  - Replace w by $\sum_i \alpha_i (x_i - \mu)$ - this can be represented compactly in matrix form by $w = XH\alpha$ where X is the design matrix, H is the centering matrix, and α is the coefficient vector.
  - Compute $\hat{C}$ in matrix form as $\hat{C} = \frac{1}{n} XHX^T$
  - Denote the matrix of pairwise inner products $K = X^T X$, i.e. $K_{ij} = \langle x_i, x_j \rangle$

Kernel PCA

- PCA:

$$\max_{w} \frac{w^T \hat{C} w}{\|w\|^2}$$

- Kernel PCA:
  - Replace w by $\sum_i \alpha_i (x_i - \mu)$ - this can be represented compactly in matrix form by $w = XH\alpha$ where X is the design matrix, H is the centering matrix, and α is the coefficient vector.
  - Compute $\hat{C}$ in matrix form as $\hat{C} = \frac{1}{n} XHX^T$
  - Denote the matrix of pairwise inner products $K = X^T X$, i.e. $K_{ij} = \langle x_i, x_j \rangle$

$$\max_{w} \frac{w^T \hat{C} w}{\|w\|^2} = \max_{\alpha} \frac{1}{n} \frac{\alpha^T HKHKH\alpha}{\alpha^T HKH\alpha}$$

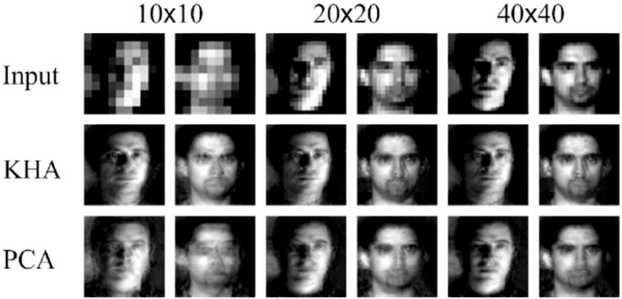
This is a Rayleigh quotient with known solution

$$HKH\beta_i = \lambda_i \beta_i$$

Kernel PCA

- Set β to be the eigenvectors of HKH, and λ the corresponding eigenvalues
- Set $\alpha = \beta\lambda^{-\frac{1}{2}}$

Example, image super-resolution:



(fig: Kim et al., PAMI 2005.)

Overview

- Kernel Ridge Regression
- Kernel PCA
- **Spectral Clustering**
- Kernel Covariance and Canonical Correlation Analysis
- Kernel Measures of Independence

Spectral Clustering

- Represent similarity of images by weights on a graph
- Normalized cuts optimizes the ratio of the cost of a cut and the volume of each cluster

$$Ncut(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{vol(A_i)}$$

Spectral Clustering

- Represent similarity of images by weights on a graph
- Normalized cuts optimizes the ratio of the cost of a cut and the volume of each cluster

$$Ncut(A_1, \dots, A_k) = \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{vol(A_i)}$$

- Exact optimization is NP-hard, but relaxed version can be solved by finding the eigenvalues of the *graph Laplacian*

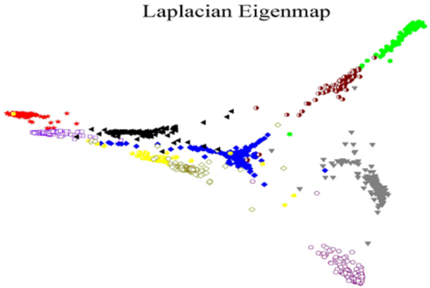
$$\mathcal{L} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

where D is the diagonal matrix with entries equal to the row sums of similarity matrix, A.

Spectral Clustering (continued)

- Compute $\mathcal{L} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$.
- Map data points based on the eigenvalues of $\mathcal{L}$

  Example, handwritten digits (0-9):



(fig: Xiaofei He)

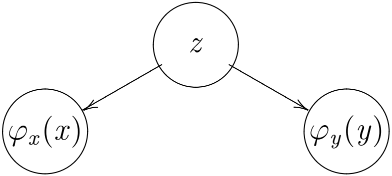
- Cluster in mapped space using k-means

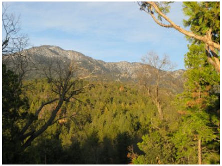
Overview

- Kernel Ridge Regression
- Kernel PCA
- Spectral Clustering
- **Kernel Covariance and Canonical Correlation Analysis**
- Kernel Measures of Independence

Multimodal Data

- A latent aspect relates data that are present in multiple modalities
- e.g. images and text

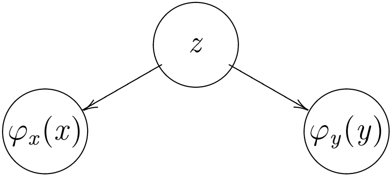


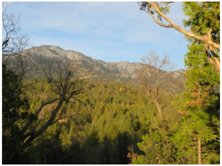
x: 

y: "A view from Idyllwild, California, with pine trees and snow capped Marion Mountain under a blue sky."

Multimodal Data

- A latent aspect relates data that are present in multiple modalities
- e.g. images and text



x: 

y: “A view from Idyllwild, California, with pine trees and snow capped Marion Mountain under a blue sky.”

- Learn kernelized projections that relate both spaces

Kernel Covariance

- KPCA is maximization of auto-covariance
- Instead maximize *cross*-covariance

$$\max_{w_x, w_y} \frac{w_x C_{xy} w_y}{||w_x|| \, ||w_y||}$$

Kernel Covariance

- KPCA is maximization of auto-covariance
- Instead maximize *cross*-covariance

$$\max_{w_x, w_y} \frac{w_x C_{xy} w_y}{||w_x|| \, ||w_y||}$$

- Can also be kernelized (replace w_x by $\sum_i \alpha_i (x_i - \mu_x)$, etc.)

$$max_{\alpha,\beta} \frac{\alpha^T H K_x H K_y H \beta}{\sqrt{\alpha^T H K_x H \alpha \beta^T H K_y H \beta}}$$

Kernel Covariance

- KPCA is maximization of auto-covariance
- Instead maximize *cross*-covariance

$$\max_{w_x, w_y} \frac{w_x C_{xy} w_y}{||w_x|| ||w_y||}$$

- Can also be kernelized (replace w_x by $\sum_i \alpha_i (x_i - \mu_x)$, etc.)

$$max_{\alpha,\beta} \frac{\alpha^T H K_x H K_y H \beta}{\sqrt{\alpha^T H K_x H \alpha \beta^T H K_y H \beta}}$$

- Solution is given by (generalized) eigenproblem

$$\begin{pmatrix} \mathbf{0} & HK_xHK_yH \\ HK_yHK_xH & \mathbf{0} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \lambda \begin{pmatrix} HK_xH & \mathbf{0} \\ \mathbf{0} & HK_yH \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

Kernel Canonical Correlation Analysis (KCCA)

- Alternately, maximize *correlation* instead of covariance

$$\max_{w_x, w_y} \frac{w_x^T C_{xy} w_y}{\sqrt{w_x^T C_{xx} w_x w_y^T C_{yy} w_y}}$$

Kernel Canonical Correlation Analysis (KCCA)

- Alternately, maximize *correlation* instead of covariance

$$\max_{w_x, w_y} \frac{w_x^T C_{xy} w_y}{\sqrt{w_x^T C_{xx} w_x w_y^T C_{yy} w_y}}$$

- Kernelization is straightforward as before

$$\max_{\alpha, \beta} \frac{\alpha^T H K_x H K_y H \beta}{\sqrt{\alpha^T \left(H K_x H\right)^2 \alpha \beta^T \left(H K_y H\right)^2 \beta}}$$

KCCA (continued)

- Problem:
- If the data in either modality are linearly independent (as many dimensions as data points), there exists a projection of the data that respects any arbitrary ordering
- Perfect correlation can always be achieved

KCCA (continued)

- Problem:
- If the data in either modality are linearly independent (as many dimensions as data points), there exists a projection of the data that respects any arbitrary ordering
- Perfect correlation can always be achieved
- This is even more likely when a kernel is used (e.g. Gaussian)

KCCA (continued)

- Problem:
- If the data in either modality are linearly independent (as many dimensions as data points), there exists a projection of the data that respects any arbitrary ordering
- Perfect correlation can always be achieved
- This is even more likely when a kernel is used (e.g. Gaussian)

- Solution: Regularize

$$\max_{w_x, w_y} \frac{w_x^T C_{xy} w_y}{\sqrt{(w_x^T C_{xx} w_x + \varepsilon_x \|w_x\|^2) \left(w_y^T C_{yy} w_y + \varepsilon_y \|w_y\|^2\right)}}$$

- As $\varepsilon_x \to \infty$, $\varepsilon_x \to \infty$, solution approaches maximum covariance

KCCA Algorithm

- Compute K_x, K_y
- Solve for α and β as the eigenvectors of

$$\begin{pmatrix} \mathbf{0} & HK_xHK_yH \\ HK_yHK_xH & \mathbf{0} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \lambda \begin{pmatrix} (HK_xH)^2 + \varepsilon_x HK_xH & \mathbf{0} \\ \mathbf{0} & (HK_yH)^2 + \varepsilon_y HK_yH \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

Content Based Image Retrieval with KCCA

- Hardoon et al., 2004
- Training data consists of images with text captions
- Learn embeddings of both spaces using KCCA and appropriately chosen image and text kernels
- Retrieval consists of finding images whose embeddings are related to the embedding of the text query

Content Based Image Retrieval with KCCA

- Hardoon et al., 2004
- Training data consists of images with text captions
- Learn embeddings of both spaces using KCCA and appropriately chosen image and text kernels
- Retrieval consists of finding images whose embeddings are related to the embedding of the text query

- A kind of multi-variate regression

Overview

- Kernel Ridge Regression
- Kernel PCA
- Spectral Clustering
- Kernel Covariance and Canonical Correlation Analysis
- **Kernel Measures of Independence**

Kernel Measures of Independence

- We know how to measure correlation in the kernelized space

Kernel Measures of Independence

- We know how to measure correlation in the kernelized space
- Independence implies zero correlation

Kernel Measures of Independence

- We know how to measure correlation in the kernelized space
- Independence implies zero correlation
- Different kernels encode different statistical properties of the data

Kernel Measures of Independence

- We know how to measure correlation in the kernelized space
- Independence implies zero correlation
- Different kernels encode different statistical properties of the data
- Use an appropriate kernel such that zero correlation in the Hilbert space implies independence

Example: Polynomial Kernel

- First degree polynomial kernel (i.e. linear) captures correlation only
- Second degree polynomial kernel captures all second order statistics
- ...

Example: Polynomial Kernel

- First degree polynomial kernel (i.e. linear) captures correlation only
- Second degree polynomial kernel captures all second order statistics
- ...
- A Gaussian kernel can be written

$$k(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2} = e^{-\gamma\langle x_i, x_i\rangle} e^{2\gamma\langle x_i, x_j\rangle} e^{-\gamma\langle x_j, x_j\rangle}$$

  and we can use the identity

$$e^z = \sum_{i=1}^{\infty} \frac{1}{i!} z^i$$

Example: Polynomial Kernel

- First degree polynomial kernel (i.e. linear) captures correlation only
- Second degree polynomial kernel captures all second order statistics
- ...
- A Gaussian kernel can be written

$$k(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2} = e^{-\gamma\langle x_i, x_i\rangle} e^{2\gamma\langle x_i, x_j\rangle} e^{-\gamma\langle x_j, x_j\rangle}$$

  and we can use the identity

$$e^z = \sum_{i=1}^{\infty} \frac{1}{i!} z^i$$

- We can view the Gaussian kernel as being related to an appropriately scaled infinite dimensional polynomial kernel

Example: Polynomial Kernel

- First degree polynomial kernel (i.e. linear) captures correlation only
- Second degree polynomial kernel captures all second order statistics
- ...
- A Gaussian kernel can be written

$$k(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2} = e^{-\gamma\langle x_i, x_i\rangle} e^{2\gamma\langle x_i, x_j\rangle} e^{-\gamma\langle x_j, x_j\rangle}$$

  and we can use the identity

$$e^z = \sum_{i=1}^{\infty} \frac{1}{i!} z^i$$

- We can view the Gaussian kernel as being related to an appropriately scaled infinite dimensional polynomial kernel
  - captures all order statistics

Hilbert-Schmidt Independence Criterion

- $\mathcal{F}$ RKHS on $\mathcal{X}$ with kernel $k_x(x, x')$, $\mathcal{G}$ RKHS on $\mathcal{Y}$ with kernel $k_y(y, y')$

Hilbert-Schmidt Independence Criterion

- $\mathcal{F}$ RKHS on $\mathcal{X}$ with kernel $k_x(x, x')$, $\mathcal{G}$ RKHS on $\mathcal{Y}$ with kernel $k_y(y, y')$
- Covariance operator: $C_{xy} : \mathcal{G} \to \mathcal{F}$ such that

$$\langle f, C_{xy} g \rangle_{\mathcal{F}} = E_{x,y}[f(x)g(y)] - E_x[f(x)]E_y[g(y)]$$

Hilbert-Schmidt Independence Criterion

- $\mathcal{F}$ RKHS on $\mathcal{X}$ with kernel $k_x(x, x')$, $\mathcal{G}$ RKHS on $\mathcal{Y}$ with kernel $k_y(y, y')$
- Covariance operator: $C_{xy} : \mathcal{G} \to \mathcal{F}$ such that

$$\langle f, C_{xy} g \rangle_{\mathcal{F}} = E_{x,y}[f(x)g(y)] - E_x[f(x)]E_y[g(y)]$$

- HSIC is the Hilbert-Schmidt norm of C_{xy} (Fukumizu et al. 2008):

$$\mathrm{HSIC} := \|C_{xy}\|_{HS}^2$$

Hilbert-Schmidt Independence Criterion

- $\mathcal{F}$ RKHS on $\mathcal{X}$ with kernel $k_x(x, x')$, $\mathcal{G}$ RKHS on $\mathcal{Y}$ with kernel $k_y(y, y')$
- Covariance operator: $C_{xy} : \mathcal{G} \to \mathcal{F}$ such that

$$\langle f, C_{xy} g \rangle_{\mathcal{F}} = E_{x,y}[f(x)g(y)] - E_x[f(x)]E_y[g(y)]$$

- HSIC is the Hilbert-Schmidt norm of C_{xy} (Fukumizu et al. 2008):

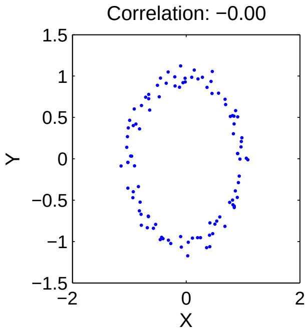
$$\mathrm{HSIC} := \| C_{xy} \|_{HS}^2$$

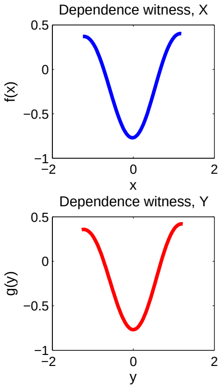
- (Biased) empirical HSIC:

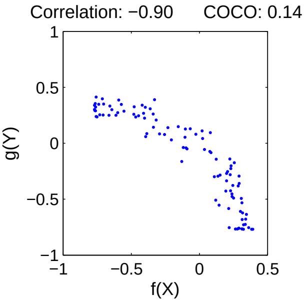
$$\widehat{HSIC} := \frac{1}{n^2} \mathrm{Tr}(K_x H K_y H)$$

Hilbert-Schmidt Independence Criterion (continued)

- Ring-shaped density, correlation approx. zero
- Maximum singular vectors (functions) of C_{xy}







Hilbert-Schmidt Normalized Independence Criterion

- Hilbert-Schmidt Independence Criterion analogous to cross-covariance
- Can we construct a version analogous to *correlation*?

Hilbert-Schmidt Normalized Independence Criterion

- Hilbert-Schmidt Independence Criterion analogous to cross-covariance
- Can we construct a version analogous to *correlation*?
- Simple modification: decompose Covariance operator (Baker 1973)

$$C_{xy} = C_{xx}^{\frac{1}{2}} V_{xy} C_{yy}^{\frac{1}{2}}$$

  where V_{xy} is the normalized cross-covariance operator (maximum singular value is bounded by 1)

Hilbert-Schmidt Normalized Independence Criterion

- Hilbert-Schmidt Independence Criterion analogous to cross-covariance
- Can we construct a version analogous to *correlation*?
- Simple modification: decompose Covariance operator (Baker 1973)

  $$C_{xy} = C_{xx}^{\frac{1}{2}} V_{xy} C_{yy}^{\frac{1}{2}}$$

  where V_{xy} is the normalized cross-covariance operator (maximum singular value is bounded by 1)
- Use norm of V_{xy} instead of the norm of C_{xy}

Hilbert-Schmidt Normalized Independence Criterion (continued)

- Define the normalized independence criterion to be the Hilbert-Schmidt norm of V_{xy}

$$\widehat{HSNIC} := \frac{1}{n^2} \operatorname{Tr} \Big[HK_xH \left(HK_xH + \varepsilon_x I\right)^{-1} \\ HK_yH \left(HK_yH + \varepsilon_y I\right)^{-1} \Big]$$

where ε_x and ε_y are regularization parameters as in KCCA

Hilbert-Schmidt Normalized Independence Criterion (continued)

- Define the normalized independence criterion to be the Hilbert-Schmidt norm of V_{xy}

$$\widehat{HSNIC} := \frac{1}{n^2} \operatorname{Tr} \Big[HK_x H \left(HK_x H + \varepsilon_x I \right)^{-1} \\ HK_y H \left(HK_y H + \varepsilon_y I \right)^{-1} \Big]$$

where ε_x and ε_y are regularization parameters as in KCCA

- If the kernels on x and y are characteristic (e.g. Gaussian kernels, see Fukumizu et al., 2008) $\| C_{xy} \|^2_{HS} = \| V_{xy} \|^2_{HS} = 0$ iff x and y are independent!

Applications of HS(N)IC

- Independence tests - is there anything to gain from the use of multi-modal data?

Applications of HS(N)IC

- Independence tests - is there anything to gain from the use of multi-modal data?
- Kernel ICA

Applications of HS(N)IC

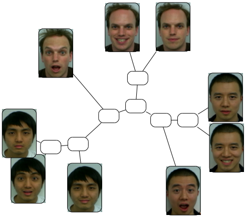
- Independence tests - is there anything to gain from the use of multi-modal data?
- Kernel ICA
- Maximize dependence with respect to some model parameters
    - Kernel target alignment (Cristianini et al., 2001)

Applications of HS(N)IC

- Independence tests - is there anything to gain from the use of multi-modal data?
- Kernel ICA
- Maximize dependence with respect to some model parameters
  - Kernel target alignment (Cristianini et al., 2001)
  - Learning spectral clustering (Bach & Jordan, 2003) - relates kernel learning and clustering

Applications of HS(N)IC

- Independence tests - is there anything to gain from the use of multi-modal data?
- Kernel ICA
- Maximize dependence with respect to some model parameters
  - Kernel target alignment (Cristianini et al., 2001)
  - Learning spectral clustering (Bach & Jordan, 2003) - relates kernel learning and clustering
  - Taxonomy discovery (Blaschko & Gretton, 2008)



Summary

In this section we learned how to

- Do basic operations in kernel space like:
  - Regularized least squares regression
  - Data centering
  - PCA

Summary

In this section we learned how to

- Do basic operations in kernel space like:
  - Regularized least squares regression
  - Data centering
  - PCA
- Learn with multi-modal data
  - Kernel Covariance
  - KCCA

Summary

In this section we learned how to

- Do basic operations in kernel space like:
  - Regularized least squares regression
  - Data centering
  - PCA
- Learn with multi-modal data
  - Kernel Covariance
  - KCCA
- Use kernels to construct statistical independence tests
  - Use appropriate kernels to capture relevant statistics
  - Measure dependence by norm of (normalized) covariance operator
  - Closed form solutions requiring only kernel matrices for each modality

Summary

In this section we learned how to

- Do basic operations in kernel space like:
  - Regularized least squares regression
  - Data centering
  - PCA
- Learn with multi-modal data
  - Kernel Covariance
  - KCCA
- Use kernels to construct statistical independence tests
  - Use appropriate kernels to capture relevant statistics
  - Measure dependence by norm of (normalized) covariance operator
  - Closed form solutions requiring only kernel matrices for each modality

- Questions?

Structured Output Learning

Christoph Lampert & Matthew Blaschko

Max-Planck-Institute for Biological Cybernetics
Department Schölkopf: Empirical Inference
Tübingen, Germany

Visual Geometry Group
University of Oxford
United Kingdom

June 20, 2009

What is Structured Output Learning?

- Regression maps from an input space to an output space

$$g : \mathcal{X} \to \mathcal{Y}$$

What is Structured Output Learning?

- Regression maps from an input space to an output space

$$g : \mathcal{X} \to \mathcal{Y}$$

- In typical scenarios, $\mathcal{Y} \equiv \mathbb{R}$ (regression) or $\mathcal{Y} \equiv \{-1, 1\}$ (classification)

What is Structured Output Learning?

- Regression maps from an input space to an output space

$$g : \mathcal{X} \rightarrow \mathcal{Y}$$

- In typical scenarios, $\mathcal{Y} \equiv \mathbb{R}$ (regression) or $\mathcal{Y} \equiv \{-1, 1\}$ (classification)
- Structured output learning extends this concept to more complex and interdependent output spaces

Examples of Structured Output Problems in Computer Vision

- Multi-class classification (Crammer & Singer, 2001)
- Hierarchical classification (Cai & Hofmann, 2004)
- Segmentation of 3d scan data (Anguelov et al., 2005)
- Learning a CRF model for stereo vision (Li & Huttenlocher, 2008)
- Object localization (Blaschko & Lampert, 2008)
- Segmentation with a learned CRF model (Szummer et al., 2008)
- ...
- More examples at CVPR 2009

Generalization of Regression

- Direct discriminative learning of $g : \mathcal{X} \rightarrow \mathcal{Y}$
  - Penalize errors for this mapping

Generalization of Regression

- Direct discriminative learning of $g : \mathcal{X} \to \mathcal{Y}$
  - Penalize errors for this mapping
- Two basic assumptions employed
  - Use of a compatibility function

$$f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

  - g takes the form of a decoding function

$$g(x) = \operatorname*{argmax}_y f(x, y)$$

Generalization of Regression

- Direct discriminative learning of $g : \mathcal{X} \rightarrow \mathcal{Y}$
  - Penalize errors for this mapping
- Two basic assumptions employed
  - Use of a compatibility function

$$f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$$

  - g takes the form of a decoding function

$$g(x) = \operatorname*{argmax}_{y} f(x, y)$$

  - linear w.r.t. *joint* kernel

$$f(x, y) = \langle w, \varphi(x, y) \rangle$$

Multi-Class Joint Feature Map

- Simple joint kernel map:
- define $\varphi_y(y_i)$ to be the vector with 1 in place of the current class, and 0 elsewhere

$$\varphi_y(y_i) = [0, \ldots, \underbrace{1}_{k\text{th position}}, \ldots, 0]^T$$

if y_i represents a sample that is a member of class k

Multi-Class Joint Feature Map

- Simple joint kernel map:
- define $\varphi_y(y_i)$ to be the vector with 1 in place of the current class, and 0 elsewhere

$$\varphi_y(y_i) = [0, \dots, \underbrace{1}_{k\text{th position}}, \dots, 0]^T$$

  if y_i represents a sample that is a member of class k
- $\varphi_x(x_i)$ can result from any kernel over $\mathcal{X}$:

$$k_x(x_i, x_j) = \langle \varphi_x(x_i), \varphi_x(x_j) \rangle$$

Multi-Class Joint Feature Map

- Simple joint kernel map:
- define $\varphi_y(y_i)$ to be the vector with 1 in place of the current class, and 0 elsewhere

$$\varphi_y(y_i) = [0, \dots, \underbrace{1}_{k\text{th position}}, \dots, 0]^T$$

  if y_i represents a sample that is a member of class k
- $\varphi_x(x_i)$ can result from any kernel over $\mathcal{X}$:

$$k_x(x_i, x_j) = \langle \varphi_x(x_i), \varphi_x(x_j) \rangle$$

- Set $\varphi(x_i, y_i) = \varphi_y(y_i) \otimes \varphi_x(x_i)$, where $\otimes$ represents the Kronecker product

Multiclass Perceptron

- Reminder: we want

$$\langle w, \varphi(x_i, y_i) \rangle > \langle w, \varphi(x_i, y) \rangle \qquad \forall y \neq y_i$$

Multiclass Perceptron

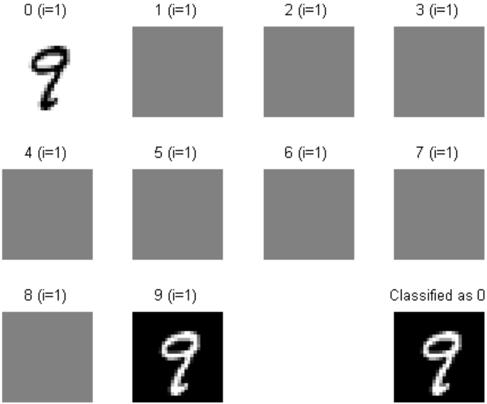
- Reminder: we want

$$\langle w, \varphi(x_i, y_i)\rangle > \langle w, \varphi(x_i, y)\rangle \qquad \forall y \neq y_i$$

- Example: perceptron training with a multiclass joint feature map
- Gradient of loss for example i is

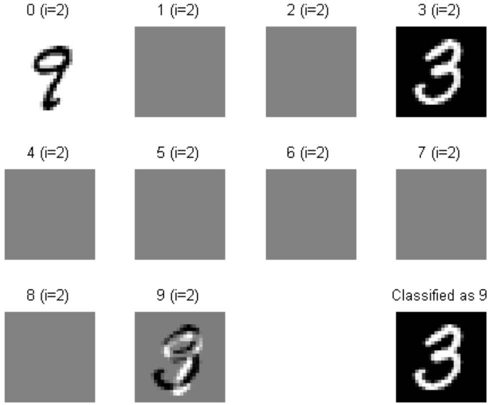
$$\partial_w \ell(x_i, y_i, w) = \begin{cases} 0 & \text{if } \langle w, \varphi(x_i, y_i)\rangle \geq \langle w, \varphi(x_i, y)\rangle \forall y \neq y_i \\ \max_{y \neq y_i} \varphi(x_i, y_i) - \varphi(x_i, y) & \text{otherwise} \end{cases}$$

Perceptron Training with Multiclass Joint Feature Map



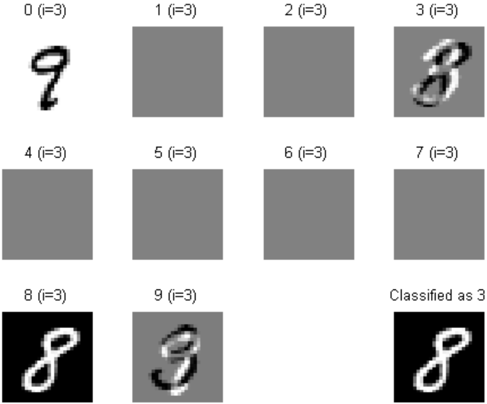
(Credit: Lyndsey Pickup)

Perceptron Training with Multiclass Joint Feature Map



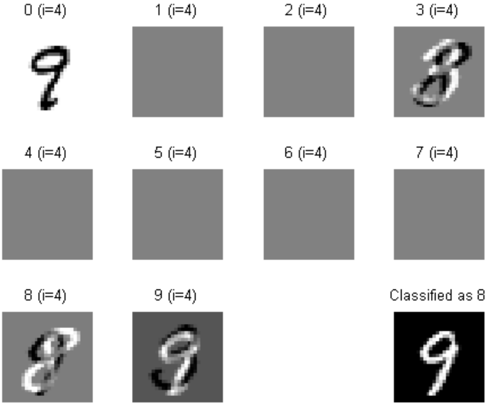
(Credit: Lyndsey Pickup)

Perceptron Training with Multiclass Joint Feature Map



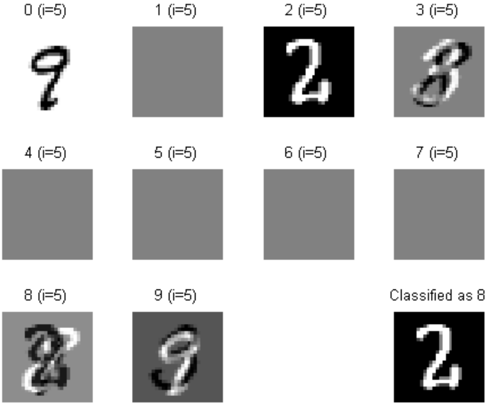
(Credit: Lyndsey Pickup)

Perceptron Training with Multiclass Joint Feature Map



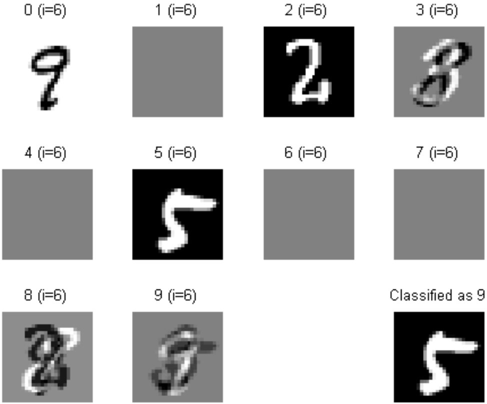
(Credit: Lyndsey Pickup)

Perceptron Training with Multiclass Joint Feature Map



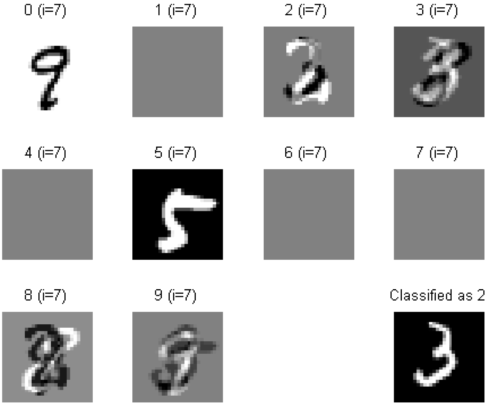
(Credit: Lyndsey Pickup)

Perceptron Training with Multiclass Joint Feature Map



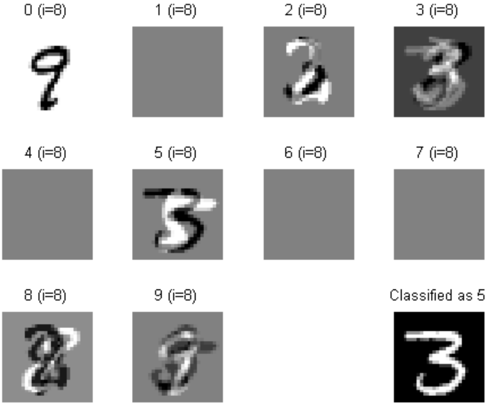
(Credit: Lyndsey Pickup)

Perceptron Training with Multiclass Joint Feature Map



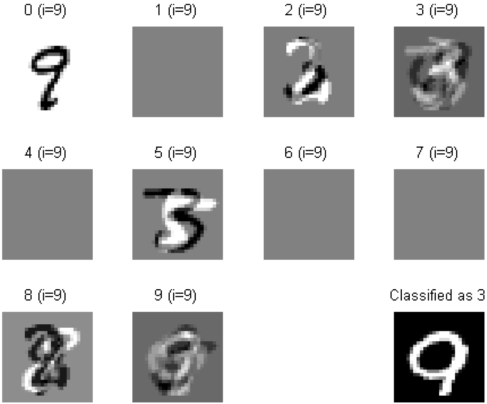
(Credit: Lyndsey Pickup)

Perceptron Training with Multiclass Joint Feature Map



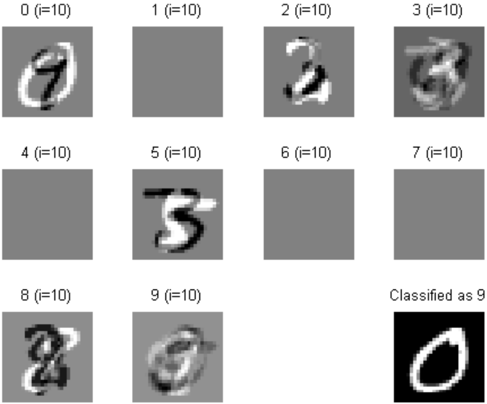
(Credit: Lyndsey Pickup)

Perceptron Training with Multiclass Joint Feature Map



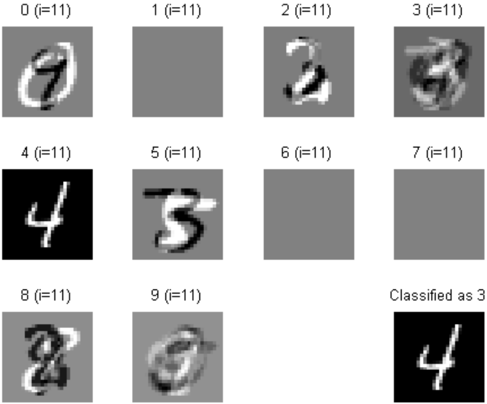
(Credit: Lyndsey Pickup)

Perceptron Training with Multiclass Joint Feature Map



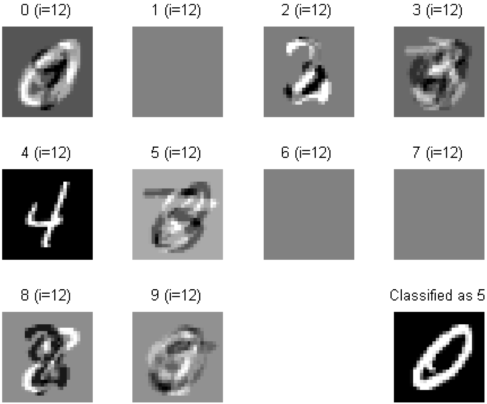
(Credit: Lyndsey Pickup)

Perceptron Training with Multiclass Joint Feature Map



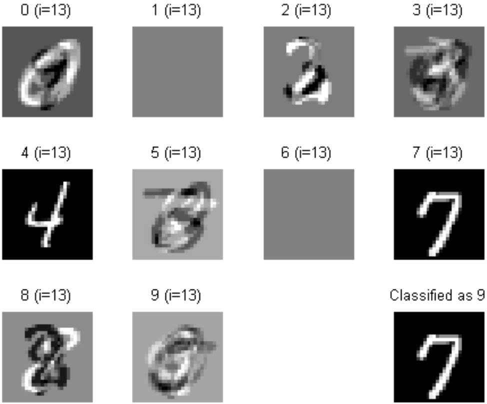
(Credit: Lyndsey Pickup)

Perceptron Training with Multiclass Joint Feature Map



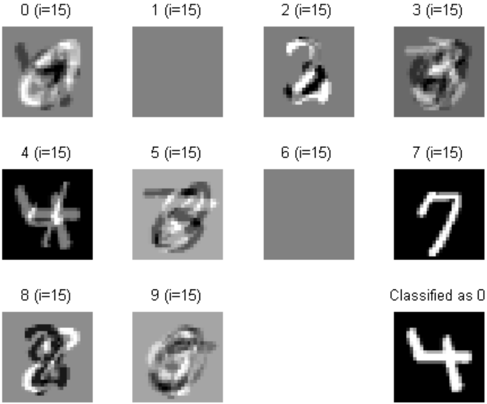
(Credit: Lyndsey Pickup)

Perceptron Training with Multiclass Joint Feature Map



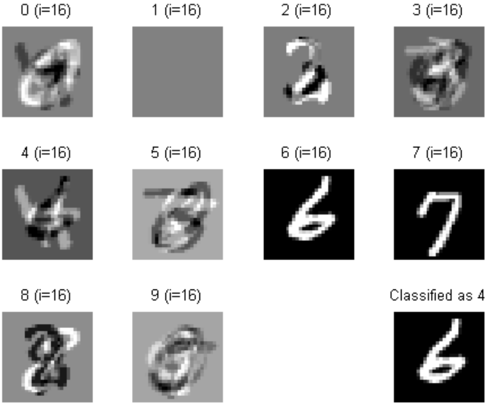
(Credit: Lyndsey Pickup)

Perceptron Training with Multiclass Joint Feature Map



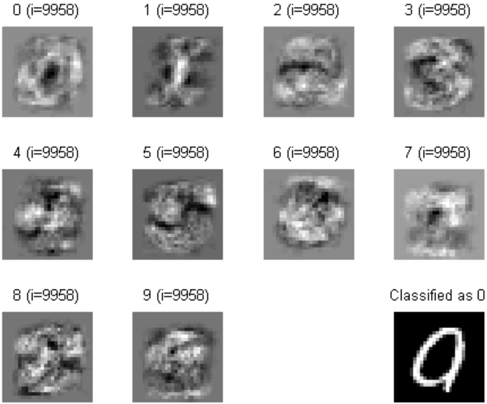
(Credit: Lyndsey Pickup)

Perceptron Training with Multiclass Joint Feature Map



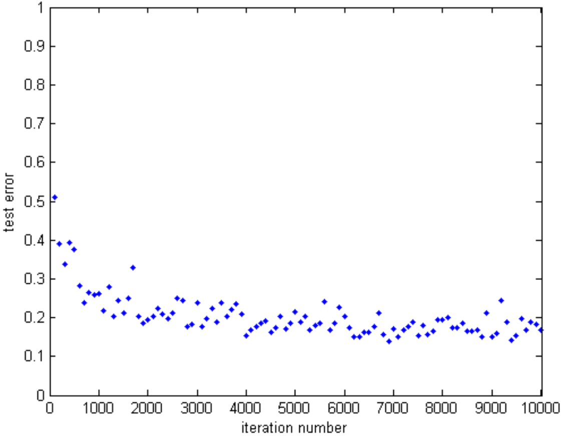
(Credit: Lyndsey Pickup)

Perceptron Training with Multiclass Joint Feature Map



Final result
(Credit: Lyndsey Pickup)

Perceptron Training with Multiclass Joint Feature Map



(Credit: Lyndsey Pickup)

Crammer & Singer Multi-Class SVM

- Instead of training using a perceptron, we can enforce a large margin and do a batch convex optimization:

$$\begin{aligned} \min_{w} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & \langle w, \varphi(x_i, y_i)\rangle - \langle w, \varphi(x_i, y)\rangle \geq 1 - \xi_i \qquad \forall y \neq y_i \end{aligned}$$

Crammer & Singer Multi-Class SVM

- Instead of training using a perceptron, we can enforce a large margin and do a batch convex optimization:

$$
\begin{aligned}
\min_{w} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \\
\text{s.t.} \quad & \langle w, \varphi(x_i, y_i)\rangle - \langle w, \varphi(x_i, y)\rangle \geq 1 - \xi_i \qquad \forall y \neq y_i
\end{aligned}
$$

- Can also be written only in terms of kernels

$$
w = \sum_{x}\sum_{y} \alpha_{xy}\varphi(x, y)
$$

- Can use a *joint kernel*

$$
k : \mathcal{X} \times \mathcal{Y} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}
$$

$$
k(x_i, y_i, x_j, y_j) = \langle \varphi(x_i, y_i), \varphi(x_j, y_j)\rangle
$$

Structured Output Support Vector Machines (SO-SVM)

- Frame structured prediction as a multiclass problem
  - predict a single element of $\mathcal{Y}$ and pay a penalty for mistakes

Structured Output Support Vector Machines (SO-SVM)

- Frame structured prediction as a multiclass problem
  - predict a single element of $\mathcal{Y}$ and pay a penalty for mistakes
- Not all errors are created equally
  - e.g. in an HMM making only one mistake in a sequence should be penalized less than making 50 mistakes

Structured Output Support Vector Machines (SO-SVM)

- Frame structured prediction as a multiclass problem
  - predict a single element of $\mathcal{Y}$ and pay a penalty for mistakes
- Not all errors are created equally
  - e.g. in an HMM making only one mistake in a sequence should be penalized less than making 50 mistakes
- Pay a loss proportional to the difference between true and predicted error (task dependent)

$$\Delta(y_i, y)$$

Margin Rescaling

Variant: Margin-Rescaled Joint-Kernel SVM for output space $\mathcal{Y}$ (Tsochantaridis et al., 2005)

- Idea: some *wrong* labels are *worse* than others: loss $\Delta(y_i, y)$
- Solve

$$\min_w \quad \|w\|^2 + C \sum_{i=1}^{n} \xi_i$$
$$\text{s.t.} \quad \langle w, \varphi(x_i, y_i) \rangle - \langle w, \varphi(x_i, y) \rangle \geq \Delta(y_i, y) - \xi_i \quad \forall y \in \mathcal{Y} \setminus \{y_i\}$$

- Classify new samples using $g : \mathcal{X} \to \mathcal{Y}$:

$$g(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} \langle w, \varphi(x, y) \rangle$$

Margin Rescaling

Variant: Margin-Rescaled Joint-Kernel SVM for output space $\mathcal{Y}$ (Tsochantaridis et al., 2005)

- Idea: some *wrong* labels are *worse* than others: loss $\Delta(y_i, y)$
- Solve

$$\min_{w} \quad \|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad \langle w, \varphi(x_i, y_i)\rangle - \langle w, \varphi(x_i, y)\rangle \geq \Delta(y_i, y) - \xi_i \quad \forall y \in \mathcal{Y} \setminus \{y_i\}$$

- Classify new samples using $g : \mathcal{X} \to \mathcal{Y}$:

$$g(x) = \operatorname*{argmax}_{y\in\mathcal{Y}} \langle w, \varphi(x, y)\rangle$$

- Another variant is *slack* rescaling (see Tsochantaridis et al., 2005)

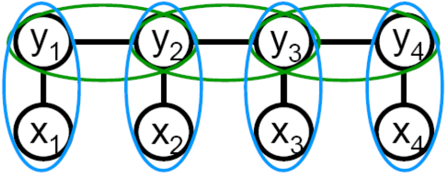
Label Sequence Learning

- For, e.g., handwritten character recognition, it may be useful to include a temporal model in addition to learning each character individually
- As a simple example take an HMM

Label Sequence Learning

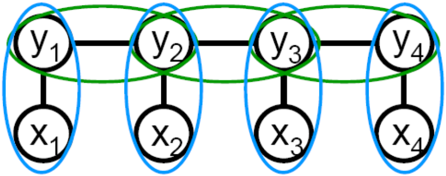
- For, e.g., handwritten character recognition, it may be useful to include a temporal model in addition to learning each character individually
- As a simple example take an HMM
- We need to model emission probabilities and transition probabilities
    - Learn these discriminatively

A Joint Kernel Map for Label Sequence Learning



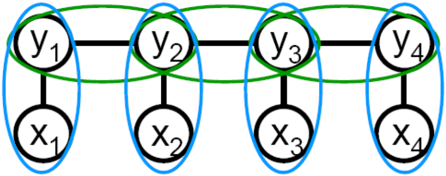
- Emissions (blue)

A Joint Kernel Map for Label Sequence Learning



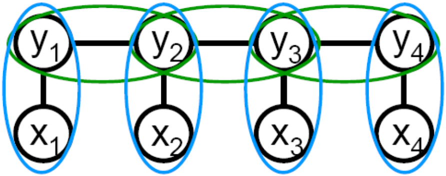
- Emissions (blue)
  - $f_e(x_i, y_i) = \langle w_e, \varphi_e(x_i, y_i) \rangle$

A Joint Kernel Map for Label Sequence Learning



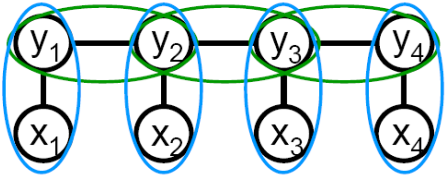
- Emissions (blue)
  - $f_e(x_i, y_i) = \langle w_e, \varphi_e(x_i, y_i) \rangle$
  - Can simply use the multi-class joint feature map for φ_e

A Joint Kernel Map for Label Sequence Learning



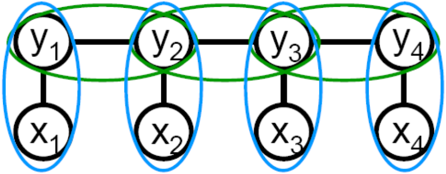
- Emissions (blue)
  - $f_e(x_i, y_i) = \langle w_e, \varphi_e(x_i, y_i) \rangle$
  - Can simply use the multi-class joint feature map for φ_e
- Transitions (green)

A Joint Kernel Map for Label Sequence Learning



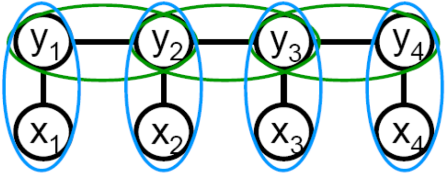
- Emissions (blue)
  - $f_e(x_i, y_i) = \langle w_e, \varphi_e(x_i, y_i) \rangle$
  - Can simply use the multi-class joint feature map for φ_e
- Transitions (green)
  - $f_t(x_i, y_i) = \langle w_t, \varphi_t(y_i, y_{i+1}) \rangle$
  - Can use $\varphi_t(y_i, y_{i+1}) = \varphi_y(y_i) \otimes \varphi_y(y_{i+1})$

A Joint Kernel Map for Label Sequence Learning (continued)



$$p(x, y) \quad \propto \quad \prod_i e^{f_e(x_i, y_i)} \prod_i e^{f_t(y_i, y_{i+1})} \quad \text{for an HMM}$$

A Joint Kernel Map for Label Sequence Learning (continued)



$$
\begin{aligned}
p(x, y) &\propto \prod_i e^{f_e(x_i, y_i)} \prod_i e^{f_t(y_i, y_{i+1})} \quad \text{for an HMM} \\
f(x, y) &= \sum_i f_e(x_i, y_i) + \sum_i f_t(y_i, y_{i+1}) \\
&= \langle w_e, \sum_i \varphi_e(x_i, y_i) \rangle + \langle w_t, \sum_i \varphi_t(y_i, y_{i+1}) \rangle
\end{aligned}
$$

Constraint Generation

$$\min_{w} \quad \|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \quad \langle w, \varphi(x_i, y_i)\rangle - \langle w, \varphi(x_i, y)\rangle \geq \Delta(y_i, y) - \xi_i \quad \forall y \in \mathcal{Y} \setminus \{y_i\}$$

Constraint Generation

$$\min_{w} \quad \|w\|^2 + C \sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \quad \langle w, \varphi(x_i, y_i) \rangle - \langle w, \varphi(x_i, y) \rangle \geq \Delta(y_i, y) - \xi_i \quad \forall y \in \mathcal{Y} \setminus \{y_i\}$$

- Initialize constraint set to be empty
- Iterate until convergence:
  - Solve optimization using current constraint set
  - Add maximially violated constraint for current solution

Constraint Generation with the Viterbi Algorithm

- To find the maximially violated constraint, we need to maximize w.r.t. y

$$\langle w, \varphi(x_i, y) \rangle + \Delta(y_i, y)$$

Constraint Generation with the Viterbi Algorithm

- To find the maximially violated constraint, we need to maximize w.r.t. y

$$\langle w, \varphi(x_i, y)\rangle + \Delta(y_i, y)$$

- For arbitrary output spaces, we would need to iterate over all elements in $\mathcal{Y}$

Constraint Generation with the Viterbi Algorithm

- To find the maximially violated constraint, we need to maximize w.r.t. y
$$\langle w, \varphi(x_i, y)\rangle + \Delta(y_i, y)$$

- For arbitrary output spaces, we would need to iterate over all elements in $\mathcal{Y}$
- For HMMs, $\max_y \langle w, \varphi(x_i, y)\rangle$ can be found using the Viterbi algorithm

Constraint Generation with the Viterbi Algorithm

- To find the maximially violated constraint, we need to maximize w.r.t. y

$$\langle w, \varphi(x_i, y)\rangle + \Delta(y_i, y)$$

- For arbitrary output spaces, we would need to iterate over all elements in $\mathcal{Y}$
- For HMMs, $\max_y \langle w, \varphi(x_i, y)\rangle$ can be found using the Viterbi algorithm
- It is a simple modification of this procedure to incorporate $\Delta(y_i, y)$ (Tsochantaridis et al., 2004)

Discriminative Training of Object Localization

- Structured output learning is not restricted to outputs specified by graphical models

Discriminative Training of Object Localization

- Structured output learning is not restricted to outputs specified by graphical models
- We can formulate object localization as a regression from an image to a bounding box

$$g : \mathcal{X} \rightarrow \mathcal{Y}$$

- $\mathcal{X}$ is the space of all images
- $\mathcal{Y}$ is the space of all bounding boxes

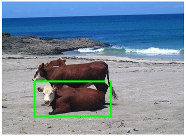
Joint Kernel between Images and Boxes: Restriction Kernel

- Note: $x|_y$ (the image restricted to the box region) is again an image.
- Compare two images with boxes by comparing the images within the boxes:

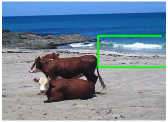
$$k_{joint}((x,y),(x',y')\ ) = k_{image}(x|_y, x'|_{y'},)$$

- Any common image kernel is applicable:
  - linear on cluster histograms: $k(h,h') = \sum_i h_i h'_i$,
  - χ^2-kernel: $k_{\chi^2}(h,h') = \exp\left(-\frac{1}{\gamma}\sum_i \frac{(h_i-h'_i)^2}{h_i+h'_i}\right)$
  - pyramid matching kernel, ...
- The resulting joint kernel is positive definite.

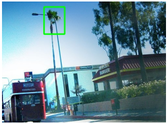
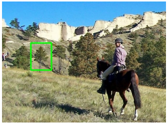
Restriction Kernel: Examples

$k_{joint}\Big($  ,  $\Big) = k\Big($  ,  $\Big)$

is large.

$k_{joint}\Big($  ,  $\Big) = k\Big($  ,  $\Big)$

is small.

$k_{joint}\Big($  ,  $\Big) = k\Big($  ,  $\Big)$

could also be large.

- Note: This behaves differently from the common tensor products

$$k_{joint}(\ (x,y),(x',y')\ ) \neq k(x,x')k(y,y'))\ !$$

Constraint Generation with Branch and Bound

- As before, we must solve

$$\max_{y \in \mathcal{Y}} \langle w, \varphi(x_i, y) \rangle + \Delta(y_i, y)$$

where

$$\Delta(y_i, y) = \begin{cases} 1 - \frac{\text{Area}(y_i \bigcap y)}{\text{Area}(y_i \bigcup y)} & \text{if } y_{i\omega} = y_\omega = 1 \\ 1 - \left(\frac{1}{2}(y_{i\omega} y_\omega + 1) \right) & \text{otherwise} \end{cases}$$

and $y_{i\omega}$ specifies whether there is an instance of the object at all present in the image

Constraint Generation with Branch and Bound

- As before, we must solve

$$\max_{y \in \mathcal{Y}} \langle w, \varphi(x_i, y) \rangle + \Delta(y_i, y)$$

where

$$\Delta(y_i, y) = \begin{cases} 1 - \frac{\text{Area}(y_i \bigcap y)}{\text{Area}(y_i \bigcup y)} & \text{if } y_{i\omega} = y_\omega = 1 \\ 1 - \left(\frac{1}{2}(y_{i\omega} y_\omega + 1)\right) & \text{otherwise} \end{cases}$$

and $y_{i\omega}$ specifies whether there is an instance of the object at all present in the image

- Solution: use branch-and-bound over the space of all rectangles in the image (Blaschko & Lampert, 2008)

Discriminative Training of Image Segmentation

- Frame discriminative image segmentation as learning parameters of a random field model

Discriminative Training of Image Segmentation

- Frame discriminative image segmentation as learning parameters of a random field model
- Like sequence learning, the problem decomposes over *cliques* in the graph

Discriminative Training of Image Segmentation

- Frame discriminative image segmentation as learning parameters of a random field model
- Like sequence learning, the problem decomposes over *cliques* in the graph
- Set the loss to the number of incorrect pixels

Constraint Generation with Graph Cuts

- As the graph is loopy, we cannot use Viterbi

Constraint Generation with Graph Cuts

- As the graph is loopy, we cannot use Viterbi
- Loopy belief propagation is approximate and can lead to poor learning performance for structured output learning of graphical models (Finley & Joachims, 2008)

Constraint Generation with Graph Cuts

- As the graph is loopy, we cannot use Viterbi
- Loopy belief propagation is approximate and can lead to poor learning performance for structured output learning of graphical models (Finley & Joachims, 2008)
- Solution: use graph cuts (Szummer et al., 2008)
- $\Delta(y_i, y)$ can be easily incorporated into the energy function

Summary of Structured Output Learning

- Structured output learning is the prediction of items in complex and interdependent output spaces

Summary of Structured Output Learning

- Structured output learning is the prediction of items in complex and interdependent output spaces
- We can train regressors into these spaces using a generalization of the support vector machine

Summary of Structured Output Learning

- Structured output learning is the prediction of items in complex and interdependent output spaces
- We can train regressors into these spaces using a generalization of the support vector machine
- We have shown examples for
  - Label sequence learning with Viterbi
  - Object localization with branch and bound
  - Image segmentation with graph cuts

Summary of Structured Output Learning

- Structured output learning is the prediction of items in complex and interdependent output spaces
- We can train regressors into these spaces using a generalization of the support vector machine
- We have shown examples for
  - Label sequence learning with Viterbi
  - Object localization with branch and bound
  - Image segmentation with graph cuts
- Questions?