MTTTS16 Learning from Multiple Sources

5 ECTS credits

Autumn 2019, University of Tampere
Lecturer: Jaakko Peltonen

Lecture 8: Multi-view learning for classification by **Co-training, part 2**

On this lecture:

- Part 1: More about Gaussian processes and Bayesian co-training
- Part 2: Co-training when views disagree (Detecting view disagreement)
- Part 3: Very briefly about semisupervised multi-task learning
- Part 4: Self-taught learning

Part 1:

Bayesian Co-training (in-depth explanation with more detail than in previous lectures)

Bayesian co-training

• The previous discussion on co-training did not present it as an integrated model of data: does not optimize a joint cost function for all views of all data

• There have been approaches called **co-regularization** that do optimize a joint cost function (not discussed here) but they still optimize one view at a time

• **Bayesian co-training**: an undirected graphical model for co-training. Maximum likelihood inference for that model is related to co-regularization.

• We will use a nonparametric **Gaussian process** approach to model input-output functions

Following the approach from Shipeng Yu, Balaji Krishnapuram, Romer Rosales, Harald Steck, R. Bharat Rao. Bayesian Co-Training. Journal of Machine Learning Research 12:2649-2680, 2011. Images from that paper.

Recap: Gaussian processes

- A **Gaussian process** is a prior over input-output functions
- A Gaussian process prior does not specify any parametric family for the functions, it only specifies how output values for two different input points are likely to be related.
- A Gaussian process is specified by a mean function and a covariance function
- Idea: for any two input points x, x', a Gaussian process prior says the output values f(x), f(x') jointly have a Gaussian distribution,
  whose mean is given by the mean function $\mu(x) = E[f(x)]$,
  and covariance is given by the covariance function
  $k(x, x') = E[(f(x) - \mu(x))(f(x') - \mu(x'))]$

- If the likelihood function is also Gaussian, then the posterior distribution over the input-output functions (after seeing the observations) is also a Gaussian process!

Following the approach from Kai Yu, Volker Tresp, and Anton Schwaighofer. Learning Gaussian Processes from Multiple Tasks. In Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), 2005.. Images from that paper.

Gaussian processes

- In Gaussian process based inference, the task is to compute the mean function and covariance function of the posterior distribution, given the prior and the observations.
- When the posterior distribution has been computed, it can be used to predict values of the output at new input points, as an expectation over the posterior.
- Gaussian process prediction gives both the prediction at the new point (= mean of the function value over the posterior) and the uncertainty about the prediction (= variance of the function value over the posterior)
- Gaussian process computation can be done in closed form if the prior and likelihood are simple.

Following the approach from Kai Yu, Volker Tresp, and Anton Schwaighofer. Learning Gaussian Processes from Multiple Tasks. In Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), 2005.. Images from that paper.

Gaussian Processes (GPs)

The gray line shows the mean function.

The light gray area shows the standard deviation (uncertainty).

This is the GP prior over functions before seeing any data.

Distribution of output values (distribution of input-output functions) shown on the vertical axis

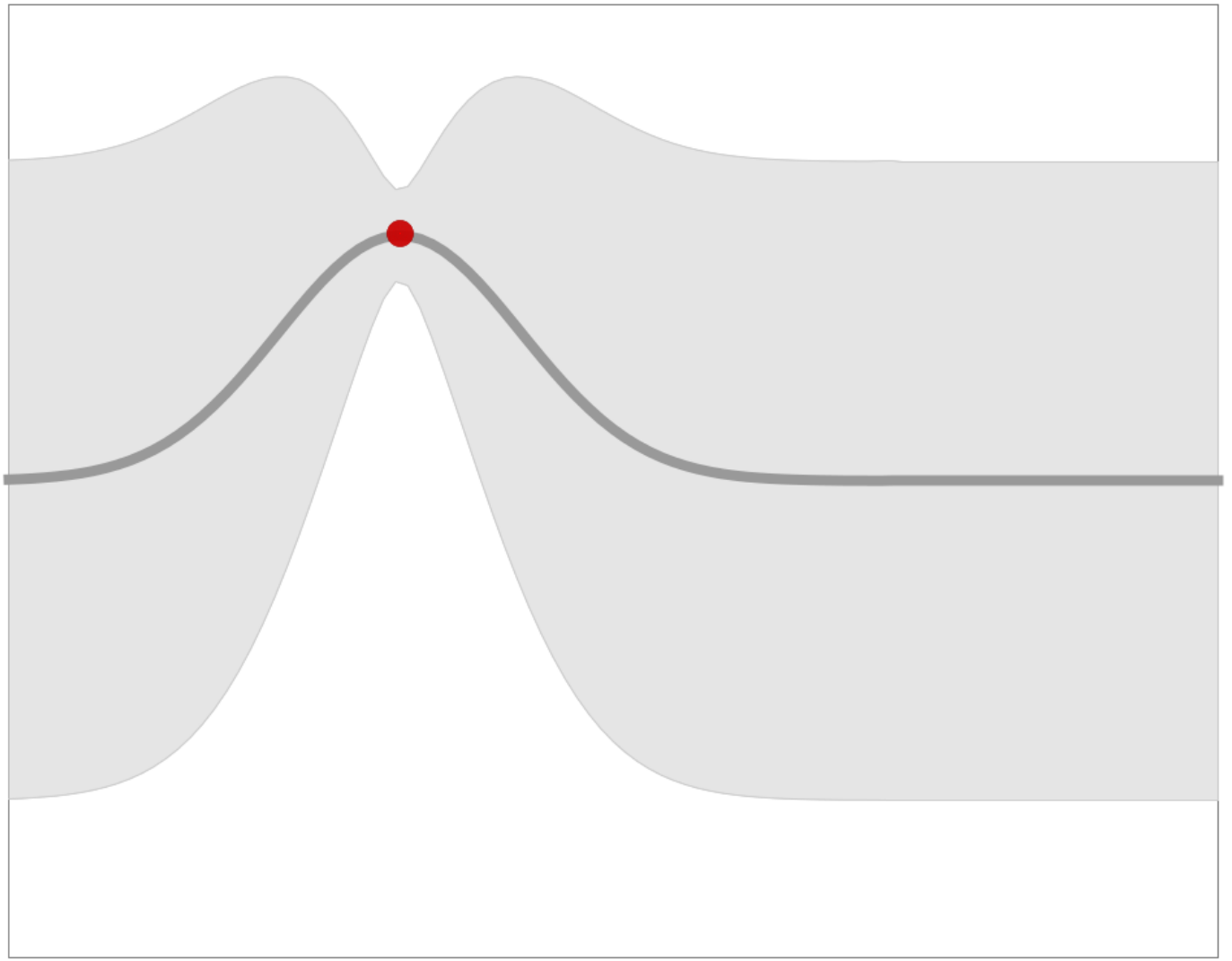
One-dimensional input values shown on the horizontal axis

Gaussian Processes (GPs)

The gray line shows the mean function.

The light gray area shows the standard deviation (uncertainty).

This is the GP **posterior** over functions after seeing one data point

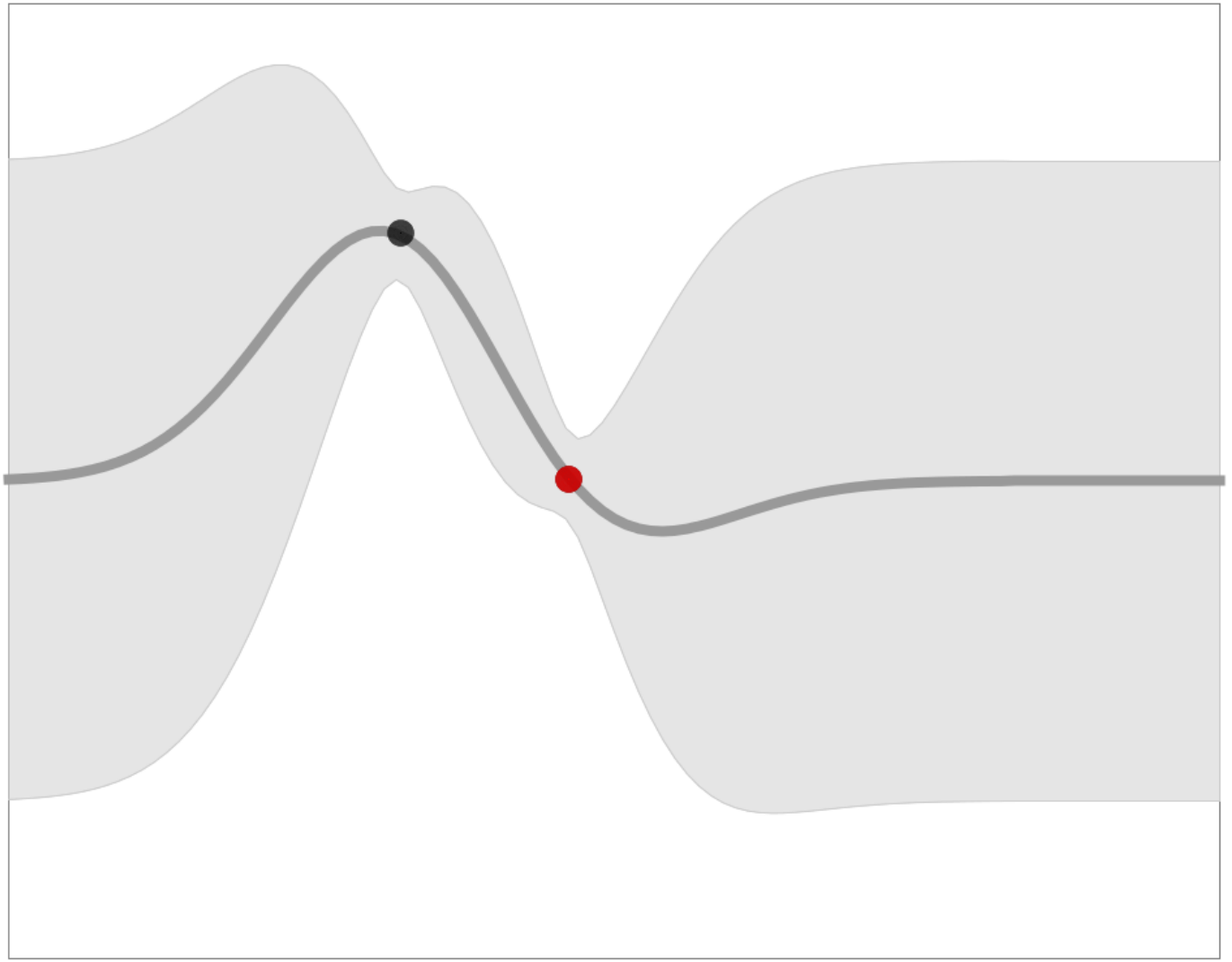


Gaussian Processes (GPs)

The gray line shows the mean function.

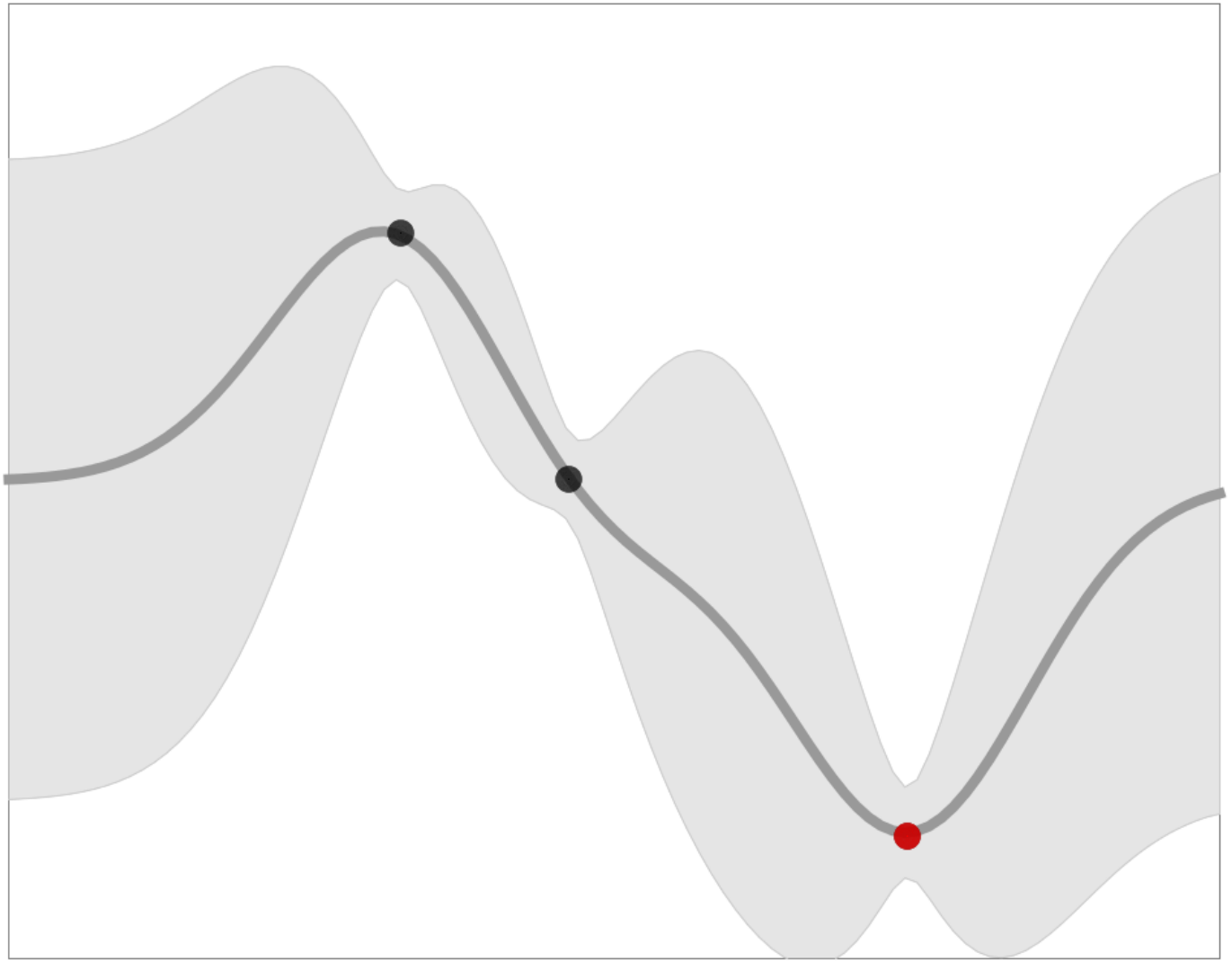
The light gray area shows the standard deviation (uncertainty).

This is the GP **posterior** over functions after seeing two data points



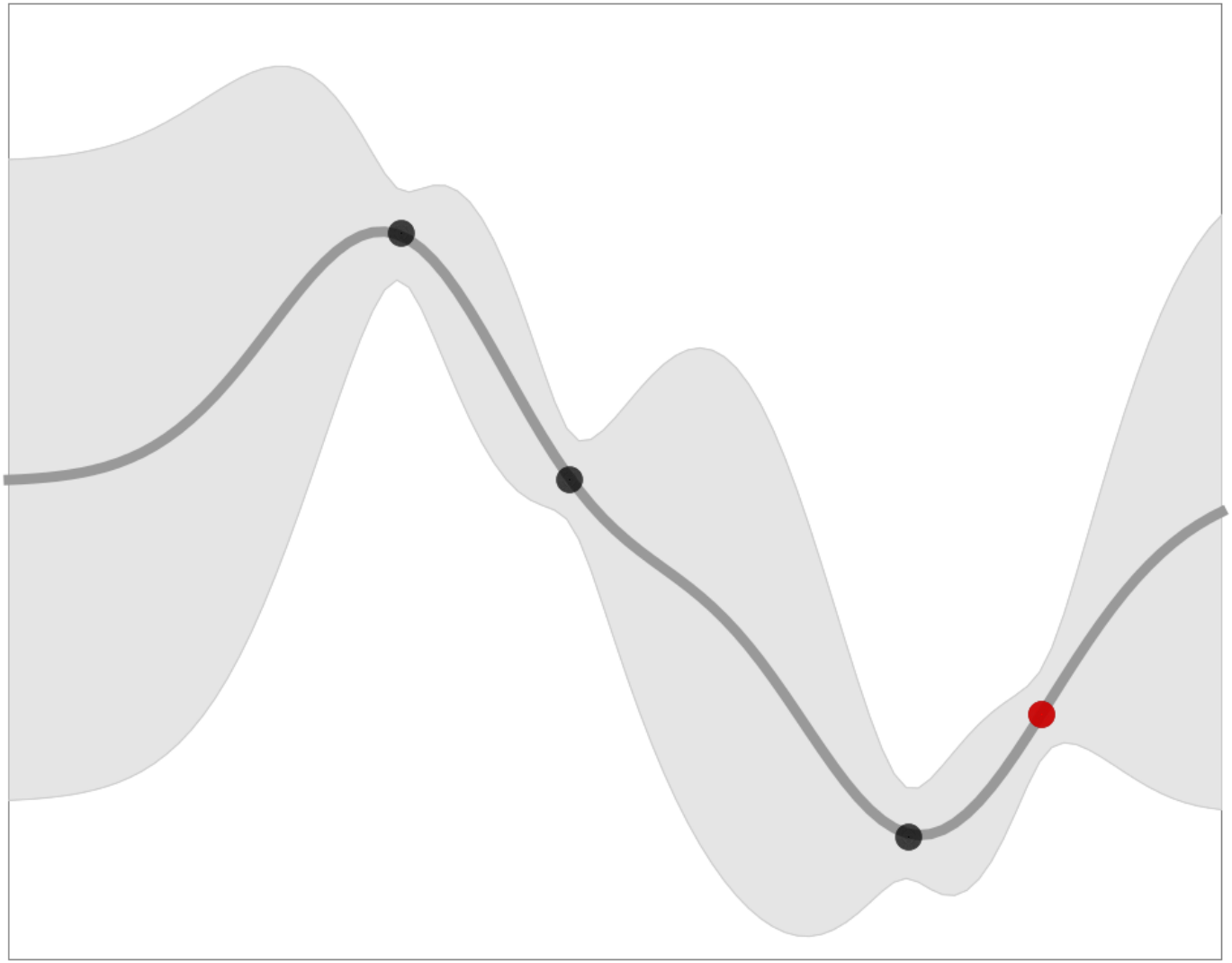
Gaussian Processes (GPs)

After seeing observations, posterior **uncertainty** about the function decreases at observations, and at **inputs correlated with the observation points**.

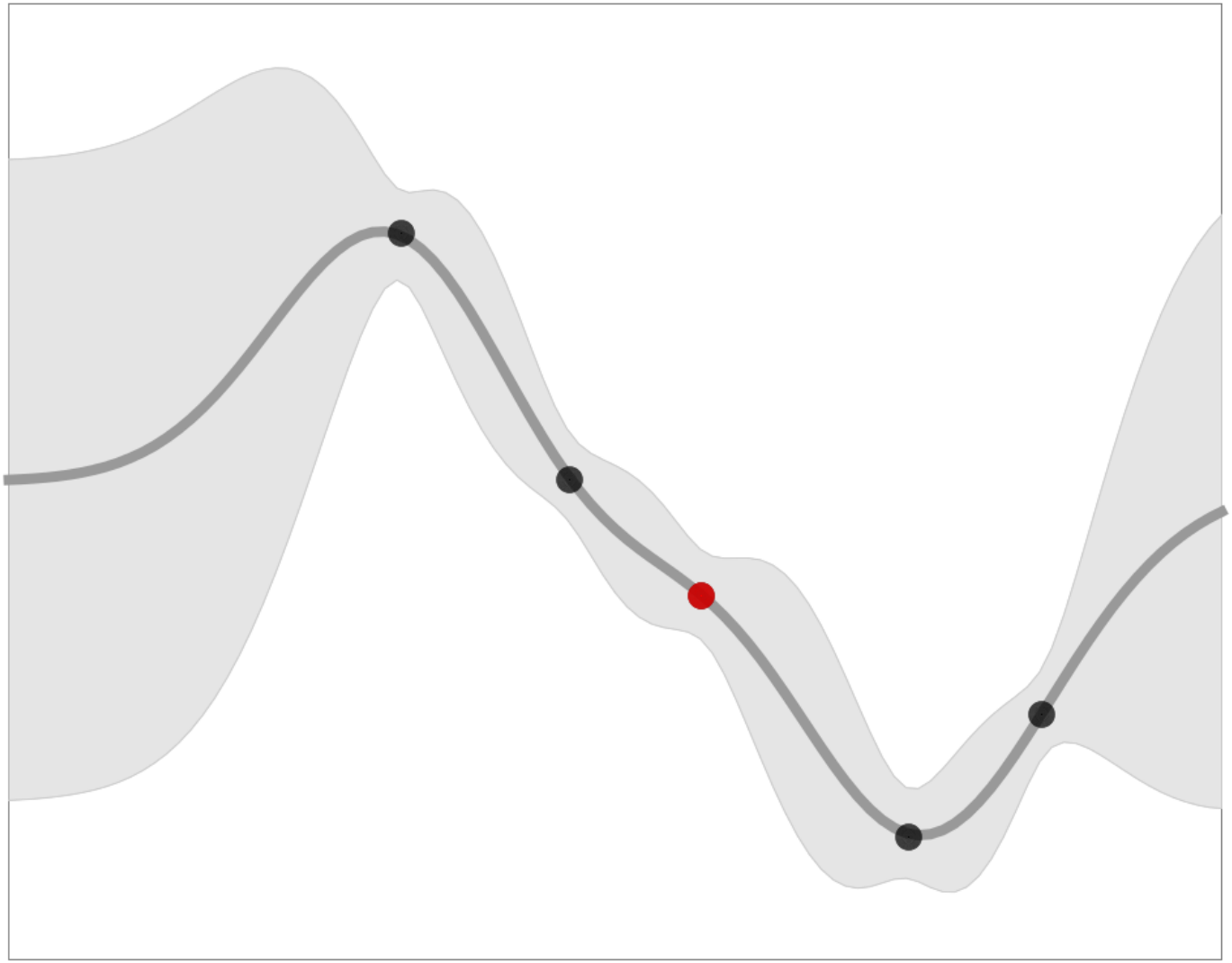


(Covariance function tells how much each pair of inputs is correlated over the possible functions)

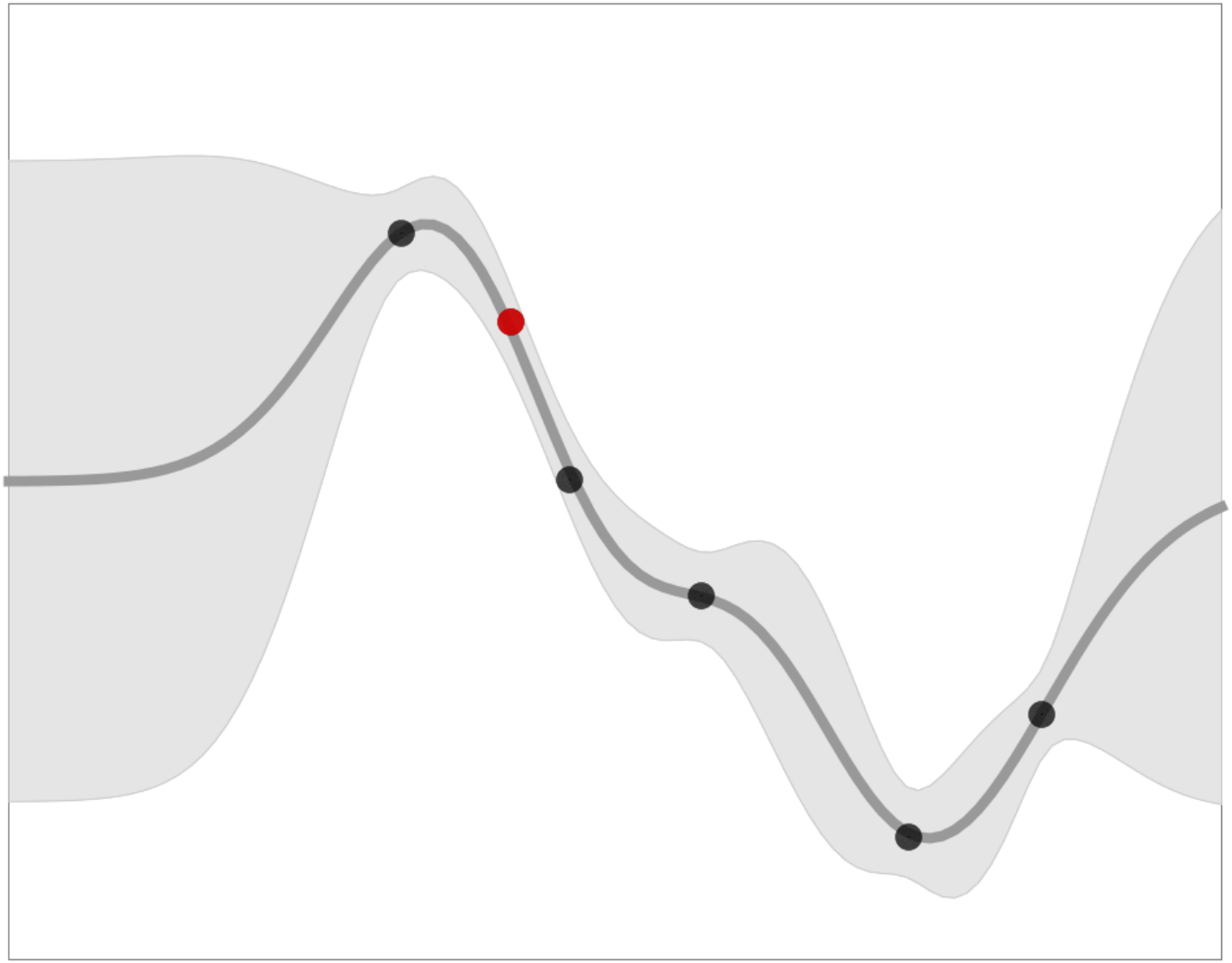
Gaussian Processes (GPs)



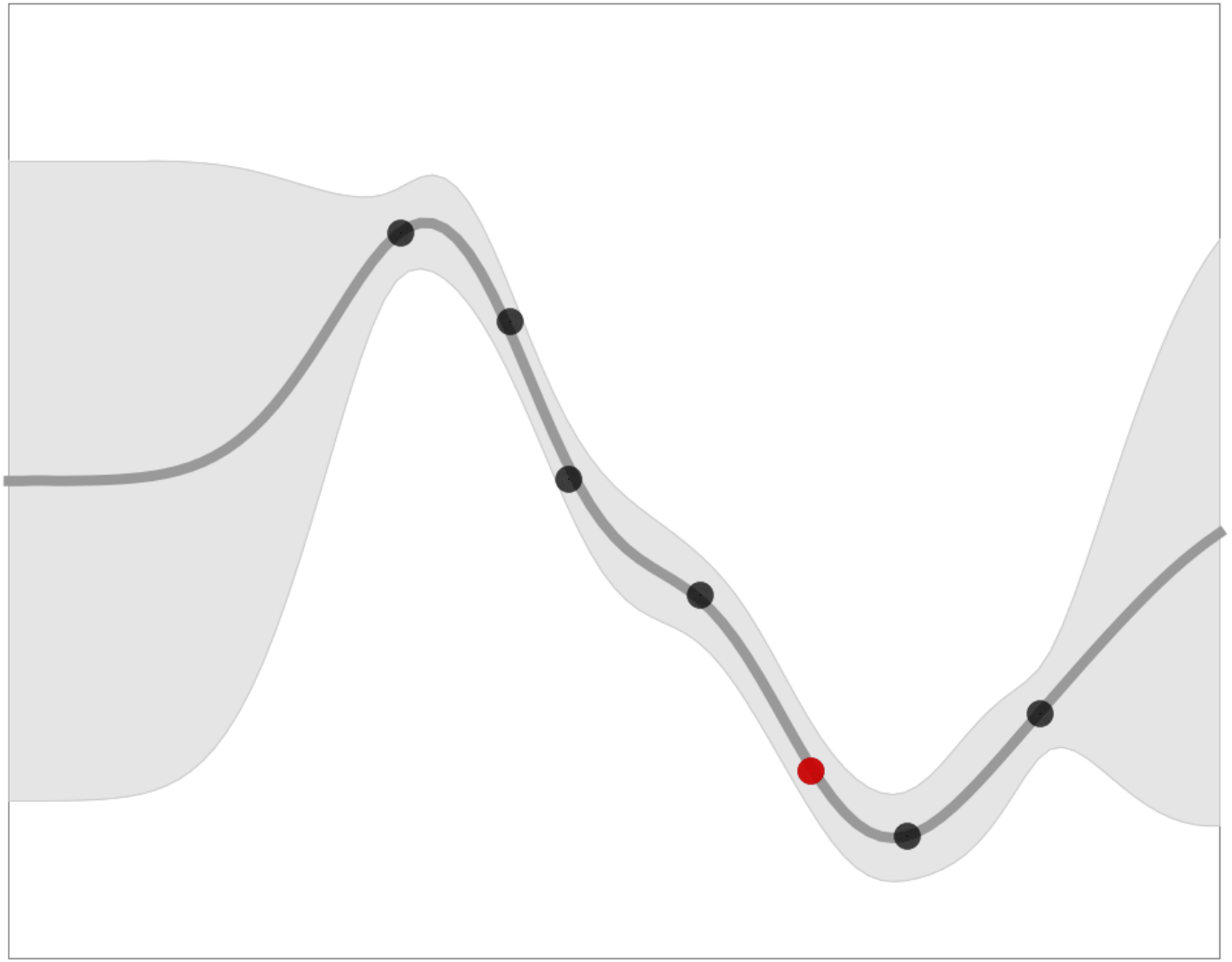
Gaussian Processes (GPs)



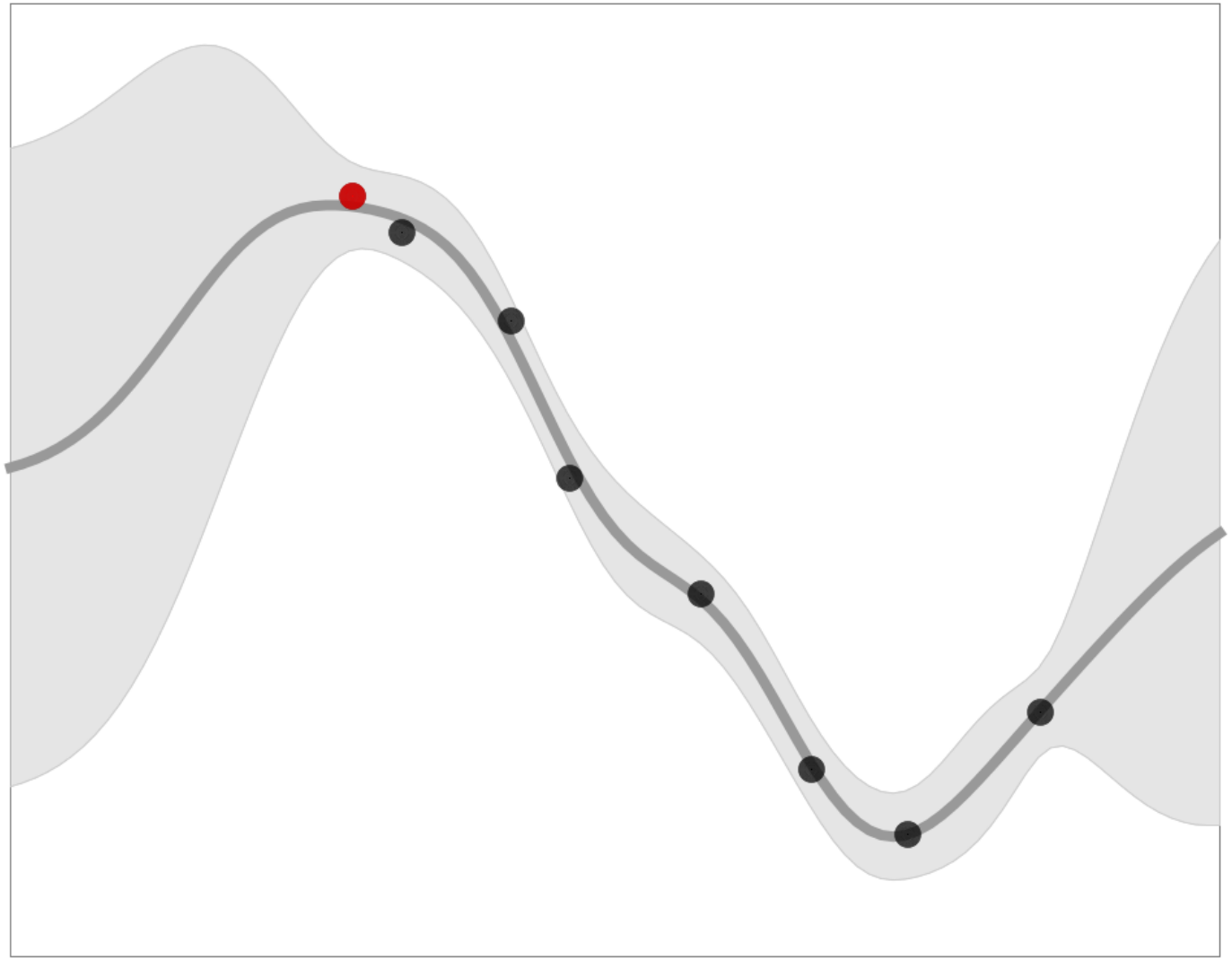
Gaussian Processes (GPs)



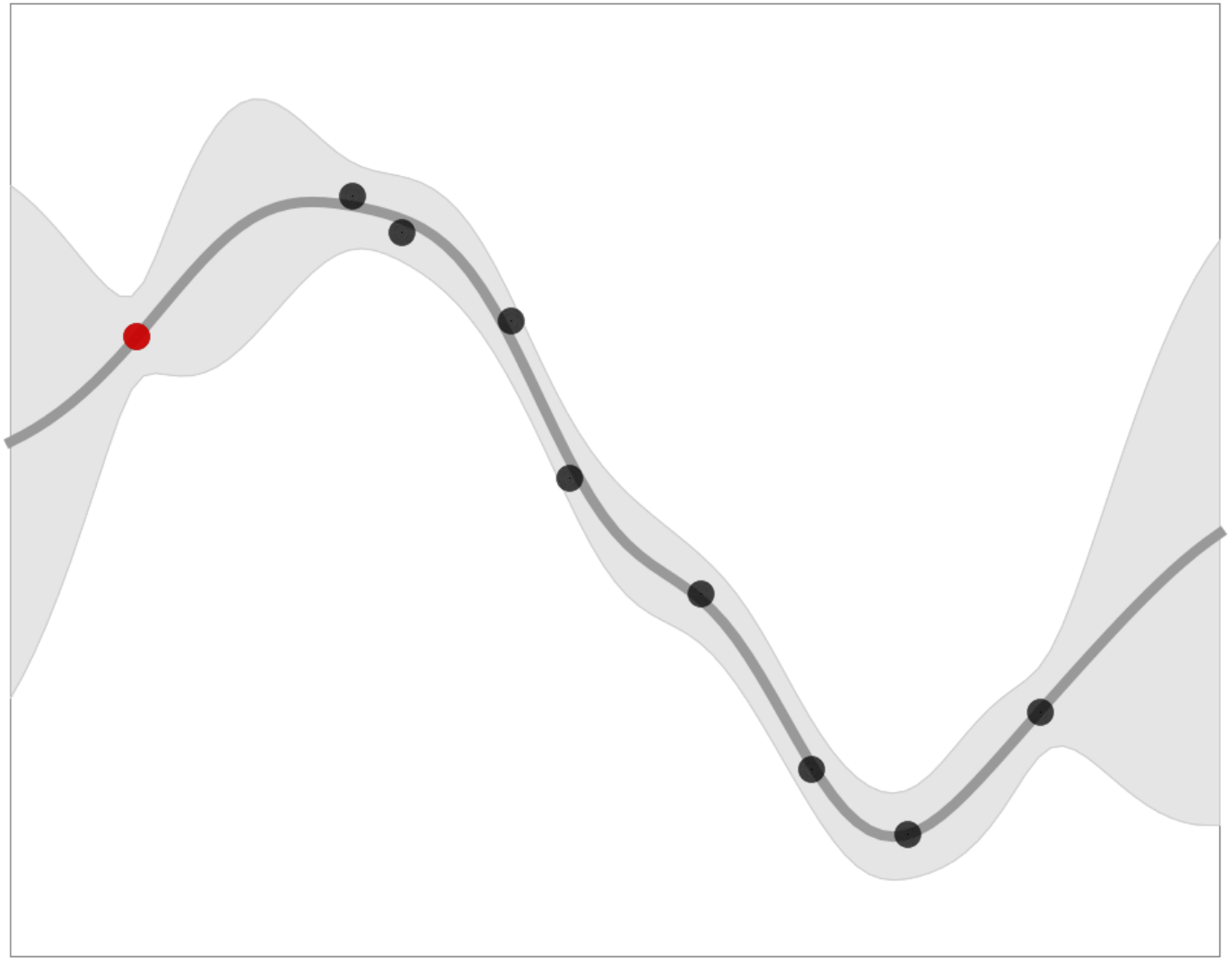
Gaussian Processes (GPs)



Gaussian Processes (GPs)



Gaussian Processes (GPs)



Gaussian processes with noiseless outputs

Assumes we directly observe the function values!

- The prediction model is $y = f(x)$
- Then the posterior given a data set D={(x,y)} is

$$p(y_{new}|x_{new}, D) = \int_f p(y_{new}|x_{new}, f)\, p(f|D)\, df$$

$$\sim \frac{1}{Z} \exp\left(-(y_{new} - \hat{y}_{x_{new},D})^2 / 2\sigma^2_{x_{new},D}\right)$$

- where $\hat{y}_{x_{new},D} = \boldsymbol{k}^T_{x_{new},D} \boldsymbol{K}_D^{-1} \boldsymbol{y}_D$, $\boldsymbol{K}_D$ is the covariance matrix of the observed data D (evaluated by computing the covariance function between all pairs of the observed data), $\boldsymbol{k}_{x_{new},D} = [k(x_{new}, x_1), \ldots, k(x_{new}, x_N)]^T$ is the covariance function computed between the new input point and all observed input points, and $\boldsymbol{y}_D = [y_1, \ldots, y_N]^T$ is the set of observed output values
- Similarly, the variance (uncertainty) at the new input point is

$$\sigma^2_{x_{new},D} = k(x_{new}, x_{new}) - \boldsymbol{k}^T_{x_{new},D} \boldsymbol{K}_D^{-1} \boldsymbol{k}_{x_{new},D}$$

- The noisy predictions are very similar, details on the next slide

Following the approach from Kai Yu, Volker Tresp, and Anton Schwaighofer. Learning Gaussian Processes from Multiple Tasks. In Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), 2005.. Images from that paper.

Gaussian processes with noisy outputs

- The prediction model with noise is $y=f(x)+e$ where e is independent Gaussian noise
- Then the posterior given a data set D={(x,y)} is

$$p(y_{new}|x_{new},D)=\int_f p(y_{new}|x_{new},f)\,p(f|D)\,df$$

$$\sim\frac{1}{Z}\exp\left(-(y_{new}-\hat{y}_{x_{new},D})^2/2\,\sigma^2_{x_{new},D}\right)$$

- where $\hat{y}_{x_{new},D}=\boldsymbol{k}^T_{x_{new},D}\left(\boldsymbol{K}_D+\sigma_n^2\boldsymbol{I}\right)^{-1}\boldsymbol{y}_D$, $\boldsymbol{K}_D$ is the covariance matrix of the observed data D (evaluated by computing the covariance function between all pairs of the observed data), $\boldsymbol{k}_{x_{new},D}=[k(x_{new},x_1),\ldots,k(x_{new},x_N)]^T$ is the covariance function computed between the new input point and all observed input points, and $\boldsymbol{y}_D=[y_1,\ldots,y_N]^T$ is the set of observed output values
- Similarly, the variance (uncertainty) at the new input point is $\sigma^2_{x_{new},D}=k(x_{new},x_{new})-\boldsymbol{k}^T_{x_{new},D}\left(\boldsymbol{K}_D+\sigma_n^2\boldsymbol{I}\right)^{-1}\boldsymbol{k}_{x_{new},D}$

Following the approach from Kai Yu, Volker Tresp, and Anton Schwaighofer. Learning Gaussian Processes from Multiple Tasks. In Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), 2005.. Images from that paper.

Bayesian co-training

- We have m different views, *n* samples, sample *i* denoted as

$$\mathbf{x}_i \triangleq (\mathbf{x}_i^{(1)}, \ldots, \mathbf{x}_i^{(m)})$$

- All samples of a particular view j denoted as

$$\mathbf{x}^{(j)} \triangleq (\mathbf{x}_1^{(j)}, \ldots, \mathbf{x}_n^{(j)})$$

- Outputs of all samples denoted as $\mathbf{y} = [y_1, \ldots, y_n]^\top$

- We will use a Gaussian process based prediction for each view. Each view j has an underlying function f_j that predicts the output value for the sample as $f_j(x_i^{(j)})$ based on input features of that view only. The function has a Gaussian process prior:

$$f_j \sim \mathcal{GP}(0, \kappa_j)$$

- The label y should depend on values of all the latent functions. How to make this explicit in a graphical model?

Following the approach from Shipeng Yu, Balaji Krishnapuram, Romer Rosales, Harald Steck, R. Bharat Rao. Bayesian Co-Training. Journal of Machine Learning Research 12:2649-2680, 2011. Images from that paper.

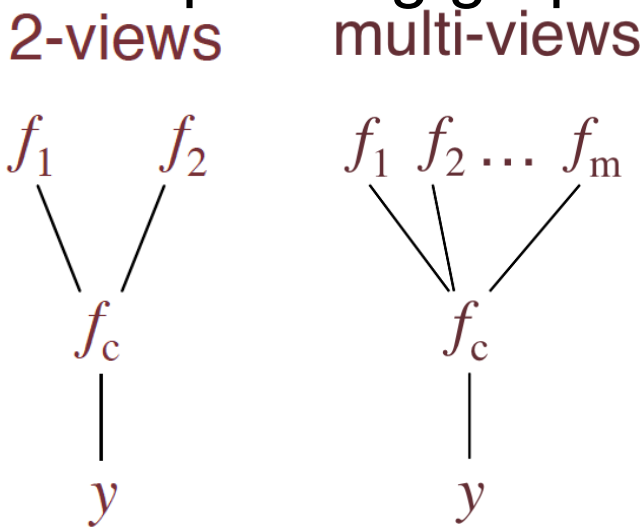
Bayesian co-training

- Idea: define a consensus function f_c that combines information from the individual functions, make the label depend on that alone.
- For a single sample, the joint distribution of the output value ans the underlying functions can be written as:

$$p(y, f_c, f_1, \ldots, f_m) = \frac{1}{Z}\Psi(y, f_c)\prod_{j=1}^{m}\Psi(f_j)\Psi(f_j, f_c)$$

where Ψ are some *potential functions* (nonnegative functions that can be suitably normalized to yield a probability distribution)

- corresponding graphical model:



Label depends only on consensus function

Individual functions depend on each other only through the consensus function

Following the approach from Shipeng Yu, Balaji Krishnapuram, Romer Rosales, Harald Steck, R. Bharat Rao. Bayesian Co-Training. Journal of Machine Learning Research 12:2649-2680, 2011. Images from that paper.

Bayesian co-training

• For n samples: let $\mathrm{f}_j = \{f_j(\mathrm{x}_i^{(j)})\}_{i=1}^n$ be the function values for the j:th view and and $\mathrm{f}_c = \{f_c(\mathrm{x}_i)\}_{i=1}^n$ be the consensus function values. Then probability of data and latent functions factorizes as:

$$p(\mathrm{y}, \mathrm{f}_c, \mathrm{f}_1, \ldots, \mathrm{f}_m) = \frac{1}{Z} \prod_{i=1}^{n} \psi(y_i, f_c(\mathrm{x}_i)) \prod_{j=1}^{m} \psi(\mathrm{f}_j) \psi(\mathrm{f}_j, \mathrm{f}_c)$$

• When a GP prior is used for each function, the **within-view potential** (which defines dependencies within each view) can be defined as:

$$\psi(\mathrm{f}_j) = \exp\left(-\frac{1}{2}\mathrm{f}_j^\top \mathrm{K}_j^{-1} \mathrm{f}_j\right), \quad \mathrm{K}_j(\mathrm{x}_k, \mathrm{x}_\ell) = \kappa_j(\mathrm{x}_k^{(j)}, \mathrm{x}_\ell^{(j)})$$

where K_j is the covariance matrix of the jth view

$$p(\{y_i, f_c(x_i), f_1(x_i), \ldots, f_m(x_i)\}_{i=1}^n)$$

$$= \frac{1}{Z}\left[\prod_{i=1}^{n} \Psi(y, f_c)\right] \prod_{j=1}^{m} \left[\prod_{i=1}^{n} \Psi(f_j(x_i^{(j)})) \Psi(f_j(x_i^{(j)}), f_c(x_i))\right]$$

Following the approach from Shipeng Yu, Balaji Krishnapuram, Romer Rosales, Harald Steck, R. Bharat Rao. Bayesian Co-Training. Journal of Machine Learning Research 12:2649-2680, 2011. Images from that paper.

Bayesian co-training

• The **consensus potential** defines relationship between each view and the consensus function, and can be defined as

$$\psi(\mathrm{f}_j, \mathrm{f}_c) = \exp\left(-\frac{\|\mathrm{f}_j - \mathrm{f}_c\|^2}{2\sigma_j^2}\right)$$

(it can be shown that this also corresponds to a Gaussian prior for the consensus function where the prior mean is the average of the individual functions.)

• The **output potential** describes relationship between the consensus function and the output, and can be defined as

$$\psi(y_i, f(\mathrm{x}_i)) = \begin{cases} \exp(-\frac{1}{2\sigma^2}\|y_i - f(\mathrm{x}_i)\|^2) & \text{for regression,} \\ \lambda(y_i f(\mathrm{x}_i)) & \text{for classification.} \end{cases}$$

Following the approach from Shipeng Yu, Balaji Krishnapuram, Romer Rosales, Harald Steck, R. Bharat Rao. Bayesian Co-Training. Journal of Machine Learning Research 12:2649-2680, 2011. Images from that paper.

Bayesian co-training

- The previous likelihood assumed all samples have labels. When some samples are unlabeled (n_l labeled samples, n_u unlabeled samples), the likelihood becomes:

$$p(\mathrm{y}_l, \mathrm{f}_c, \mathrm{f}_1, \ldots, \mathrm{f}_m) = \frac{1}{Z} \prod_{i=1}^{n_l} \psi(y_i, f_c(\mathbf{x}_i)) \prod_{j=1}^{m} \psi(\mathrm{f}_j) \psi(\mathrm{f}_j, \mathrm{f}_c)$$

(where output potentials computed only for labeled samples, but within-view functions&potentials and consensus function&potential computed for all samples)

- Inference: the standard task is to predict labels, that is, compute $p(\mathrm{y})$ for a new input given training data

- Try to integrate out some of the functions

Following the approach from Shipeng Yu, Balaji Krishnapuram, Romer Rosales, Harald Steck, R. Bharat Rao. Bayesian Co-Training. Journal of Machine Learning Research 12:2649-2680, 2011. Images from that paper.

Bayesian co-training

- Idea 1: try to integrate out the consensus function. It can be shown

$$p(\mathbf{f}_1,\ldots,\mathbf{f}_m) = \frac{1}{Z}\exp\left\{-\frac{1}{2}\sum_{j=1}^{m}\mathbf{f}_j^\top \mathbf{K}_j^{-1}\mathbf{f}_j - \frac{1}{2}\sum_{j<k}\left[\frac{\|\mathbf{f}_j-\mathbf{f}_k\|^2}{\sigma_j^2\sigma_k^2}\Big/\sum_\ell \frac{1}{\sigma_\ell^2}\right]\right\}$$

- This means the functions together have a GP prior

$$(\mathbf{f}_1,\ldots,\mathbf{f}_m) \sim \mathcal{N}(0,\Lambda^{-1})$$

where the inverse covariance has a block structure: given two tasks j, j'

$$\Lambda(j,j) = \mathbf{K}_j^{-1} + \frac{1}{\sum_\ell \frac{1}{\sigma_\ell^2}}\sum_{k\neq j}\frac{1}{\sigma_j^2\sigma_k^2}\mathbf{I}, \quad \Lambda(j,j') = -\frac{1}{\sum_\ell \frac{1}{\sigma_\ell^2}}\frac{1}{\sigma_j^2\sigma_{j'}^2}\mathbf{I}, \quad j'\neq j$$

and for regression the joint probability with output labels is

$$p(\mathbf{y},\mathbf{f}_1,\ldots,\mathbf{f}_m) = \frac{1}{Z}\exp\left\{-\frac{1}{2\rho\sigma^2}\sum_j\frac{\sum_{i=1}^n(y_i-f_j(\mathbf{x}_i))^2}{\sigma_j^2} - \frac{1}{2}\sum_j \mathbf{f}_j^\top\mathbf{K}_j^{-1}\mathbf{f}_j - \frac{1}{2\rho}\sum_{j<k}\frac{\|\mathbf{f}_j-\mathbf{f}_k\|^2}{\sigma_j^2\sigma_k^2}\right\}$$

- Inference could proceed from there but would need to infer a lot of functions.

Following the approach from Shipeng Yu, Balaji Krishnapuram, Romer Rosales, Harald Steck, R. Bharat Rao. Bayesian Co-Training. Journal of Machine Learning Research 12:2649-2680, 2011. Images from that paper.

Bayesian co-training

• Idea 2: try to integrate out the individual functions, so that the consensus function remains. It can be shown its marginal is

$$p(\mathrm{f}_c) = \mathcal{N}(0, \mathrm{K}_c)$$

where $\mathrm{K}_c = \left[\sum_j (\mathrm{K}_j + \sigma_j^2 \mathrm{I})^{-1} \right]^{-1}$ is called the co-training kernel

• the multi-view learning task essentially becomes a single-view learning task where only one function is learned

• Because of the matrix inverse, each element in the resulting co-training kernel depends on all elements of the original kernel values of each view (between all labeled and unlabeled samples)

• *Thus Bayesian co-training is equivalent to single-view learning with a specially designed (non-stationary) kernel*.
• The co-training kernel depends only on inputs, not labels, and can be computed over both labeled+unlabeled samples

Following the approach from Shipeng Yu, Balaji Krishnapuram, Romer Rosales, Harald Steck, R. Bharat Rao. Bayesian Co-Training. Journal of Machine Learning Research 12:2649-2680, 2011. Images from that paper.

Reminder: Gaussian processes

- If the prediction model is noise-free, $y=f(x)+e$,
- Then the posterior given a data set D={(x,y)} is

$$p(y|x,D)=\int_f p(y|x,f)\,p(f|D)\,df$$

$$\sim\frac{1}{Z}\exp\left(-(y-\hat{y}_{x,D})^2/2\,\sigma^2_{x,D}\right)$$

- where $\hat{y}_{x,D}=\boldsymbol{k}^T_{x,D}\boldsymbol{K}_D^{-1}\boldsymbol{y}_D$, $\boldsymbol{K}_D$ is the covariance matrix of the observed data D (evaluated by computing the covariance function between all pairs of the observed data), $\boldsymbol{k}_{x,D}=[k(x,x_1),\ldots,k(x,x_N)]^T$ is the covariance function computed between the new input point and all observed input points, and $\boldsymbol{y}_D=[y_1,\ldots,y_N]^T$ is the set of observed output values
- Similarly, the variance (uncertainty) at the new input point is $\sigma^2{}_{x,D}=k(x,x)-\boldsymbol{k}^T_{x,D}\boldsymbol{K}_D^{-1}\boldsymbol{k}_{x,D}$

Following the approach from Kai Yu, Volker Tresp, and Anton Schwaighofer. Learning Gaussian Processes from Multiple Tasks. In Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), 2005.. Images from that paper.

Bayesian co-training

- The standard GP inference on the previous slide can be used to predict function values given the observations. The equations are the same, but the kernel is now the co-training kernel.
- Note! The co-training kernel involves matrix inverses. It cannot be computed element-by-element separately: it can only be computed as part of a full-sized square matrix.
- This means that the kernel elements for the training samples depend on **what new data will be available**!
- This co-training can only be used in a **transductive** setting, where input values are known both for old and test samples during training, and the task is to predict the unknown test labels
- Thus we first compute a big kernel K_c over all data (test, labeled training, unlabeled training).
- Then $\boldsymbol{K}_D$ is obtained by leaving out rows&columns of K_c corresponding to test+unlabeled training samples. Similarly $\boldsymbol{k}_{x,D}$ is obtained by leaving out all terms except new-to-old crossterms

Following the approach from Shipeng Yu, Balaji Krishnapuram, Romer Rosales, Harald Steck, R. Bharat Rao. Bayesian Co-Training. Journal of Machine Learning Research 12:2649-2680, 2011. Images from that paper.

Bayesian co-training

- Thus we first compute a big kerne K_c over all data (test, labeled training, unlabeled training).
- Then is obtained by leaving out rows&columns of K_c corresponding to test+unlabeled training samples. Similarly $\boldsymbol{k}_{x,D}$ is obtained by leaving out all terms except new-to-old crossterms
- Then we apply the standard GP inference equations

- Hyperparameters: each view has a hyperparameters σ_j^2 which tells how much the view can deviate from the consensus
- The hyperparameters can be optimized to maximize the marginal likelihood of data given the hyperparameters: given a vector y_l of output values, their marginal likelihood is

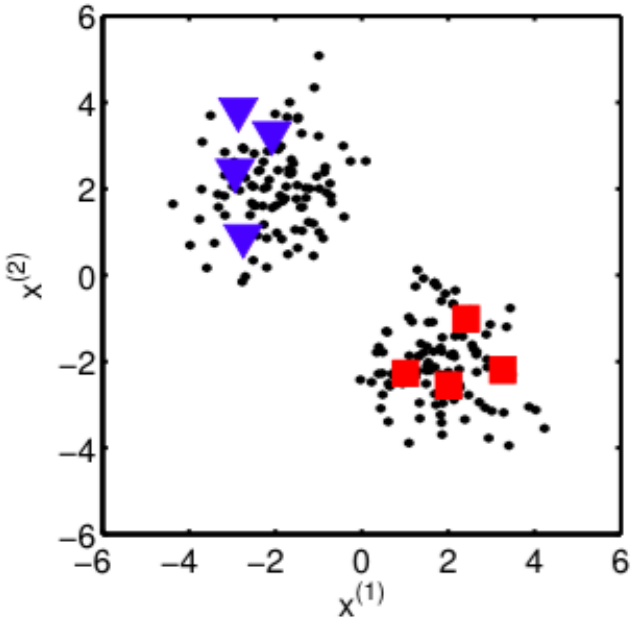
$$\mathcal{L} = -\frac{1}{2}\mathrm{y}_l^\top \mathrm{G}^{-1}\mathrm{y}_l - \frac{1}{2}\log\det\mathrm{G} - \frac{n}{2}\log 2\pi$$

where $\mathrm{G} \triangleq \mathrm{K}_c(1:n_l, 1:n_l) + \sigma^2\mathrm{I}$ and the likelihood can be optimized by gradient methods with respect to hyperparameters

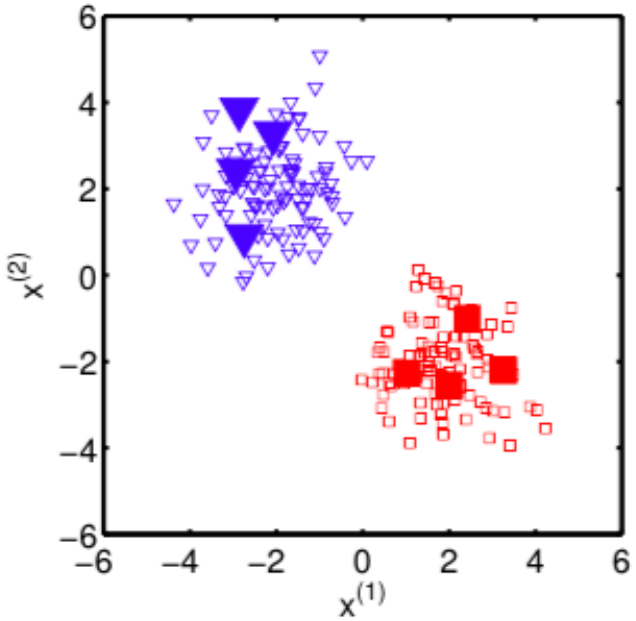
Following the approach from Shipeng Yu, Balaji Krishnapuram, Romer Rosales, Harald Steck, R. Bharat Rao. Bayesian Co-Training. Journal of Machine Learning Research 12:2649-2680, 2011. Images from that paper.

Bayesian co-training - when does it work?

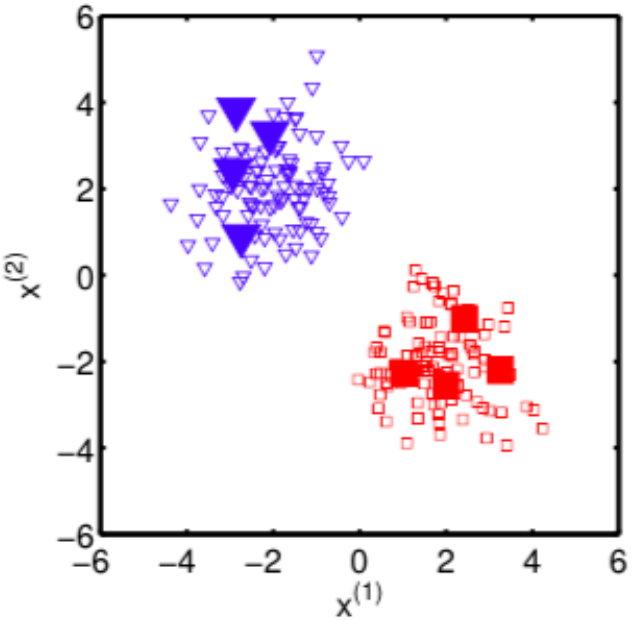
• Example result 1 on artificial data (horizontal + vertical direction are the two views):



(a) Toy data 1 (T1)



(b) Co-training on T1



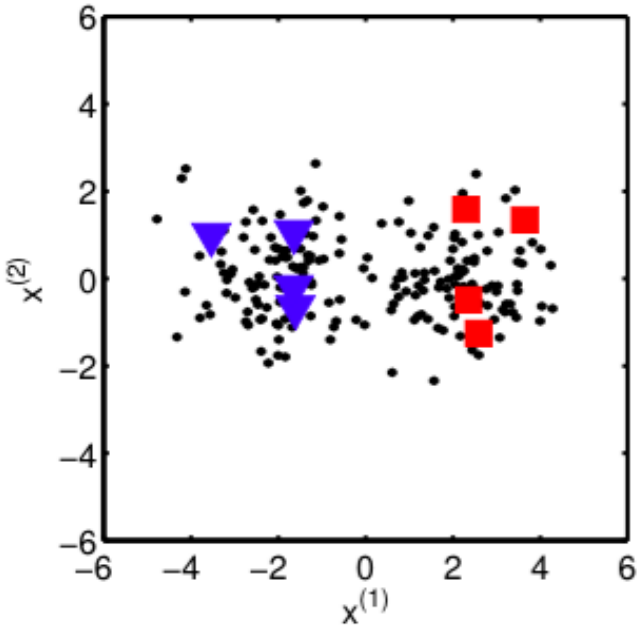
(c) Bayesian co-training on T1

On this data co-training assumptions are satisfied and both classical and Bayesian co-training succeed

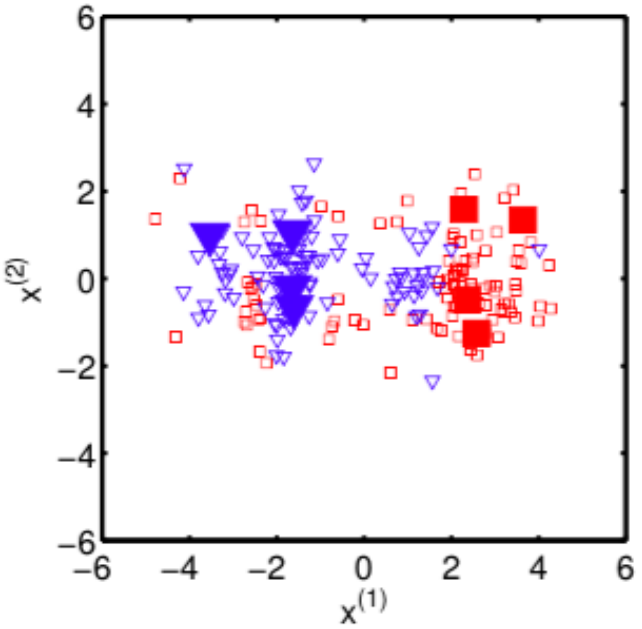
Following the approach from Shipeng Yu, Balaji Krishnapuram, Romer Rosales, Harald Steck, R. Bharat Rao. Bayesian Co-Training. Journal of Machine Learning Research 12:2649-2680, 2011. Images from that paper.

Bayesian co-training - when does it work?

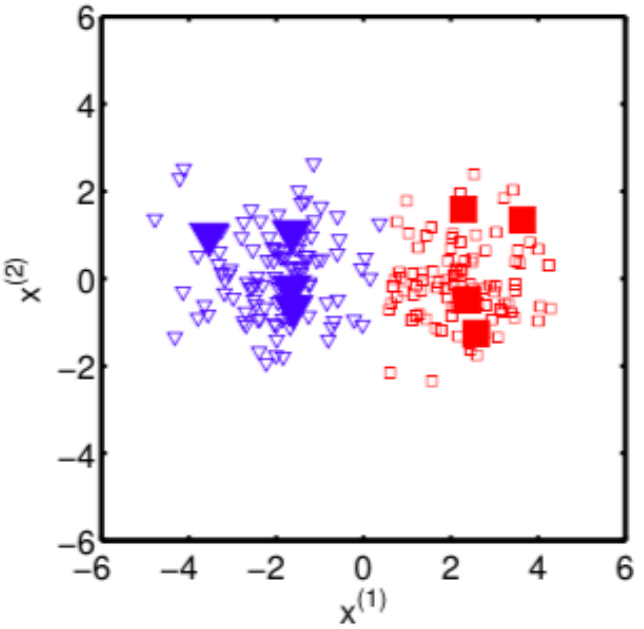
• Example result on artificial data 2 (horizontal + vertical direction are the two views):



(d) Toy data 2 (T2)



(e) Co-training on T2



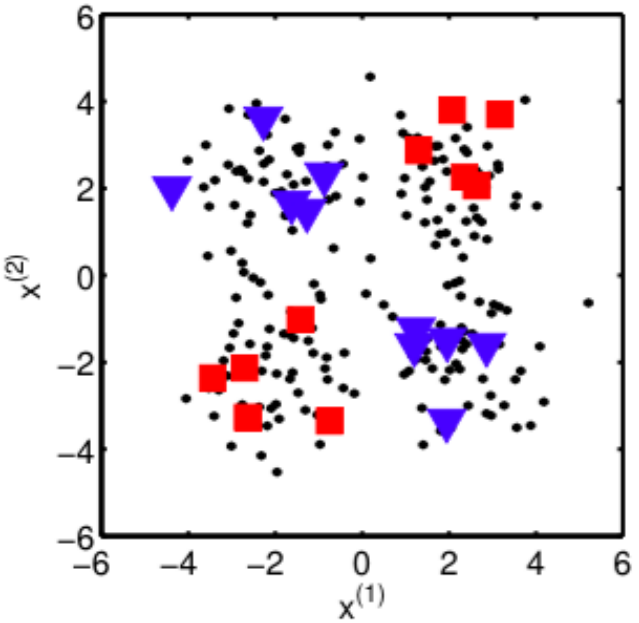
(f) Bayesian co-training on T2

On this data the vertical direction is not sufficient for good classification (contradicts classical co-training assumption; labels added to unlabeled points based on vertical direction will be noise), but Bayesian co-training still works since it can penalize the weight of the vertical direction.

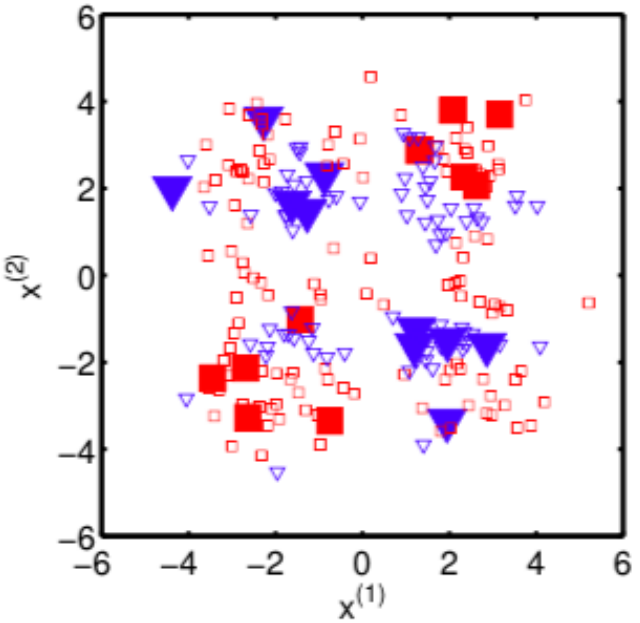
Following the approach from Shipeng Yu, Balaji Krishnapuram, Romer Rosales, Harald Steck, R. Bharat Rao. Bayesian Co-Training. Journal of Machine Learning Research 12:2649-2680, 2011. Images from that paper.

Bayesian co-training - when does it work?

• Example result on artificial data 3 (horizontal + vertical direction are the two views):



(g) Toy data 3 (T3)



(i) (Bayesian) co-training on T3

On this data neither view is sufficient by itself, and even Bayesian co-training fails. The consensus-based co-training kernel works poorly for this kind of data.

Following the approach from Shipeng Yu, Balaji Krishnapuram, Romer Rosales, Harald Steck, R. Bharat Rao. Bayesian Co-Training. Journal of Machine Learning Research 12:2649-2680, 2011. Images from that paper.

Bayesian co-training

• Example result: citeseer data (scientific papers in 6 classes; predict if paper belongs to largest class or not). Three views of the papers: (1) text of the paper itself, (2) text of citations (inbound links) to the paper, (3) text of citations from the paper to others (outbound links).

	# Train +2/-10		# Train +4/-20	
Model	AUC	F1	AUC	F1
Inbound Link	0.5451 ± 0.0025	0.3510 ± 0.0011	0.5479 ± 0.0035	0.3521 ± 0.0017
Outbound Link	0.5550 ± 0.0119	0.3552 ± 0.0053	0.5662 ± 0.0124	0.3600 ± 0.0059
Text+Link	0.5730 ± 0.0177	0.1386 ± 0.0561	0.5782 ± 0.0218	0.1474 ± 0.0721
Co-Trained GPLR	0.6459 ± 0.1034	0.4001 ± 0.2186	0.6519 ± 0.1091	0.4042 ± 0.2321
Bayesian Co-Training	$\mathbf{0.6536 \pm 0.0419}$	$\mathbf{0.4210 \pm 0.0401}$	$\mathbf{0.6880 \pm 0.0300}$	$\mathbf{0.4530 \pm 0.0293}$

• Performance measured by information retrieval criteria (AUC and F1 are different combinations of “precision” and “recall”, higher values are better)

Following the approach from Shipeng Yu, Balaji Krishnapuram, Romer Rosales, Harald Steck, R. Bharat Rao. Bayesian Co-Training. Journal of Machine Learning Research 12:2649-2680, 2011. Images from that paper.

Part 2: Multi-view learning with disagreeing views

Multi-view learning with view disagreement

- Multi-view approaches such as canonical correlation analysis assume the views **agree** (they describe the same data, and have some subspace that is well correlated)
- Real domains may have **view disagreement**: samples in each view do not belong to the same class e.g. due to corruption or other noise
- If disagreeing samples can be detected and left out, multi-view learning can be applied with the remaining samples

Following the approach from Christoudias, M., Urtasun, R., and Darrell, T. 2008. Multi-View Learning in the Presence of View Disagreement. 9 pp., In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI 2008). Images from that paper.

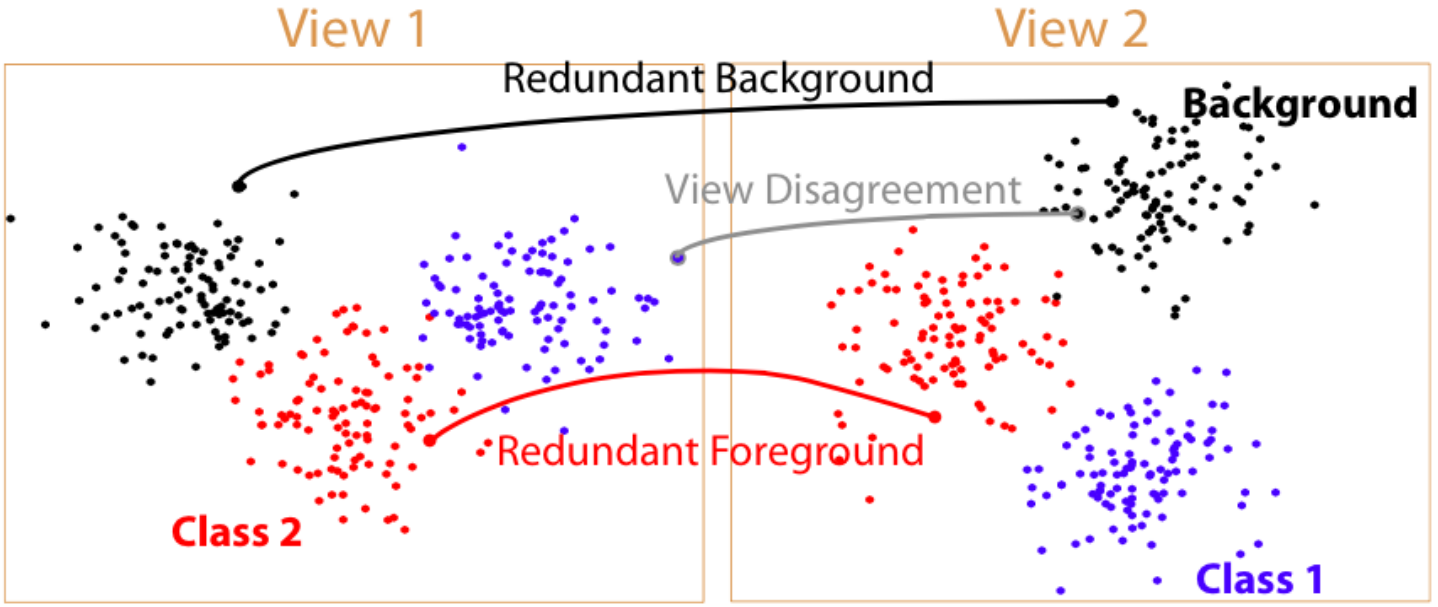
Multi-view learning with view disagreement

- Multi-view approaches such as canonical correlation analysis assume the views **agree** (they describe the same data, and have some subspace that is well correlated)
- Real domains may have **view disagreement**: samples in each view do not belong to the same class e.g. due to corruption or other noise
- If disagreeing samples can be detected and left out, multi-view learning can be applied with the remaining samples

Following the approach from Christoudias, M., Urtasun, R., and Darrell, T. 2008. Multi-View Learning in the Presence of View Disagreement. 9 pp., In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI 2008). Images from that paper.

Multi-view learning with view disagreement, example

- Two-view problem with normally distributed classes.
- 2 foreground classes (red+blue), 1 background class (black) of corrupted samples.
- Each point in view 1 corresponds to a point in view 2.



If views (data point classes) agree, the two views are redundant. Disagreement may occur because of incorrect pairing of views. Multi-view learning with these pairings leads to corrupted foreground class models.

Following the approach from Christoudias, M., Urtasun, R., and Darrell, T. 2008. Multi-View Learning in the Presence of View Disagreement. 9 pp., In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI 2008). Images from that paper.

Multi-view learning with view disagreement, example

• Multi-view learning methods assume views will agree, e.g. their cost functions may penalize disagreement between view-specific predictors with forms like

$$\min \sum_{\mathbf{x_k} \in U} \sum_{i \neq j} \| f_i(x_k^i) - f_j(x_k^j) \|_2^2$$

• If paired samples from each view in reality belong to **different classes**, views disagree about the sample.

• Idea: assume there is a background class that can co-occur with (be paired to) any foreground class; each foreground class is only paired to itself or the background class

• Example: audio vs video. People may say “yes” without nodding, or nod without saying “yes”.

• Idea: model the background class, detect view disagreement

Following the approach from Christoudias, M., Urtasun, R., and Darrell, T. 2008. Multi-View Learning in the Presence of View Disagreement. 9 pp., In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI 2008). Images from that paper.

Multi-view learning with view disagreement, example

- Use a **conditional entropy criterion** to detect disagreeing samples.
- Idea: given a fixed location in one view, how much uncertainty is there about the location in the other view?
- If there is much uncertainty, the views are likely to disagree about this sample (the fixed location is likely to be a “background sample”)
- Conditional entropy of location in view i given location in view j:

$$H(x^i|x^j)$$

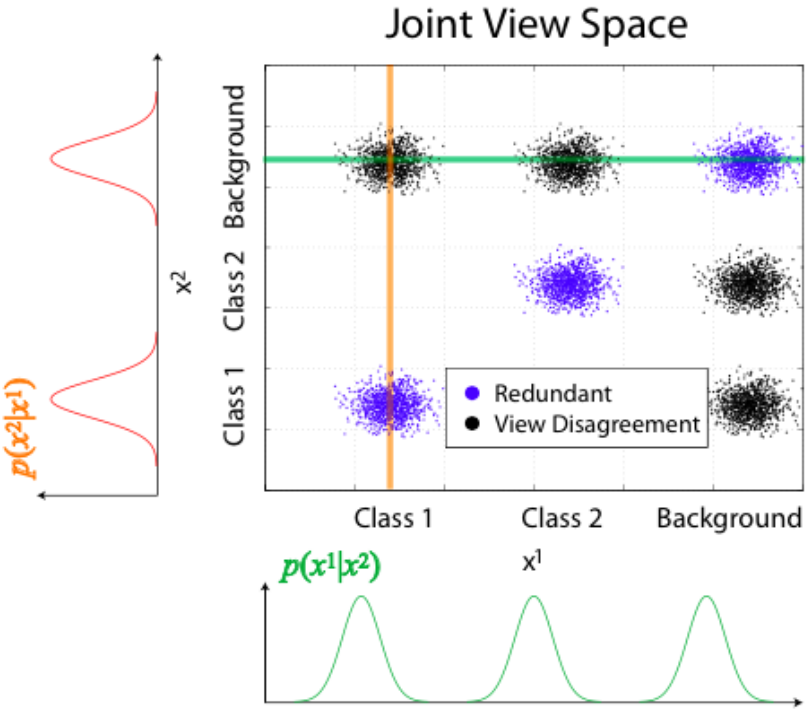


illustration of the conditional entropy idea

Following the approach from Christoudias, M., Urtasun, R., and Darrell, T. 2008. Multi-View Learning in the Presence of View Disagreement. 9 pp., In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI 2008). Images from that paper.

Multi-view learning with view disagreement, example

• Detect foreground samples: check if the conditional entropyof view i given view j for that sample is below the average value between those views:

$$m(x^i, x_k^j) = \begin{cases} 1, & H(x^i|x_k^j) < \bar{H}_{ij} \\ 0, & \text{otherwise} \end{cases}$$

where $H(x^i|x_k^j) = -\sum_{x^i \in U^i} p(x^i|x_k^j) \log p(x^i|x_k^j)$

$$\bar{H}_{ij} = \frac{1}{M} \sum_{\mathbf{x}_k \in U} H(x^i|x_k^j)$$

(U, Ui are the sets of possible values)

• A sample is a redundant foreground sample (good!) if all views confidently predict each other's locations:

• A sample is a redundant background sample (ok) if all views are uncertain about each other's locations

Following the approach from Christoudias, M., Urtasun, R., and Darrell, T. 2008. Multi-View Learning in the Presence of View Disagreement. 9 pp., In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI 2008). Images from that paper.

Multi-view learning with view disagreement, example

• A sample k is a redundant foreground sample (good!) if all views confidently predict each other's locations:

$$\prod_{i=1}^{V} \prod_{j \neq i} m(x^i, x_k^j) = 1$$

• A sample k is a redundant background sample (ok) if all views are uncertain about each other's locations

$$\sum_{i=1}^{V} \sum_{j \neq i} m(x^i, x_k^j) = 0$$

• Otherwise the views disagree about the sample. Two particular views disagree if the xor operator is 1 (one view is confident, the other is not): $m(x^i, x_k^j) \oplus m(x^j, x_k^i) = 1$

• In practice estimate probabilities by $p(x^i | x_k^j) = \frac{f(x^i, x_k^j)}{\sum_{x^i \in U^i} f(x^i, x_k^j)}$ where f are multivariate kernel density estimators

Following the approach from Christoudias, M., Urtasun, R., and Darrell, T. 2008. Multi-View Learning in the Presence of View Disagreement. 9 pp., In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI 2008). Images from that paper.

Multi-view learning with view disagreement, example

- Revised co-training algorithm with view disagreement:

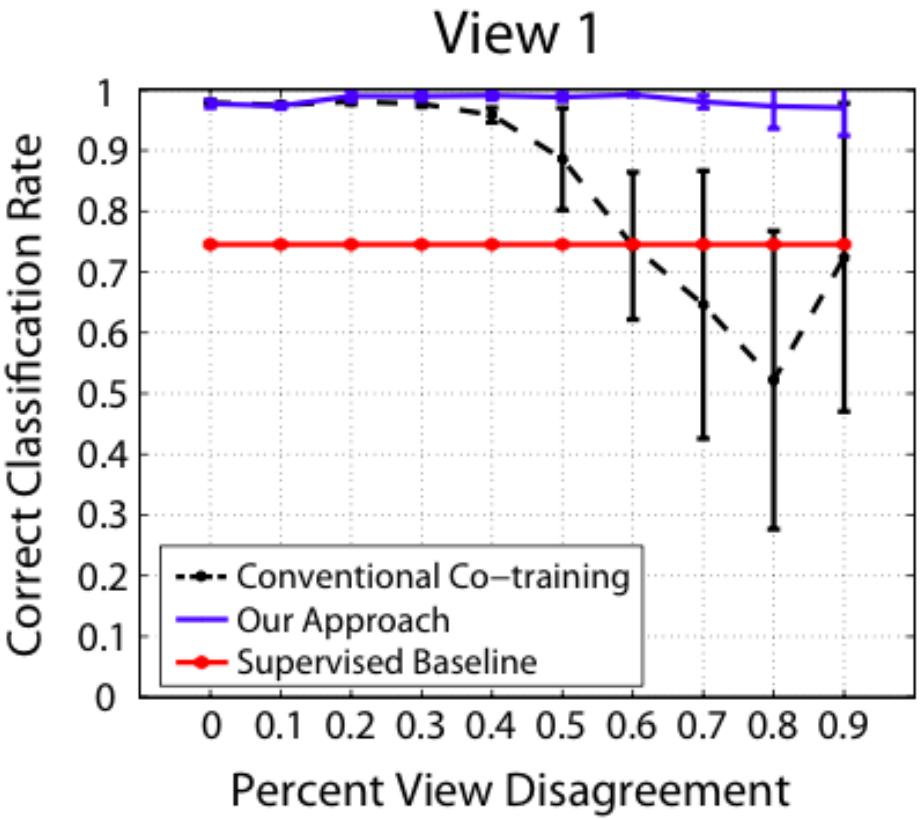
Algorithm 1 Multi-View Bootstrapping in the Presence of View Disagreement

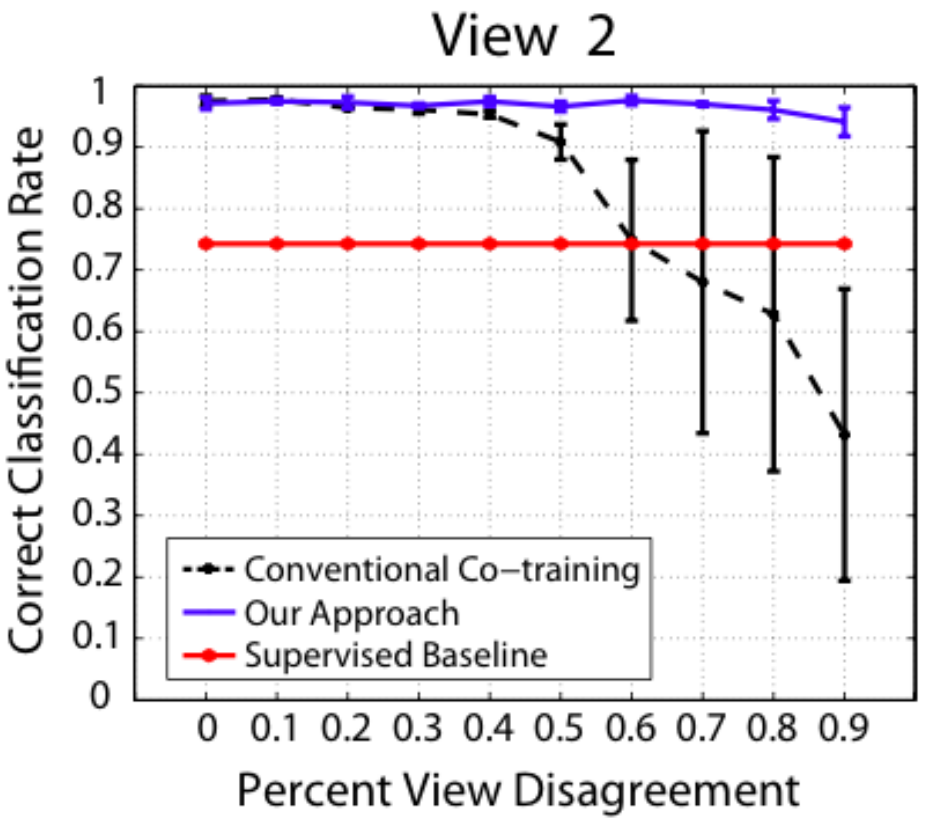
Given classifiers f_i and labeled seed sets S_i, $i = 1, ..., V$, unlabeled dataset U and parameters N and T:
Set $t = 1$.
repeat
  for $i = 1, ..., V$ **do**
    Train f_i on S_i
    Evaluate f_i on U^i
    Sort U in decreasing order by f_i confidence
    for each $\mathbf{x}_k \in U$, $k = 1, ..., N$ **do**
      for $j \neq i$ **do**
        if $\neg(m(x^i, x^j_k) \oplus m(x^j, x^i_k))$ **then**
          $U^j = U^j \backslash \{x^j_k\}$
          $S^j = S^j \cup \{x^j_k\}$
        end if
      end for
      $U^i = U^i \backslash \{x^i_k\}$
      $S^i = S^i \cup \{x^i_k\}$
    end for
  end for
  Set $t = t + 1$.
until $|U| = \emptyset$ or $t = T$

Following the approach from Christoudias, M., Urtasun, R., and Darrell, T. 2008. Multi-View Learning in the Presence of View Disagreement. 9 pp., In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI 2008). Images from that paper.

Multi-view learning with view disagreement, example

- Toy example with varying amount of view disagreement:





Following the approach from Christoudias, M., Urtasun, R., and Darrell, T. 2008. Multi-View Learning in the Presence of View Disagreement. 9 pp., In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI 2008). Images from that paper.

References for part 2

- Christoudias, M., Urtasun, R., and Darrell, T. 2008. **Multi-View Learning in the Presence of View Disagreement.** 9 pp., In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI 2008).

- K. Ganchev, J. V. Graca, J. Blitzer, and B. Taskar. **Multi-View Learning over Structured and Non-Identical Outputs.** In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI 2008).

- Bach, F., Lanckriet, G. and Jordan, M.I. **Multiple kernel learning, conic duality, and the SMO algorithm.** 21st International conference on machine learning, ACM, 2004.

- Sonnenburg, S., Rätsch, G. Schäfer, C. and Schölkopf, B. **Large scale multiple kernel learning.** The Journal of Machine Learning Research. 7:1531-1565, 2006.

- Farquhar, J., Hardoon, D.R., Meng, H., Shawe-Taylor, J. and Szedmak, S. **Two-view learning: SVM-2K, theory and practice.** NIPS 18, pp. 355-362, 2006.

Part 3: Semi-supervised multi-task learning

Semisupervised multi-task learning, introduction

- The overall idea is:

1. define a semisupervised single-task classifier

   the classifier will be defined based on a graph of data sample similarities, and unlabeled samples will affect the learning through the graph

2. define several such classifiers, and make the learning of their parameters depend on each other through a joint prior

   the prior could be e.g. a Dirichlet process type prior, allowing there to be as many clusters of similar tasks as needed

Following the approach from Q. Liu, X. Liao, H. Li, J. R. Stack, and L. Carin. Semisupervised multitask learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31(6):1074-1086, 2009. Images from that paper.

Semisupervised multi-task learning, introduction

- 1. define a semisupervised single-task classifier
  the classifier will be defined based on a graph of data sample similarities, and unlabeled samples will affect the learning through the graph

• Example: consider a kernel-based classifier, such as a Gaussian process where the covariance function is a kernel.

• Consider a kernel based on distances: $k(x,x')=\exp(-a*d^2(x,x'))$

• Now define a graph that connects K-nearest neighbors of all points in the data set (labeled and unlabeled), and define distances as distances along the graph.

• For faraway points this distance takes into account the “shape” of the data, which is learned mostly from the unlabeled points.

• Learning the classifier can be done as normal, using this new kernel

Following the approach from Q. Liu, X. Liao, H. Li, J. R. Stack, and L. Carin. Semisupervised multitask learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31(6):1074-1086, 2009. Images from that paper.

Semisupervised multi-task learning, introduction

- The overall idea is:

1. define a semisupervised single-task classifier - DONE

2. define several such classifiers, and make the learning of their parameters depend on each other through a joint prior

   In the case of a Gaussian process classifier, the prior can be e.g. over the parameters of the kernel, such as an overall scale “a”; or the prior could be over parameters of the graph such as how many neighbors are connected, and so on.

   See lecture “Multitask learning with task clustering or gating “ for a wide variety of priors that can be applied over the parameters.

Following the approach from Q. Liu, X. Liao, H. Li, J. R. Stack, and L. Carin. Semisupervised multitask learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31(6):1074-1086, 2009. Images from that paper.

Part 4: “Self-taught learning”

Self-taught learning

- The previous multitask unsupervised learning approach still essentially assumed that the unlabeled data within each task came from the same distribution as the labeled data
- Can we do multi-task learning (or transfer learning) without this assumption?
- For example, suppose plan to do classification of image, e.g. classifying animal images - is the animal in the image an elephant or a rhino.
- It would be difficult to gather unlabeled images of elephants&rhinos - if you know it is an elephant or rhino, that already means you are likely to know the label!
- Can we improve the classification using some random unlabeled images downloaded from the internet?

Following the approach from R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *proceedings of ICML 2007, the 24th International Conference on Machine Learning*, pages 759-766, ACM, 2007. Images from that paper.

Self-taught learning

- Such data comes mostly from outside the problem domain - the probably most random images from the internet won't be about elephants or rhinos
- But they may still contain some of the same feature properties that are useful for elephants and rhinos too!
- That is: the problem we are solving (and its data) may come from a **larger problem domain** where some of the same features are likely to be useful across many problems (and their data)
- For example images of rhinos & elephants are part of the larger class of " images of animals", which are part of the larger class of "natural images", which are part of the larger class of "images".
- Thus, features useful for solving classification problems may be shared between "images or rhinos & elephants" and other "natural images"

Following the approach from R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *proceedings of ICML 2007, the 24th International Conference on Machine Learning*, pages 759-766, ACM, 2007. Images from that paper.

Self-taught learning

- **Idea:** given a task T of interest unlabeled data from a larger problem domain can be used to learn which data features are noise, and which features contain trends and statistical structure (such features can be useful in classification problems!)
- For example, in images, any natural images will contain some of the same basic features like corners, edges similar to those in elephants and rhinos.
- This is different from normal semisupervised learning: This kind of unlabeled data does not share the same class labels or the generative distribution of the labeled data as in task T
- Thus the unlabeled data generally cannot be reasonably assigned to the class labels of task T (it is not reasonable to try to classify whether a picture of a tree is more like an elephant or a rhino)

Following the approach from R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *proceedings of ICML 2007, the 24th International Conference on Machine Learning*, pages 759-766, ACM, 2007. Images from that paper.

Self-taught learning

- Supervised classification uses labeled examples of each class of interest

Supervised Classification

- Semi-supervised learning uses also unlabeled examples of those classes

Semi-supervised Learning

- Transfer learning uses labeled datasets of different classes

Transfer Learning

- Self-taught learning just needs additional unlabeled images

Self-taught Learning

Following the approach from R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *proceedings of ICML 2007, the 24th International Conference on Machine Learning*, pages 759-766, ACM, 2007. Images from that paper.

Self-taught learning

- How to learn from lots of unlabeled images?
- For example, apply some fully unsupervised statistical model to learn which features are meaningful over the collection of all images
  - Like principal component analysis
  - Or some sparse feature extraction method like this:

$$\text{minimize}_{b,a} \quad \sum_i \|x_u^{(i)} - \sum_j a_j^{(i)} b_j\|_2^2 + \beta \|a^{(i)}\|_1$$
$$\text{s.t.} \qquad \|b_j\|_2 \leq 1, \quad \forall j \in 1, ..., s$$

(tries to reconstruct data coordinates as linear combinations, with weights $a_j^{(i)}$, of a small number of basis vectors b_j)

- The learn the individual tasks using those features, and the data (labeled and unlabeled images) from that task ---> reduces to the earlier methods

Following the approach from R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *proceedings of ICML 2007, the 24th International Conference on Machine Learning*, pages 759-766, ACM, 2007. Images from that paper.

References for parts 3 and 4

- Q. Liu, X. Liao, H. Li, J. R. Stack, and L. Carin. **Semisupervised multitask learning.** *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31(6):1074-1086, 2009.

- R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. **Self-taught learning: transfer learning from unlabeled data.** In *proceedings of ICML 2007, the 24th International Conference on Machine Learning*, pages 759-766, ACM, 2007.

- P. S. Dhillon, S. Sellamanickam, S. K. Selvaraj. **Semi-supervised multi-task learning of structured prediction models for web information extraction**, *CIKM 2011*