Covariate shift

IN4320 Machine Learning
29/03/2017
Wouter Kouw

Outline

- **Covariate shift**
  - Real-world example
  - Class-posterior distributions
- **Importance-weighing**
  - Probability ratio
  - Weighted classifier
  - Effective sample size
- **Nonparametric estimator**
  - Maximum Mean Discrepancy
  - Kernel Mean Matching
  - Constraints
- **Recap**

Problem

- **Suppose that a Dutch hospital hires you to build a diabetes diagnosis system.**
  - Clinical research indicates that the level of Hemoglobin A1c in a person's blood is strongly related to the disease development.
  - They have data on patients and controls from the last 10 years.
- **You use the data to train a classifier that diagnoses diabetes based on a small blood measurement.**
  - The system performs well and is widely adopted.

Problem

- **Now, a hospital in the United States hears of your success and wants to deploy your system.**
  - Suddenly, it performs a lot worse.
- **What has happened?**

Covariate shift

- **It seems that there are differences between people from the Netherlands and people from the U.S.**
  - Specifically, the normal/abnormal levels of Hemoglobin A1c are different.
- **What do you do now?**
  - You consider gathering new labeled data from patients in the U.S.
  - But that might take years, due to different regulations.
- **Maybe it would also be possible for your system to adapt to the new situation, given some unlabeled data...**

Covariate shift

- **If you have a classification problem where the training data and the test data are different, then you are in a *transfer learning* setting.**

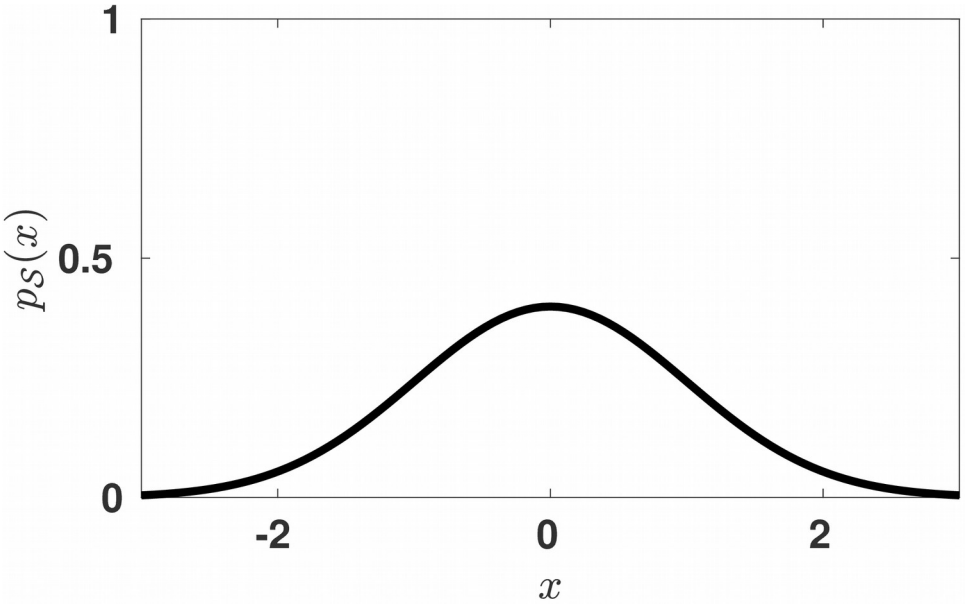
- **If you have measured/extracted the same features for both the training and test data, then you are in the specific case of *domain adaptation*.**

- **If you have reason to believe that the class-posterior distributions are the same for both the training and test data, then you are in a *covariate shift* setting.**

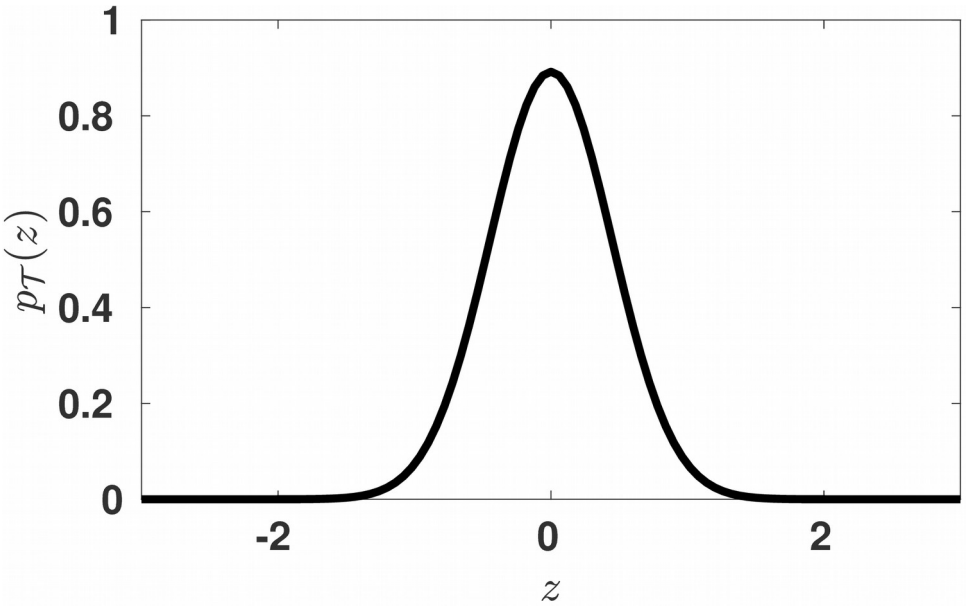
Covariate shift

– **Data distributions p(x)**

Source domain



Target domain

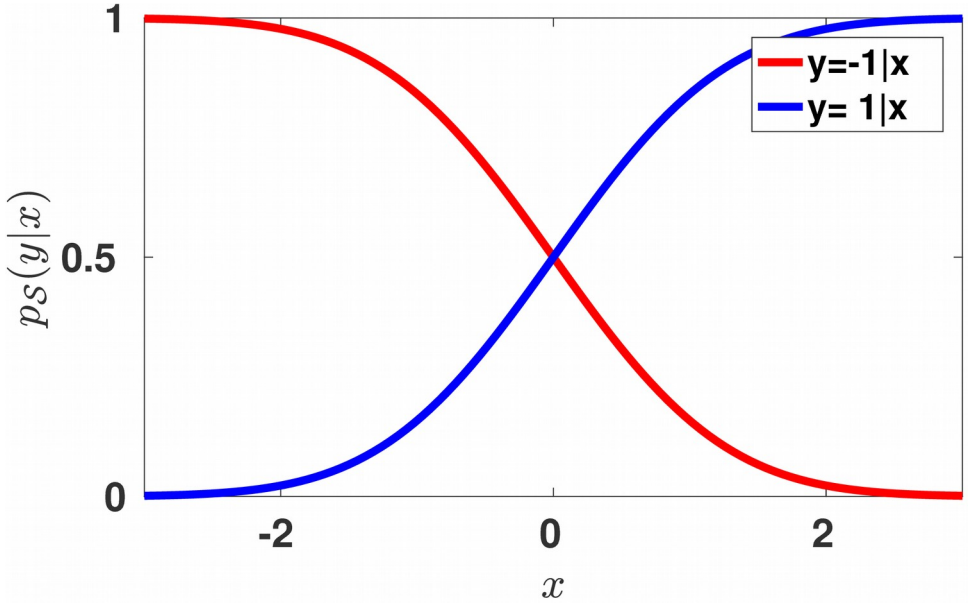


Moreno-Torres et al., 2012

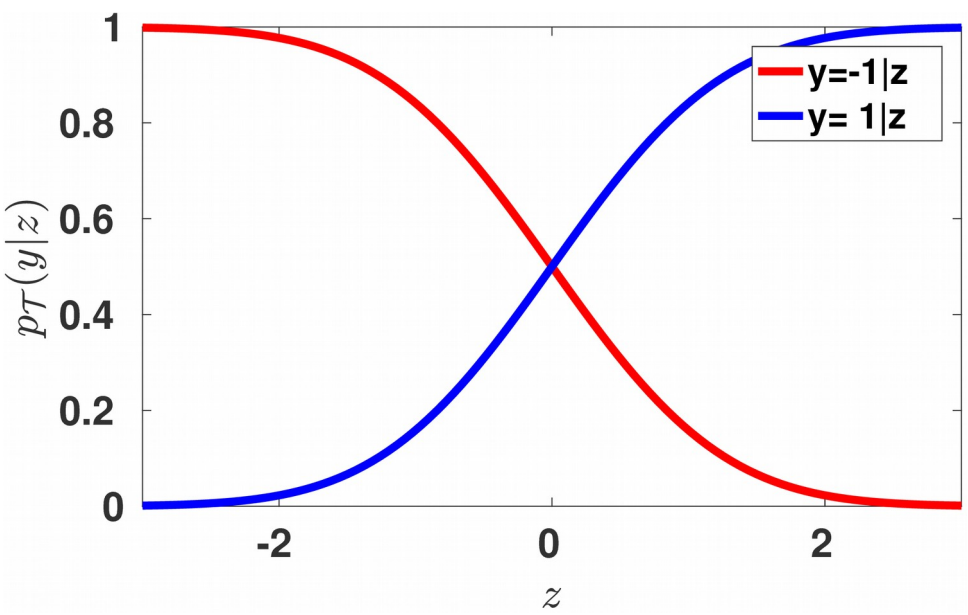
Covariate shift

– **Class-posterior distributions p(y|x)**

Source domain



Target domain



Covariate shift

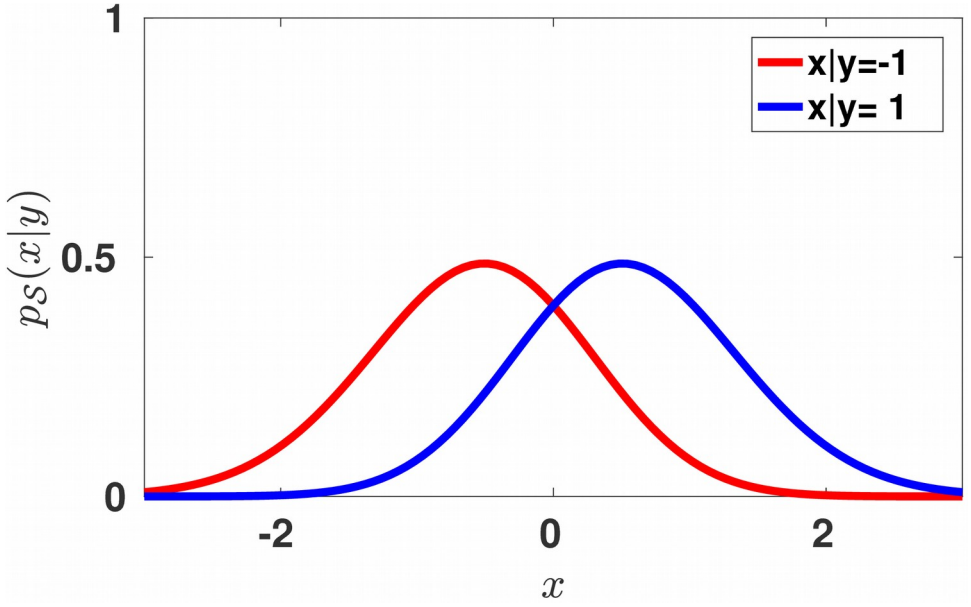
– **Remember Bayes' rule:**

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

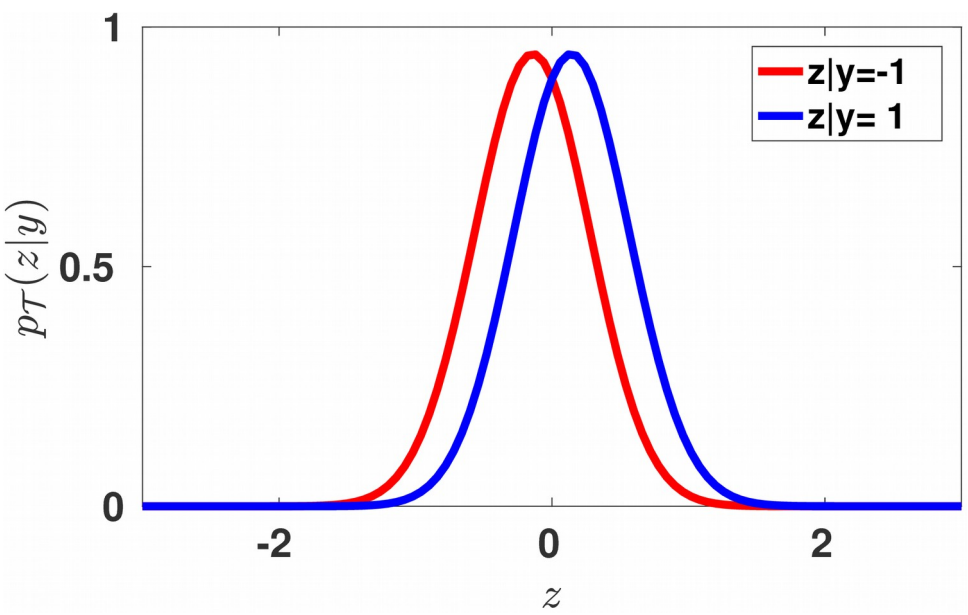
Covariate shift

– **Class-conditional likelihoods p(x|y)**

Source domain

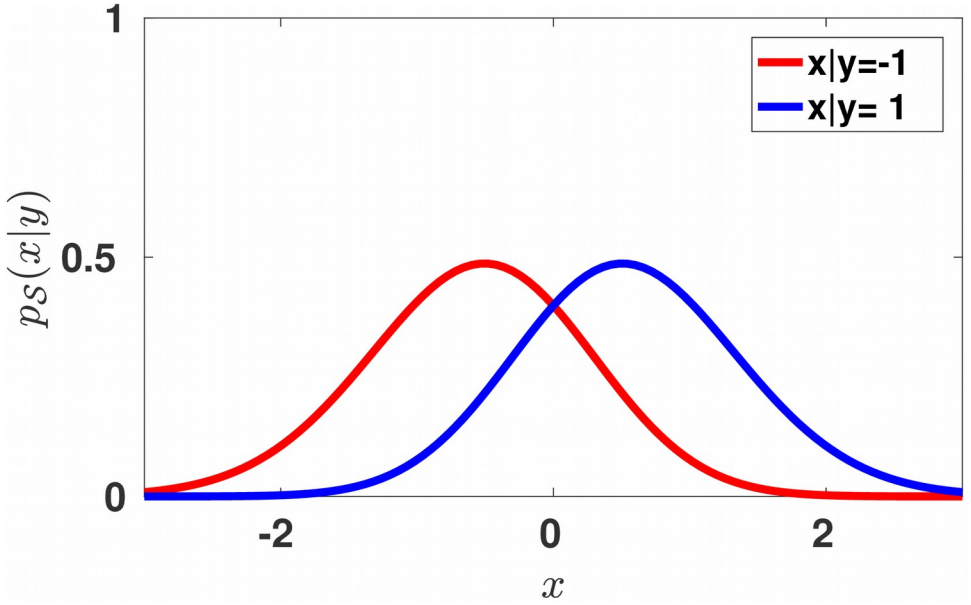


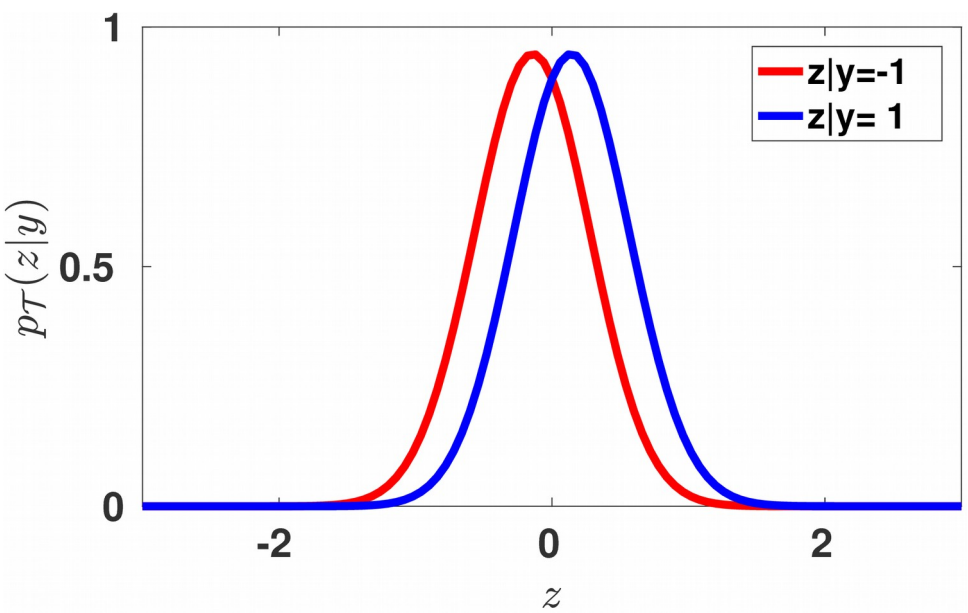
Target domain



Covariate shift

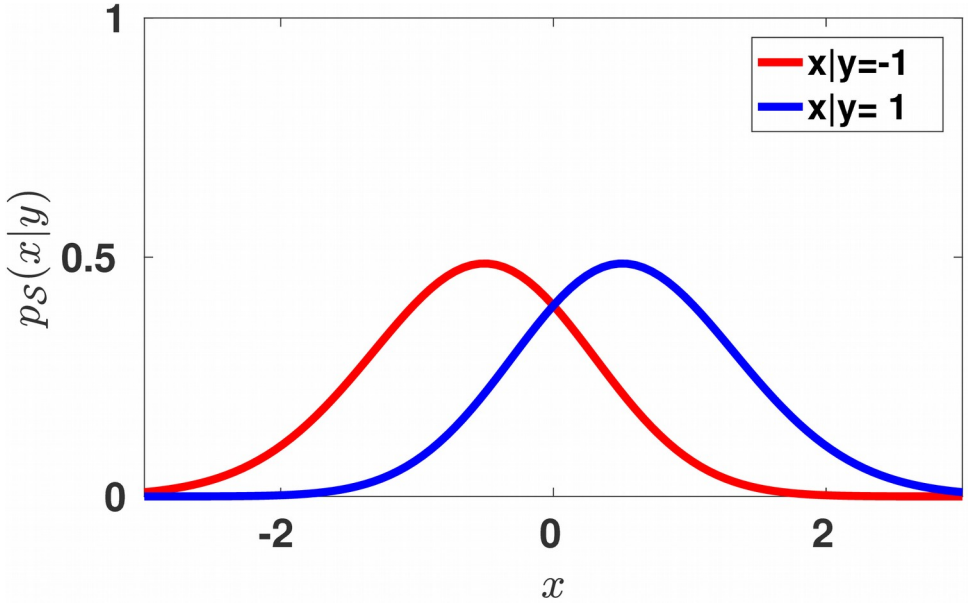
- Red curve: Hemoglobin levels of healthy people.
- Blue curve: Hemoglobin levels of diabetics.
- Source (left): data from the Netherlands.
- Target (right): data from the United States.

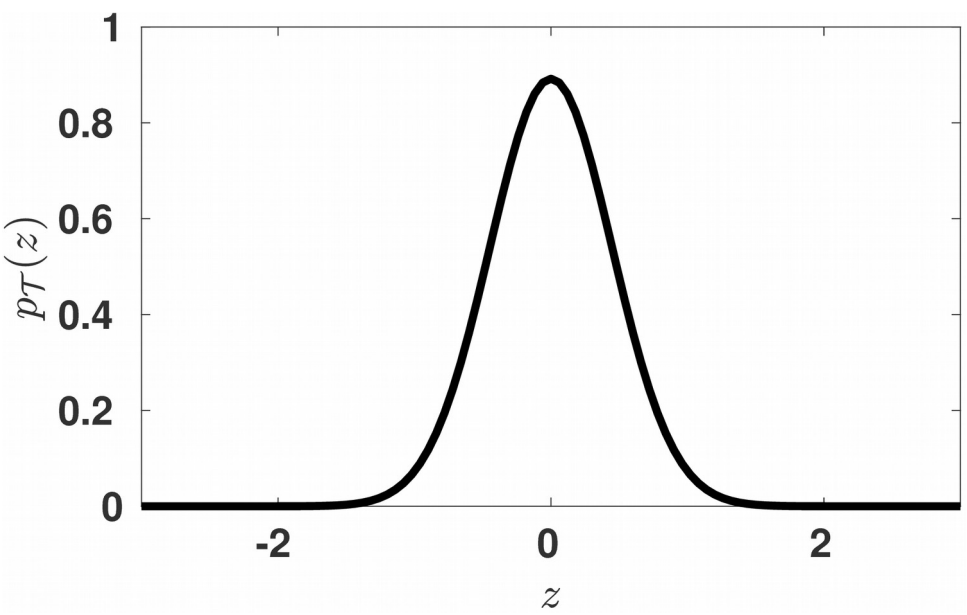




Covariate shift

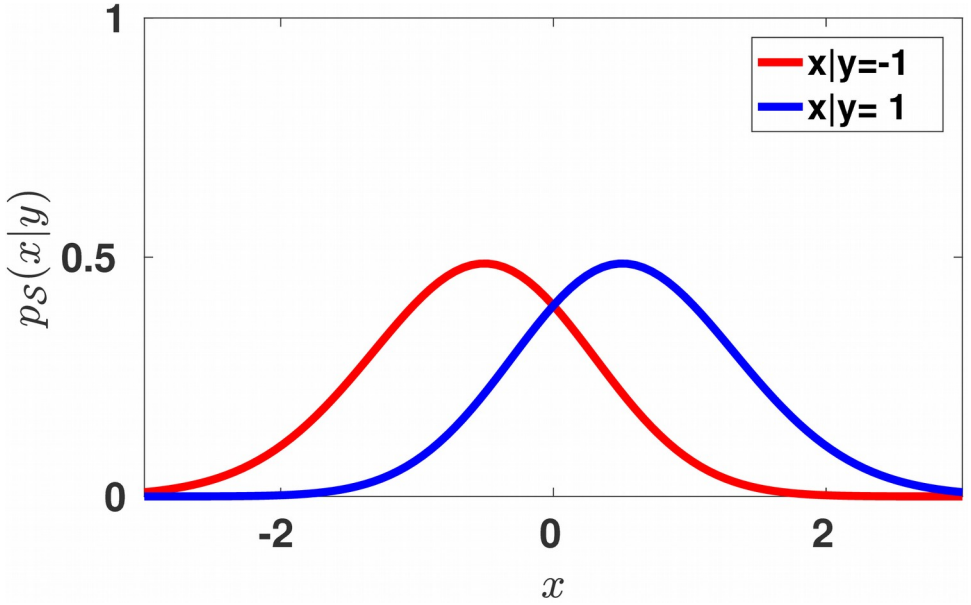
– **But we don’t know the target labels.**

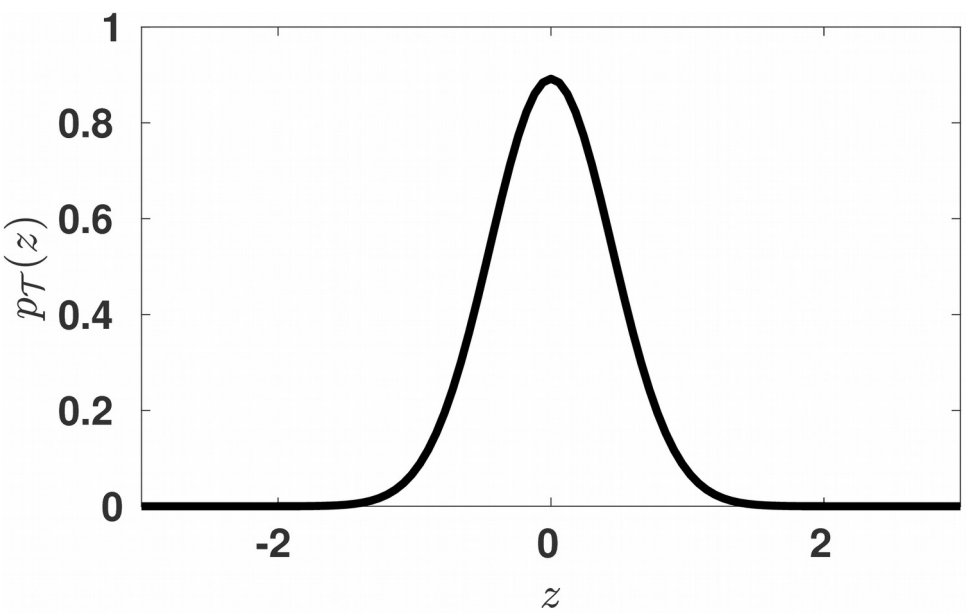




Covariate shift

- **But we don’t know the target labels.**
  - Can we train a classifier on the labeled source and the unlabeled target data that will generalize well to the target domain?





Risk

- **If we were to look at the expected loss, i.e. *risk*:**
  - Source:

$$R_{\mathcal{S}} = \int_{\Omega} \sum_{y} \ell(h(x|\theta), y) p_{\mathcal{S}}(x, y) \, \mathrm{d}x$$

  - Target :

$$R_{\mathcal{T}} = \int_{\Omega} \sum_{y} \ell(h(x|\theta), y) p_{\mathcal{T}}(x, y) \, \mathrm{d}x$$

Risk

– **Then it becomes easy to see that we could correct for the difference in probabilities:**

$$R_{\mathcal{T}} = \int_{\Omega} \sum_{y} \ell(h(x|\theta), y) p_{\mathcal{T}}(x, y) \, \mathrm{d}x$$
$$= \int_{\Omega} \sum_{y} \ell(h(x|\theta), y) \frac{p_{\mathcal{T}}(x, y)}{p_{\mathcal{S}}(x, y)} p_{\mathcal{S}}(x, y) \, \mathrm{d}x$$

– **But we still need to know the target joint distribution to evaluate this risk.**

Shimodaira, 2000

Risk

- **Using the assumption on the class-posterior distributions:** $p_{\mathcal{T}}(y|x) = p_{\mathcal{S}}(y|x)$

$$\frac{p_{\mathcal{T}}(x,y)}{p_{\mathcal{S}}(x,y)} = \frac{p_{\mathcal{T}}(y|x)p_{\mathcal{T}}(x)}{p_{\mathcal{S}}(y|x)p_{\mathcal{S}}(x)} = \frac{p_{\mathcal{T}}(x)}{p_{\mathcal{S}}(x)}$$

- **In the covariate shift setting, the source-risk can be weighed to obtain the target risk:**

$$R_{\mathcal{W}} = \int_{\Omega} \sum_{y} \ell(h(x|\theta), y) \frac{p_{\mathcal{T}}(x)}{p_{\mathcal{S}}(x)} p_{\mathcal{T}}(x,y) \, \mathrm{d}x$$

Weighted classifier

TU Delft

- **So, how to go from a risk function to a classifier?**

- **First, plug in samples:**

$$\hat{R}_{\mathcal{W}} = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i|\theta), y_i) \frac{p_{\mathcal{T}}(x_i)}{p_{\mathcal{S}}(x_i)}$$

- **Then, plug in a loss function (here quadratic):**

$$\hat{R}_{\mathcal{W}} = \frac{1}{n} \sum_{i=1}^{n} \left(h(x_i|\theta) - y_i\right)^2 \frac{p_{\mathcal{T}}(x_i)}{p_{\mathcal{S}}(x_i)}$$

- **Finally, plug in a hypothesis space (here linear):**

$$\hat{R}_{\mathcal{W}} = \frac{1}{n} \sum_{i=1}^{n} \left(x_i\theta - y_i\right)^2 \frac{p_{\mathcal{T}}(x_i)}{p_{\mathcal{S}}(x_i)}$$

Weighted classifier

– **If I denote the ratio of probabilities as** w(x_i)**, then the solution of the average loss w.r.t. the classifier parameters is:**

$$\frac{\partial}{\partial \theta^*} \frac{1}{n} \sum_{i=1}^{n} \left(x_i \theta - y_i\right)^2 w(x_i) \equiv 0$$

$$\frac{1}{n} \sum_{i=1}^{n} 2(x_i \theta^* - y_i)(x_i) w(x_i) = 0$$

$$\frac{2}{n} \sum_{i=1}^{n} x_i^2 w(x_i) \theta^* = \frac{2}{n} \sum_{i=1}^{n} y_i x_i w(x_i)$$

$$\theta^* = \frac{\sum_{i=1}^{n} y_i x_i w(x_i)}{\sum_{i=1}^{n} x_i^2 w(x_i)}$$

Weighted classifier

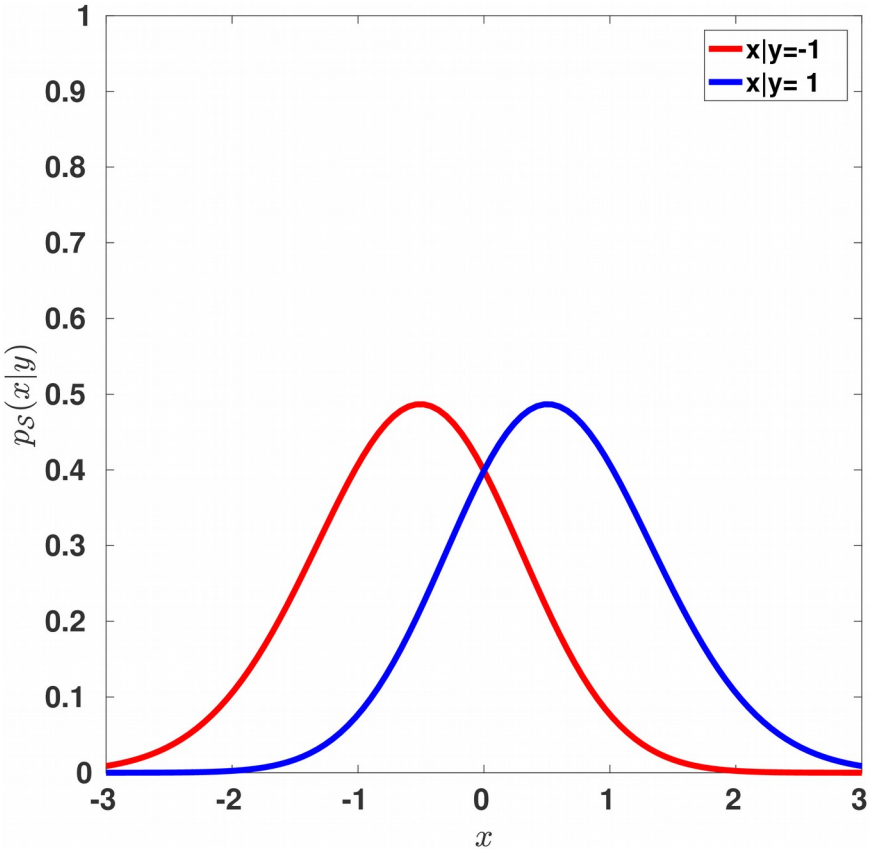
– **Or in matrix form:**

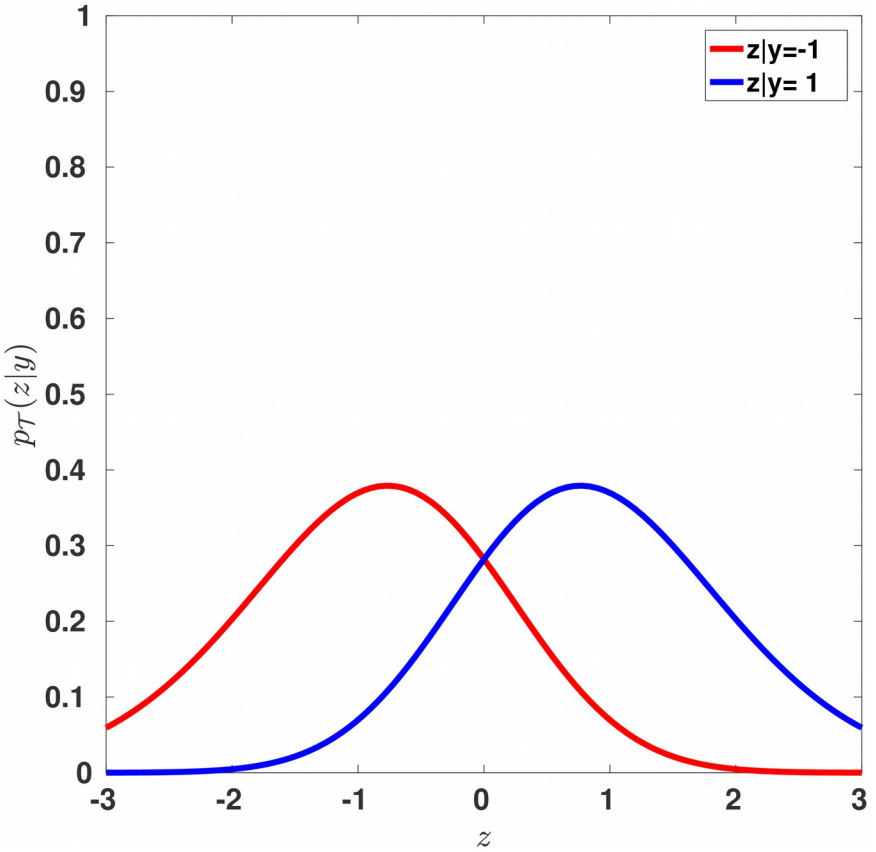
$$\theta^* = (XWX)^{-1}XWy$$

where $W = \begin{bmatrix} w(x_1) & \dots & 0 \\ & \ddots & \\ 0 & \dots & w(x_n) \end{bmatrix}$

Weighted classifier

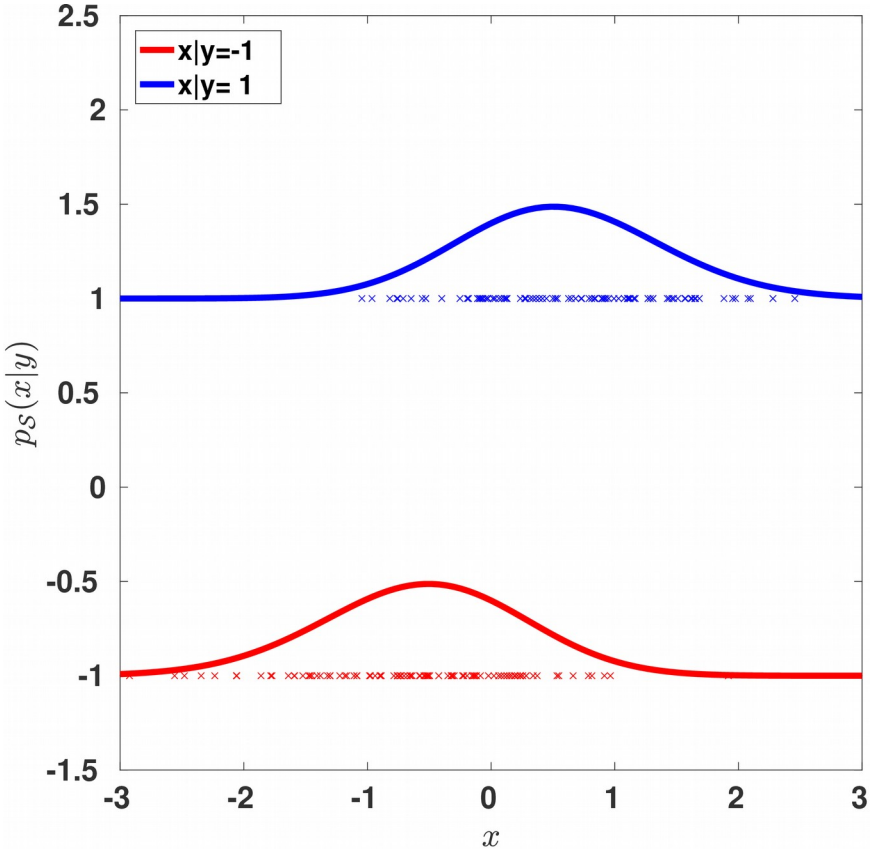
– **Example**

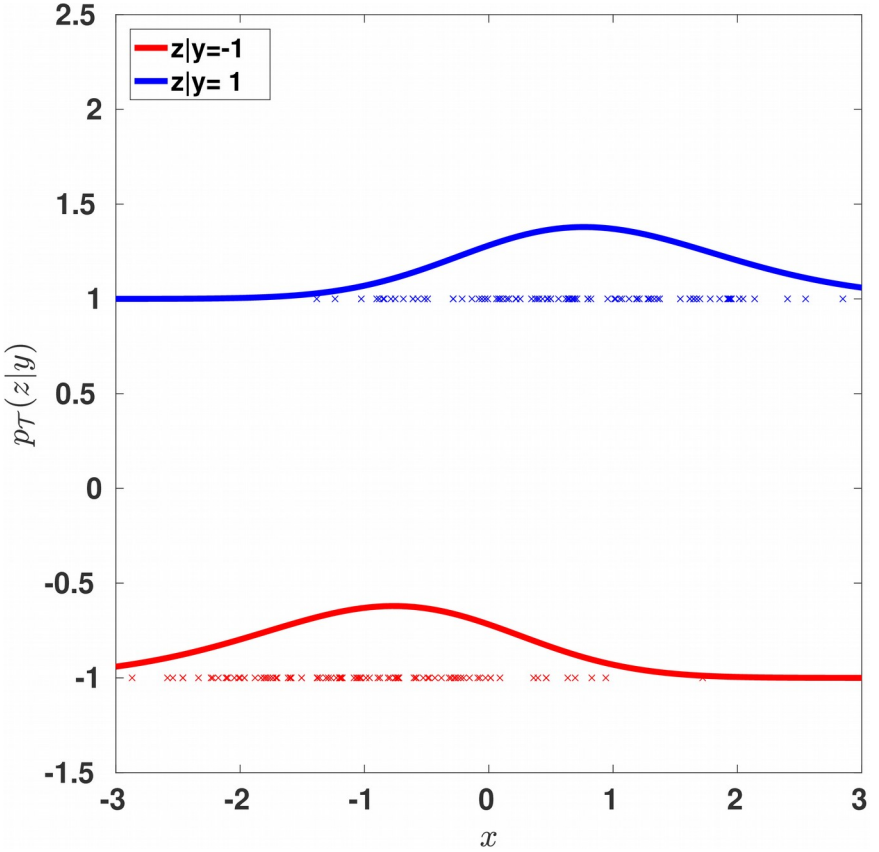




Weighted classifier

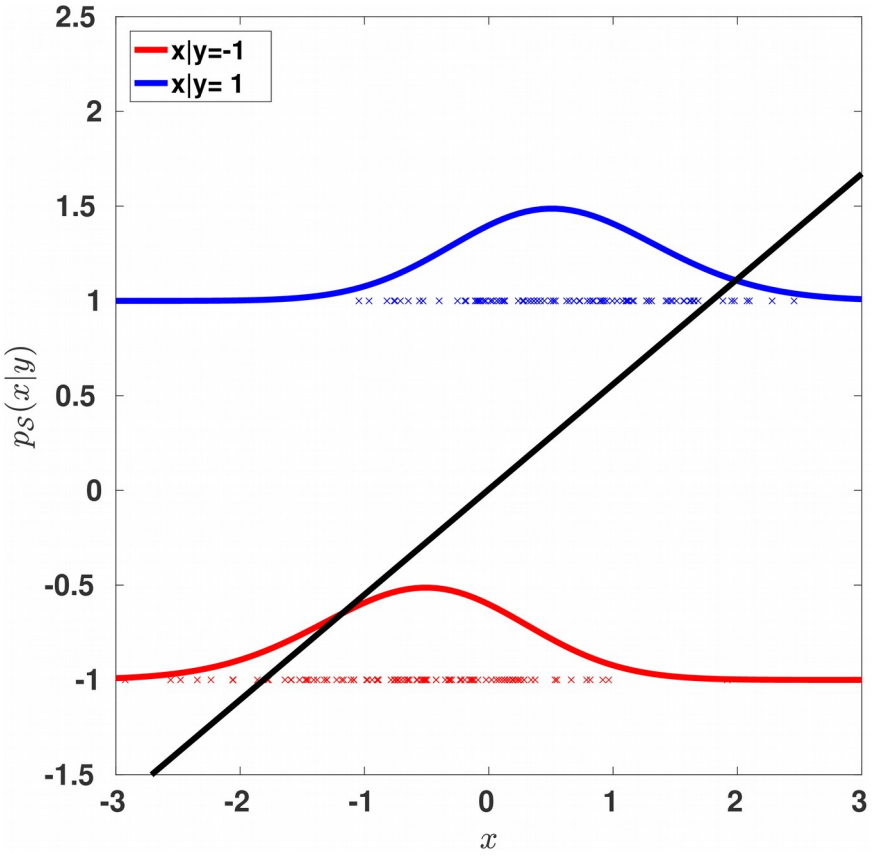
– **Example**

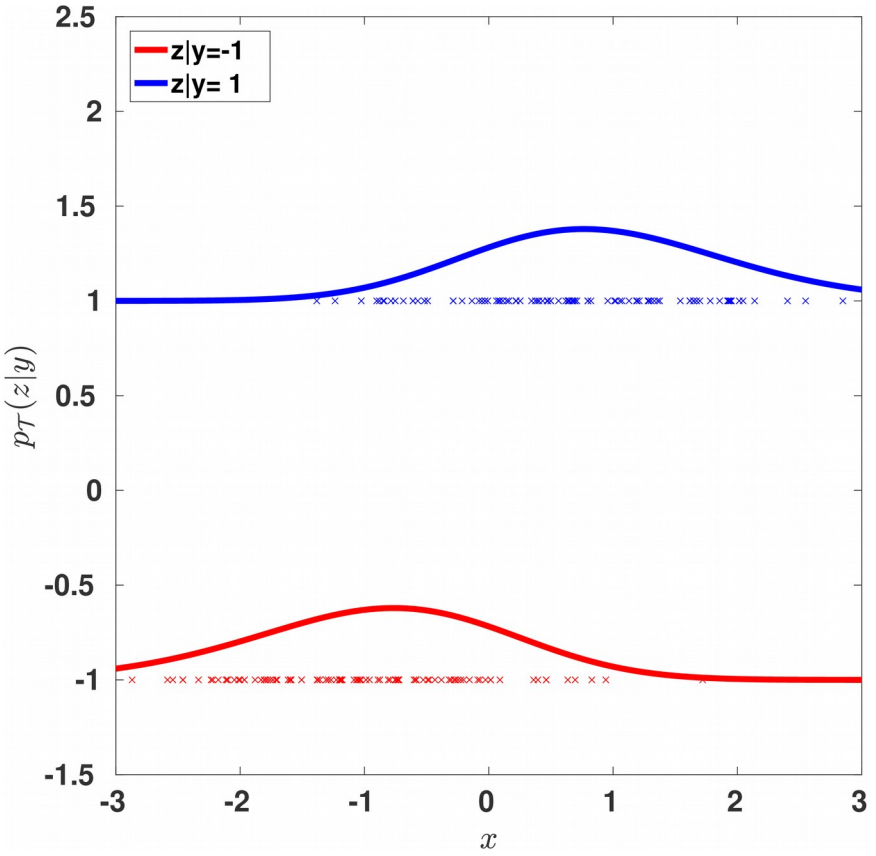




Weighted classifier

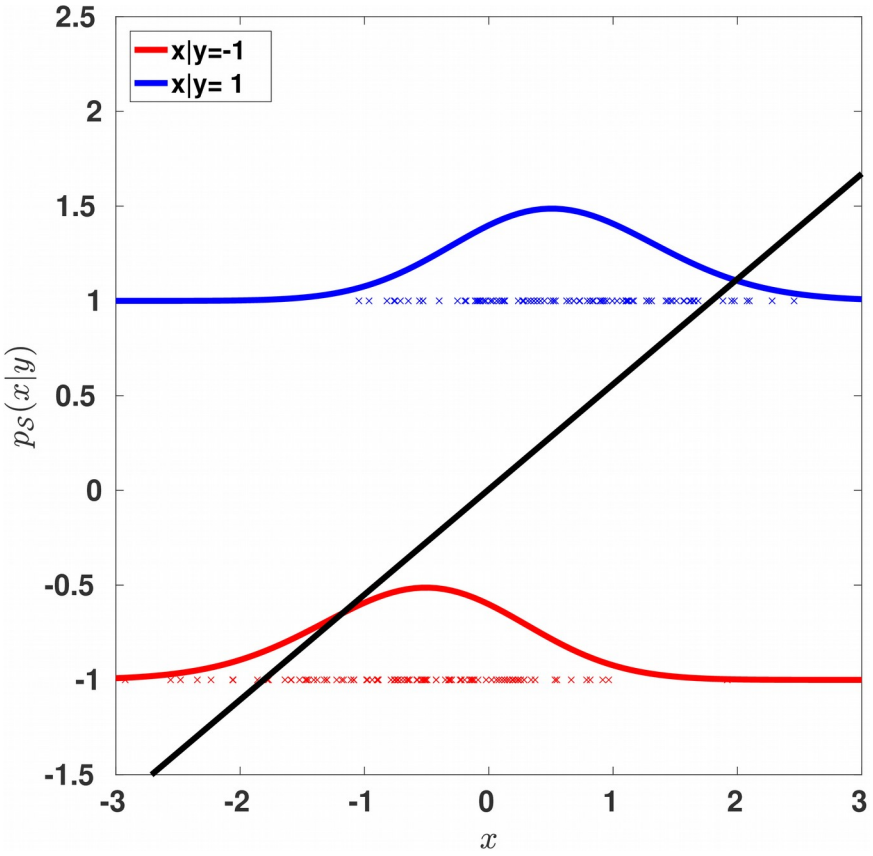
– **Example**

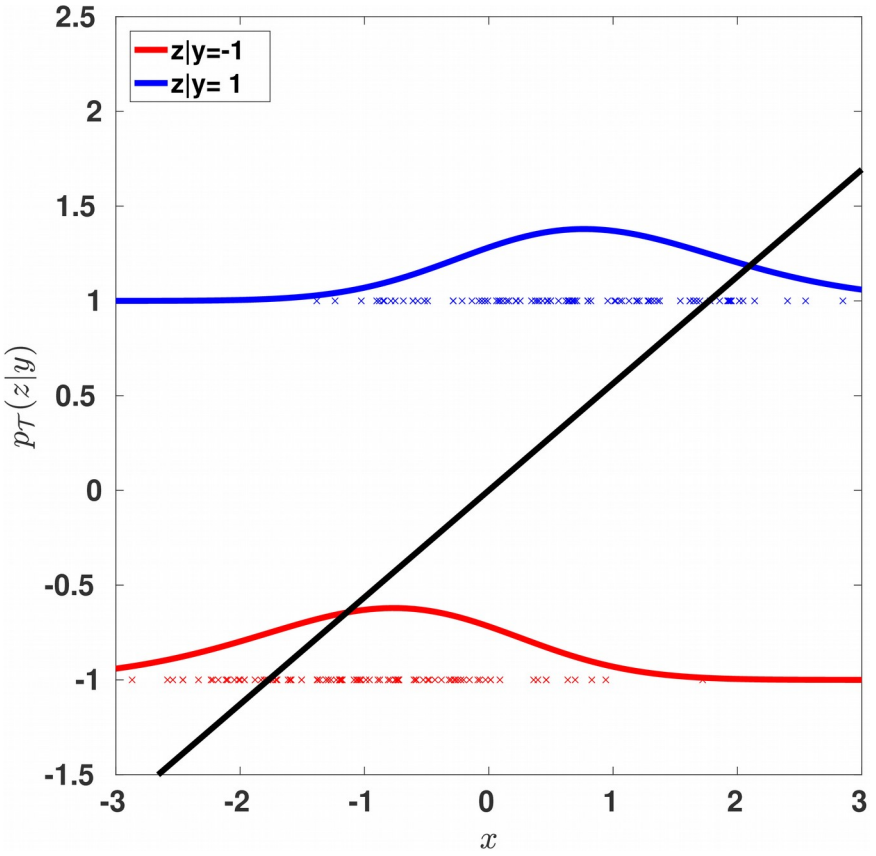




Weighted classifier

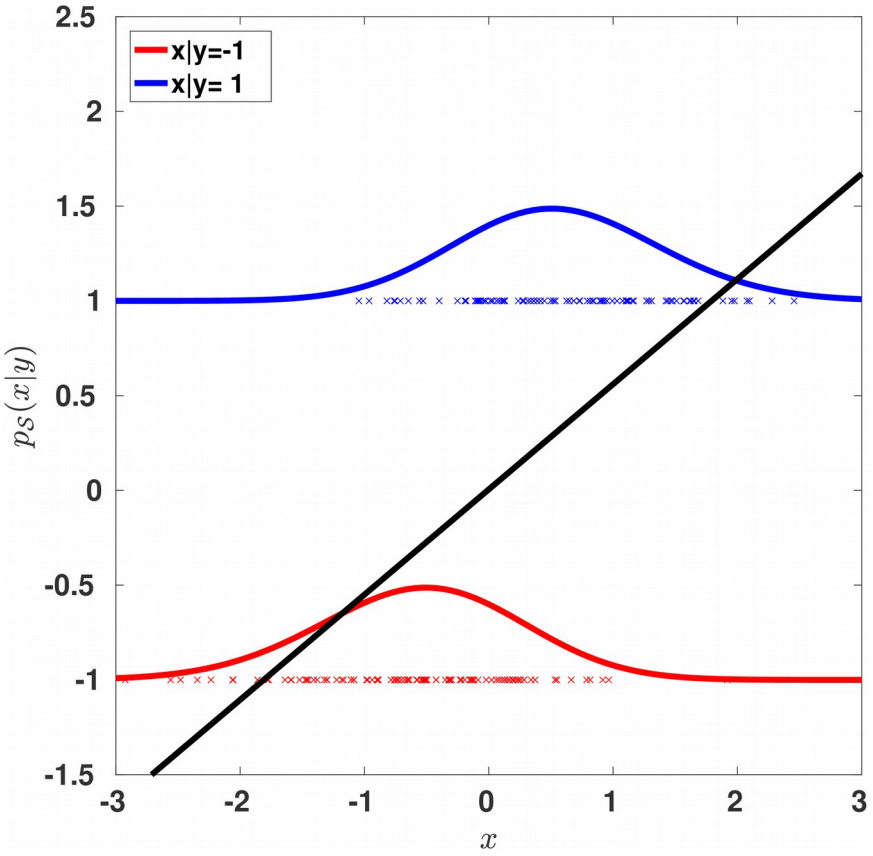
– **Example**

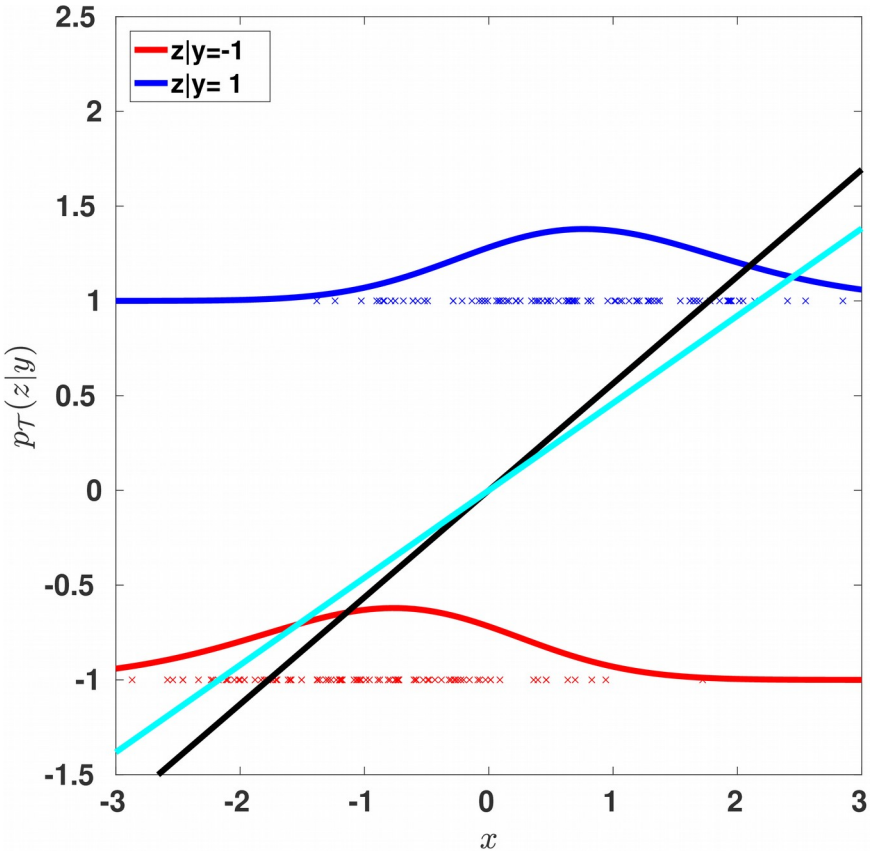




Weighted classifier

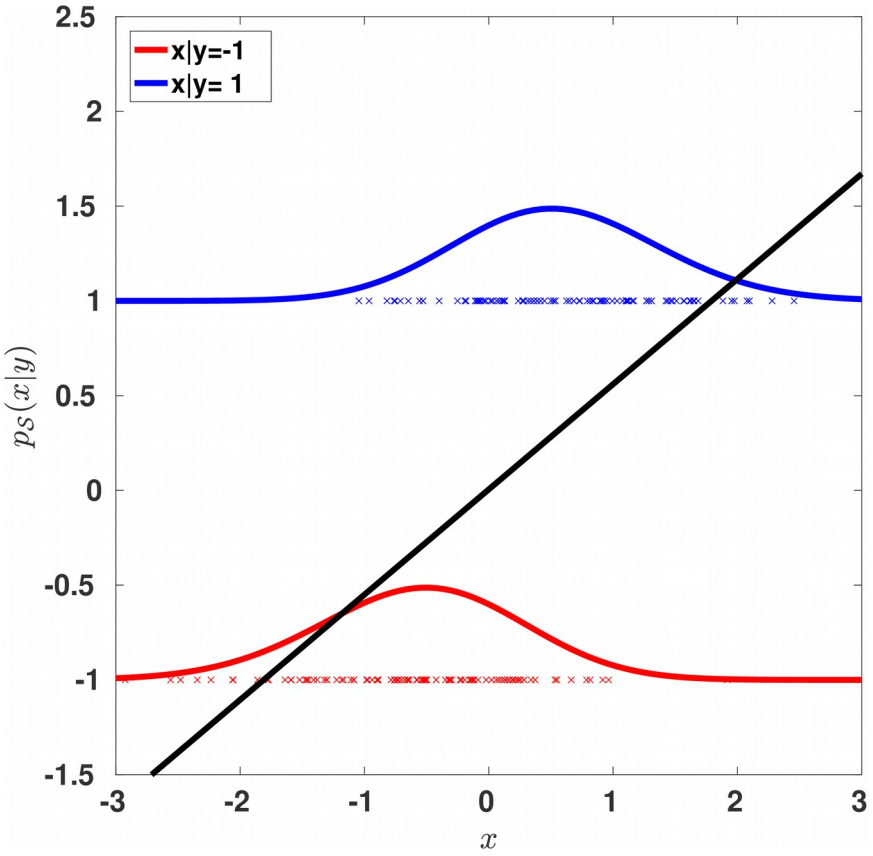
– **Example**

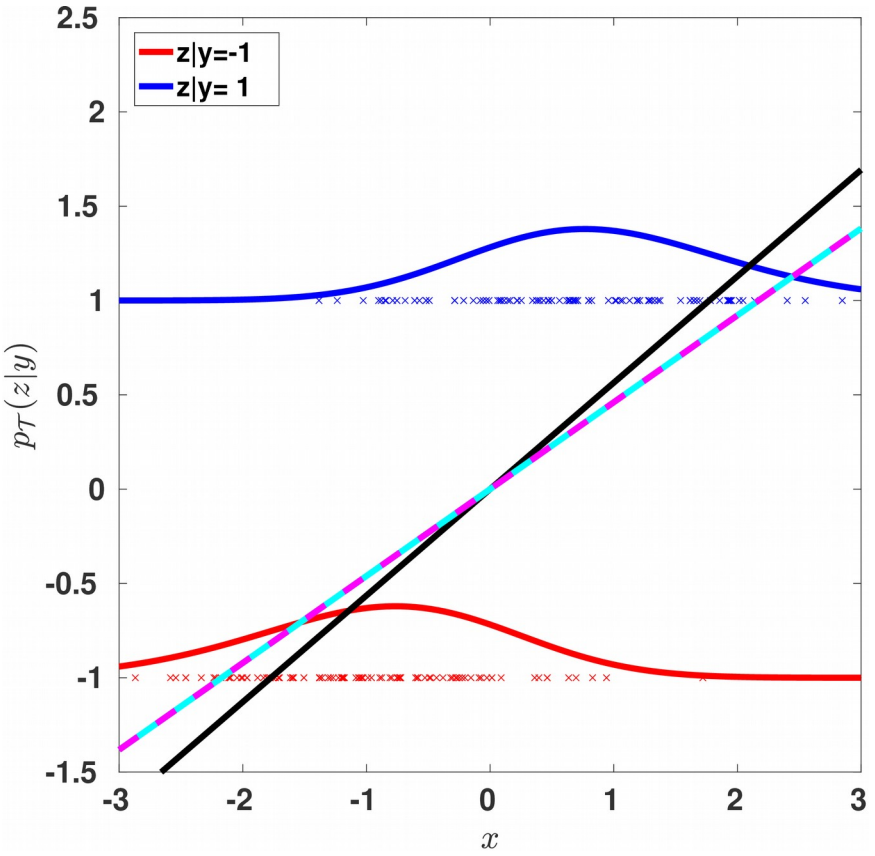




Weighted classifier

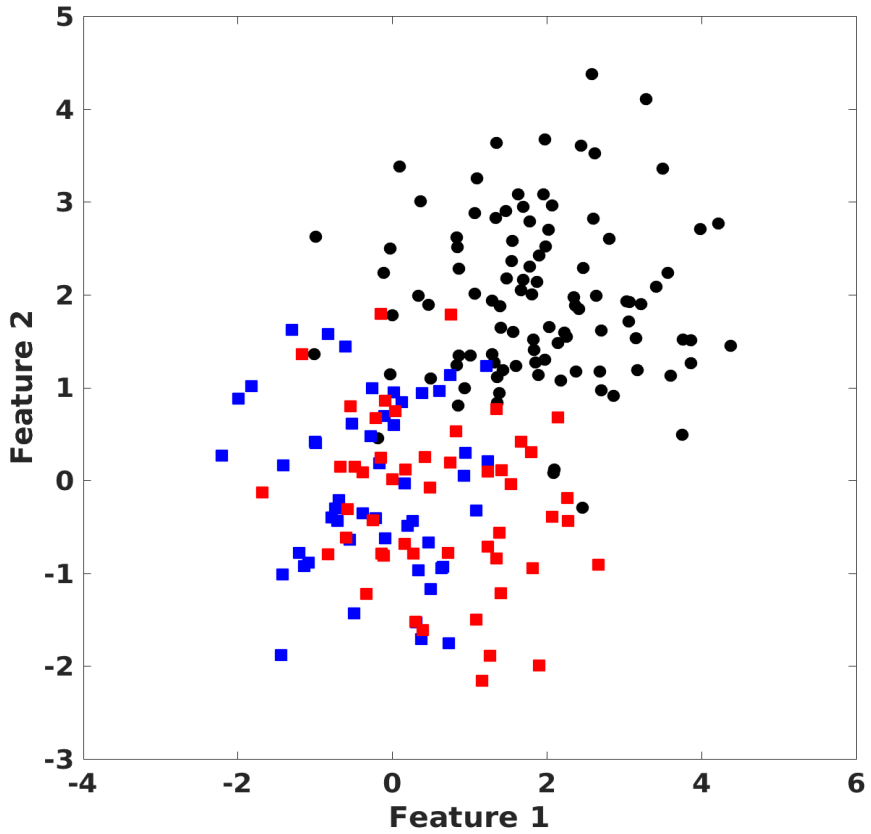
– **Example**





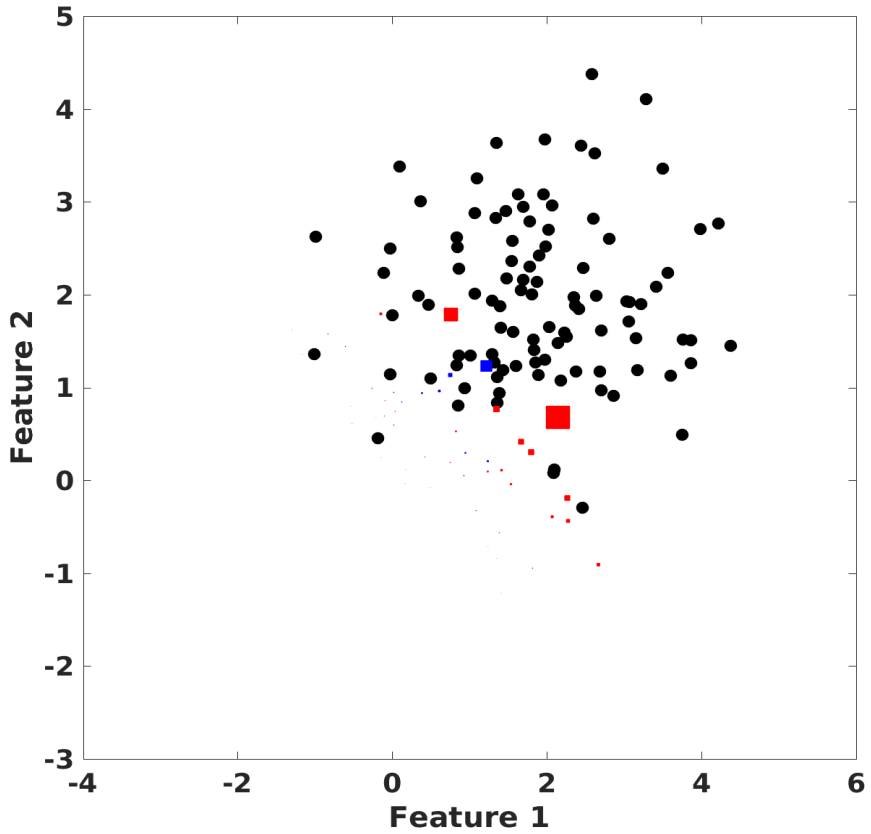
Problem

- **However, the weights introduce a problem:**
  - The target samples that lie closest to the source distribution receive large weights, while other samples receive small weights.



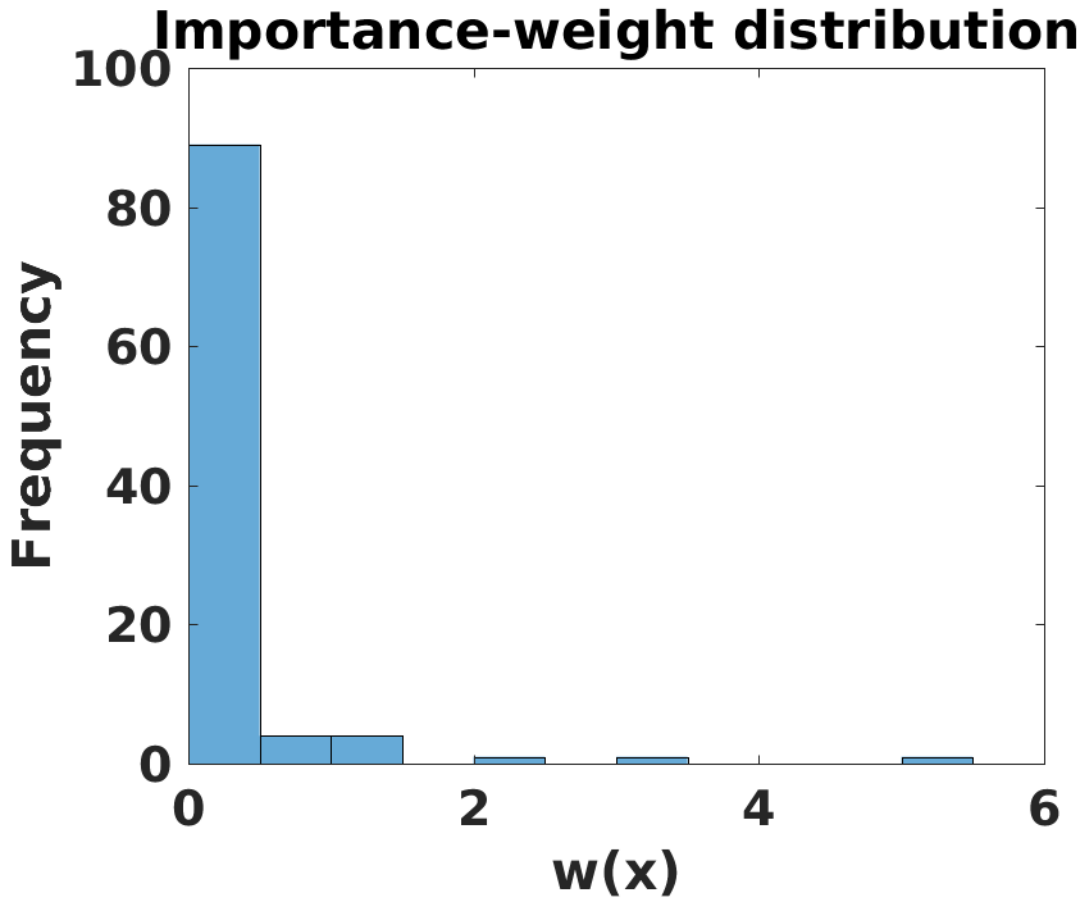
Problem

- **However, the weights introduce a problem:**
  - The target samples that lie closest to the source distribution receive large weights, while other samples receive small weights.



Problem

– **A good visualization of this problem is by taking the histogram of your estimated importance-weights:**



Problem

- **The bimodal weights effectively reduce the sample size of your training set.**
  - Leads to a highly variable, and often pathological, classifier.

- **This is not an estimation problem.**
  - Even if you know the distributions completely, there are problem settings where you can not resort to using the probability ratio.

- **What could we try next?**

Distribution matching

- **The goal of importance-weighing is to match the source and target distribution as much as possible:**

$$p_{\mathcal{T}}(x) \approx w(x) p_{\mathcal{S}}(x)$$

- **If I have a function that describes divergence, then I can also find weights by optimization:**

$$\begin{aligned} \underset{w \in W}{\text{minimize}} \quad & D\big(p_{\mathcal{S}}(x) w, p_{\mathcal{T}} \big) \\ \text{s.t.} \quad & w \geq 0 \end{aligned}$$

- **The weights don't depend on a distribution assumption**

  → nonparametric estimator.

Distribution matching

- **There are a lot of choices for the divergence measure:**
  - Kullback-Leibler divergence
  - Total Variation distance
  - Bhattacharyya distance
  - Wasserstein metric (Earth's move distance)
  - Kolmogorov-Smirnoff statistic
- **But we are going to use a measure based on the two-sample problem.**

Maximum Mean Discrepancy

TU Delft

- **The two-sample problem is concerned with testing whether two sets of samples stem from the same distribution.**

- **The idea is that if they *are* from the same distribution, then *their expected values should be same*, regardless of applying any continuous transformation.**

  - The distance between the expected values is 0 iff $p_S = p_T$:

$$\mathrm{MMD}[\mathcal{F}, p_{\mathcal{S}}, p_{\mathcal{T}}] = \sup_{f \in \mathcal{F}} \|\mathbb{E}_{p_{\mathcal{S}}} f(x) - \mathbb{E}_{p_{\mathcal{T}}} f(z)\|^2$$

  - But you cannot evaluate this function.

Fortet & Mourier, 1953

Maximum Mean Discrepancy

TU Delft

- **The measure uses distributions instead of samples.**
  - Approximate expectations with sample averages:

$$\hat{\mathrm{MMD}}[\mathcal{F}, X, Z] = \sup_{f \in \mathcal{F}} \| \frac{1}{n} \sum_{i=1}^{n} f(x_i) - \frac{1}{m} \sum_{j=1}^{m} f(z_j) \|^2$$

where $X \sim p_{\mathcal{S}}$ and $Z \sim p_{\mathcal{T}}$

  - The empirical MMD converges to the true MMD as both sample sizes increase (n,m → infinity)

Borgwardt et al, 2006

Maximum Mean Discrepancy

TU Delft

- *F* **refers to the class of all continuous functions.**

- **This class is way too large to search over.**
  - We're going to approximate it using kernels and Hilbert space:

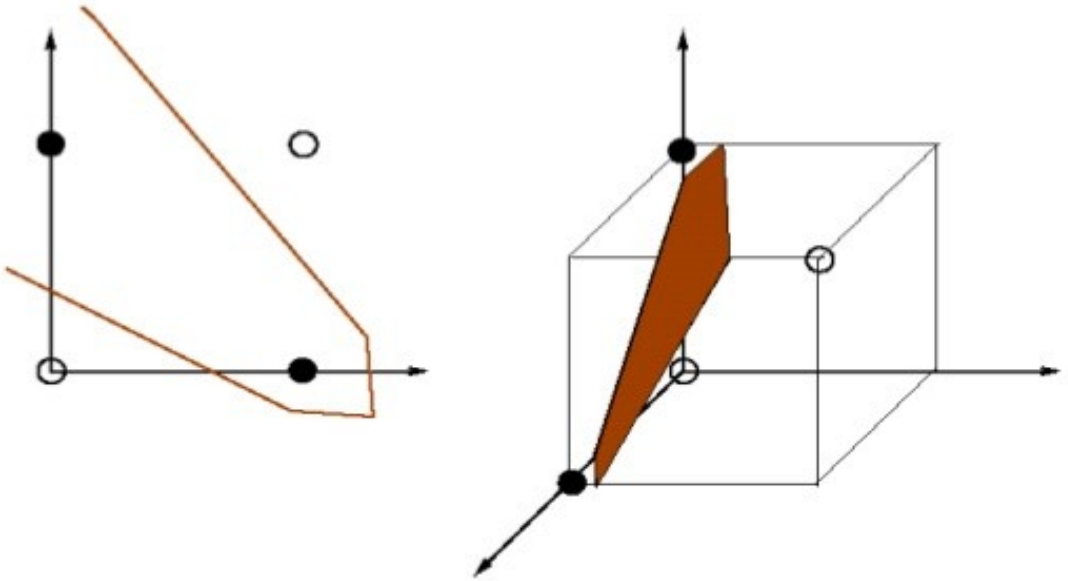
$$\hat{\mathrm{MMD}}[\mathcal{H}, X, Z] = \| \frac{1}{n}\sum_{i=1}^{n}\phi(x_i) - \frac{1}{m}\sum_{j=1}^{m}\phi(z_j) \|^2$$

$$= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{i'=1}^{n}K(x_i, x_i') - \frac{2}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m}K(x_i, z_j) + \frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{m}K(z_j, z_j')$$

Borgwardt et al, 2006

Kernels

- A quick note on kernels:
  - There are some problems where you would like to map your features to a higher-dimensional space.



$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) \rightarrow \begin{bmatrix} x_1 x_1 \\ x_2 x_2 \\ x_1 x_2 \end{bmatrix}$$

Kernels

- **The higher the dimensionality, the more likely that your dataset becomes linearly separable.**
  - So, what about an infinite number of dimensions?

$$\phi(x) = \exp(-\frac{1}{2}\|x\|^2)\left[\frac{x^0}{0!} \quad \frac{x^1}{1!} \quad \frac{x^2}{2!} \quad \dots\right]$$

  - The kernel trick consists of applying the inner product of bases:

$$\phi(x)\phi(x')^\top = \exp(-\frac{1}{2}\|x\|^2)\exp(-\frac{1}{2}\|x'\|^2)\sum_{j=0}^{\infty}\frac{(x^\top x)^j}{j!}$$

  - Leading to the radial basis function:

$$K(x, x') = \exp(-\frac{1}{2}\|x - x'\|^2)$$

www.cs.toronto.edu/~duvenaud/cookbook/index.html

Kernel Mean Matching

– **Going back to the original nonparametric weight estimator and plugging in the empirical MMD:**

$$\begin{aligned} \underset{w \in W}{\text{minimize}} \quad & \hat{\text{MMD}}[\mathcal{H}, wX, Z] \\ \text{s.t.} \quad & w \geq 0 \end{aligned}$$

– Expanding it:

$$\begin{aligned} w* &= \arg\min_{w \in W} \hat{\text{MMD}}[\mathcal{H}, wX, Z] \\ &= \arg\min_{w \in W} \frac{1}{n^2} w^\top K(x_i, x_{i'}) w - \frac{2}{nm} \sum_{j=1}^{m} w^\top K(x_i, z_j) + \frac{1}{m^2} \sum_{j=1}^{m} \sum_{j'=1}^{m} K(z_j, z_{j'}) \\ &= \arg\min_{w \in W} \frac{1}{n^2} w^\top K w - \frac{2}{nm} w^\top \kappa \end{aligned}$$

Huang et al., 2006

Kernel Mean Matching

- **Now we have a complete optimization problem, right?**
  - Notice anything that might be problematic?

$$w* = \arg\min_{w \in W} \frac{1}{n^2} w^\top K w - \frac{2}{nm} w^\top \kappa$$

  - This problem has a trivial solution: w = **0**

Kernel Mean Matching

– **When moving from parametric to nonparametric weights, we ignored the following property:**

$$
\begin{aligned}
1 &= \int_{\Omega} p_{\mathcal{T}}(x)\ \mathrm{d}x \\
&= \int_{\Omega} \frac{p_{\mathcal{T}}(x)}{p_{\mathcal{S}}(x)} p_{\mathcal{S}}(x)\ \mathrm{d}x \\
&= \int_{\Omega} w(x) p_{\mathcal{S}}(x)\ \mathrm{d}x \\
&\approx \frac{1}{n} \sum_{i=1}^{n} w(x_i)
\end{aligned}
$$

Kernel Mean Matching

- **We can incorporate this property as a constraint.**
  - The approximately equivalent property can be achieved through constraining the absolute distance between the average and 1:

$$|\frac{1}{n}\sum_{i=1}^{n} w(x_i) - 1| \leq \epsilon$$

- **Leading to the kernel mean matching weight estimator:**

$$\begin{aligned} \underset{w \in W}{\text{minimize}} \quad & \frac{1}{n^2} w^\top K w - \frac{2}{nm} w^\top \kappa \\ \text{s.t.} \quad & w \geq 0 \\ & |\frac{1}{n}\sum_{i=1}^{n} w(x_i) - 1| \leq \epsilon \end{aligned}$$

Kernel Mean Matching

– **This estimator can be regularized:**

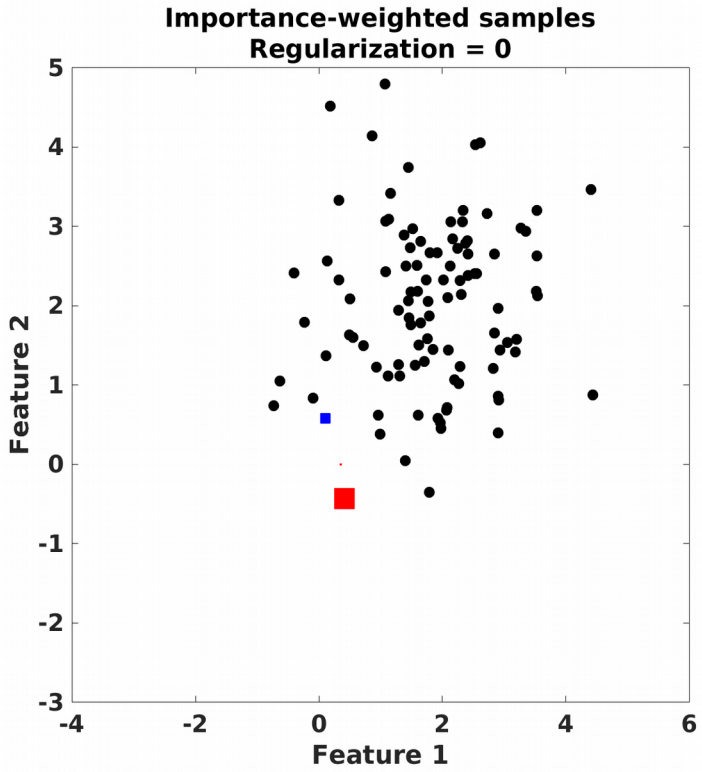
$$w^* = \arg\min_{w \in W} \frac{1}{n^2} w^\top K(x, x')w - \frac{2}{nm} \sum_{j=1}^{m} K(x, z_j) + \lambda \|w\|^2$$

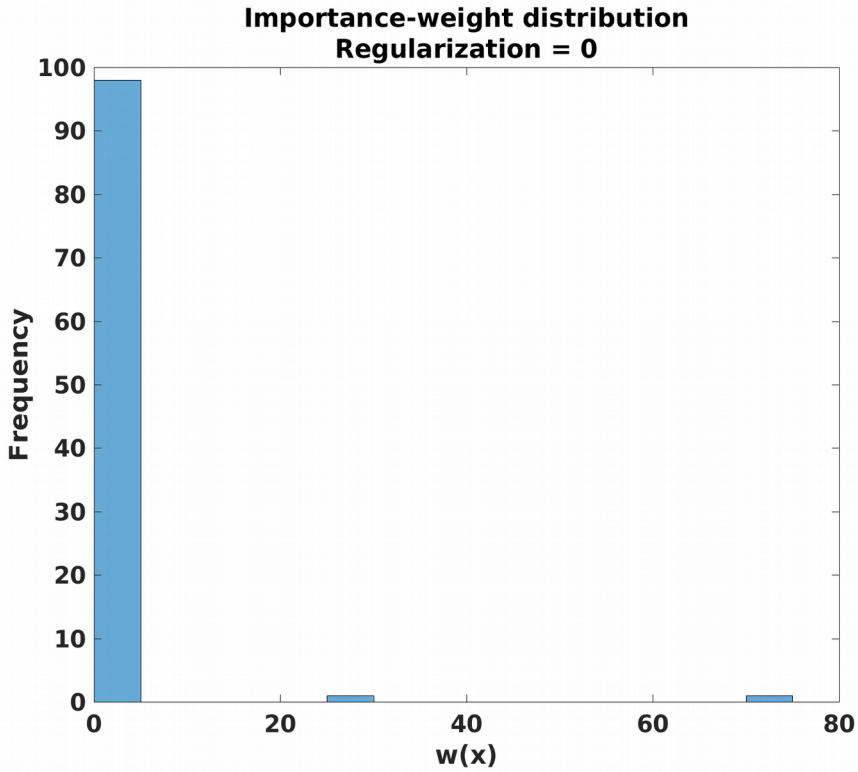
$$= \arg\min_{w \in W} \frac{1}{n^2} w^\top \big(K(x, x') + \lambda I\big)w - \frac{2}{nm} \sum_{j=1}^{m} K(x, z_j)$$

– Which gives you a knob to turn to make the overall weight distribution concentrate more around its average.

Kernel Mean Matching

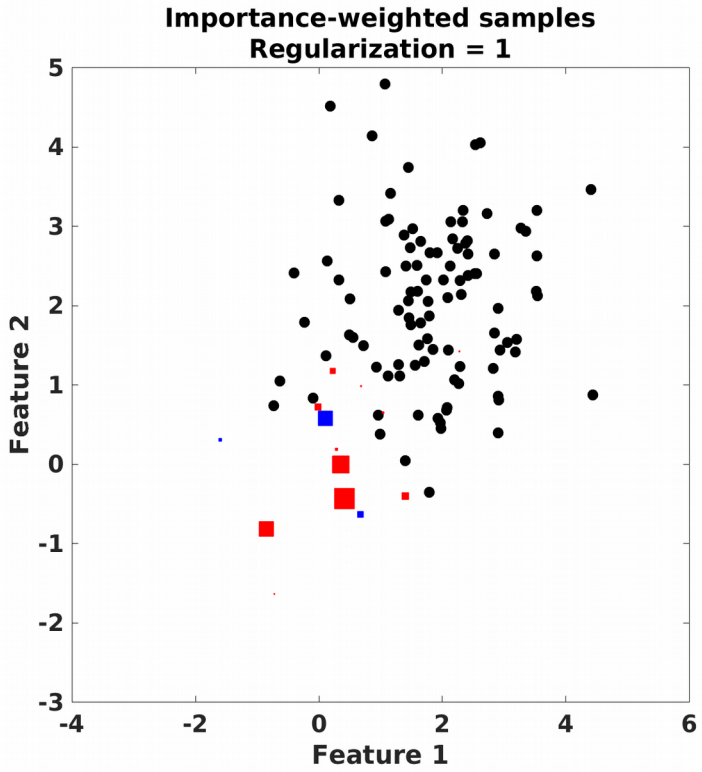
– **Increasing the amount of regularization:**

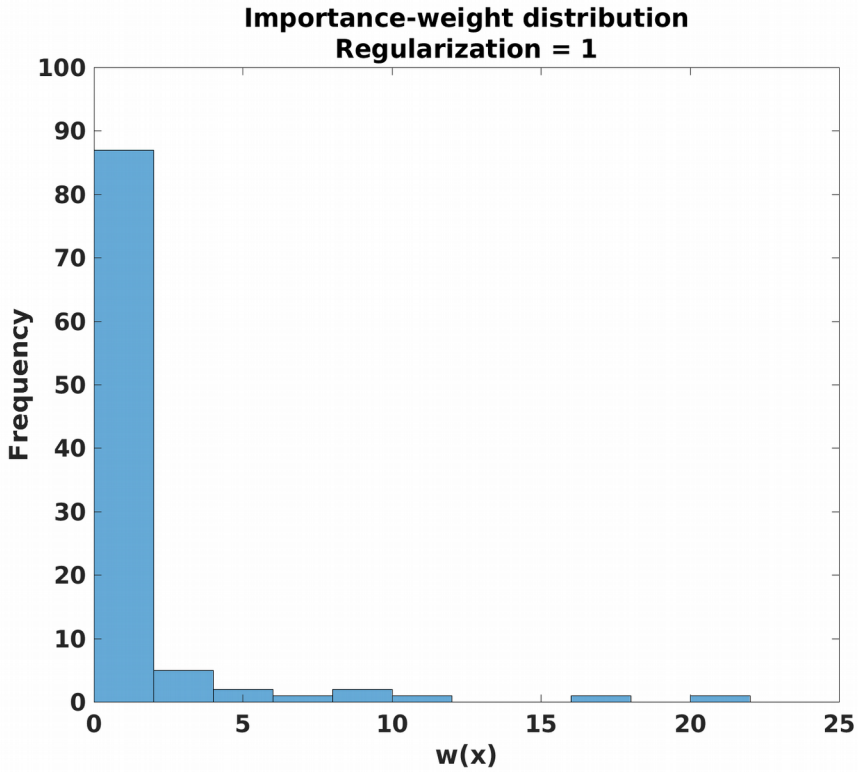




Kernel Mean Matching

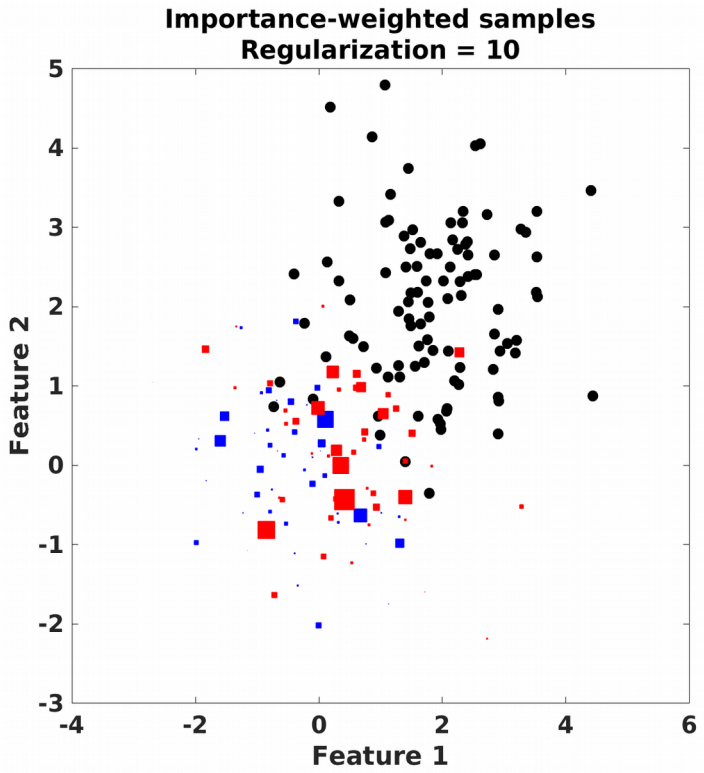
– **Increasing the amount of regularization:**

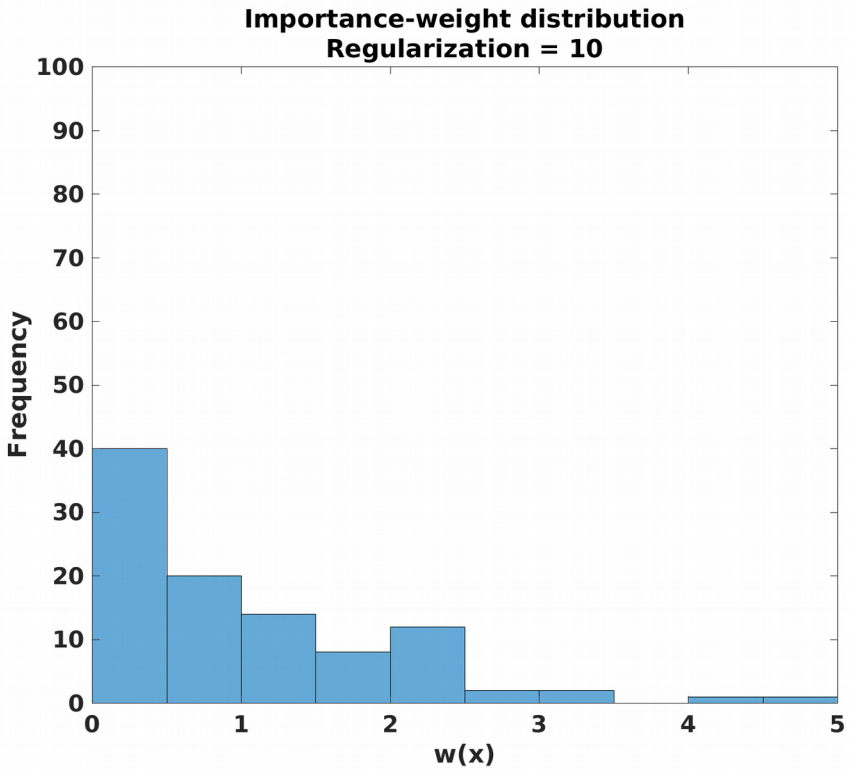




Kernel Mean Matching

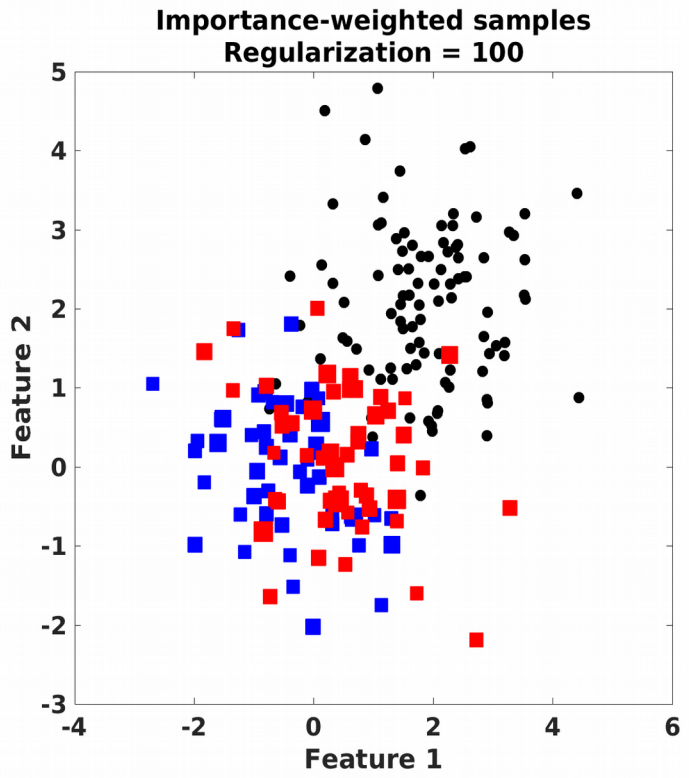
– **Increasing the amount of regularization:**

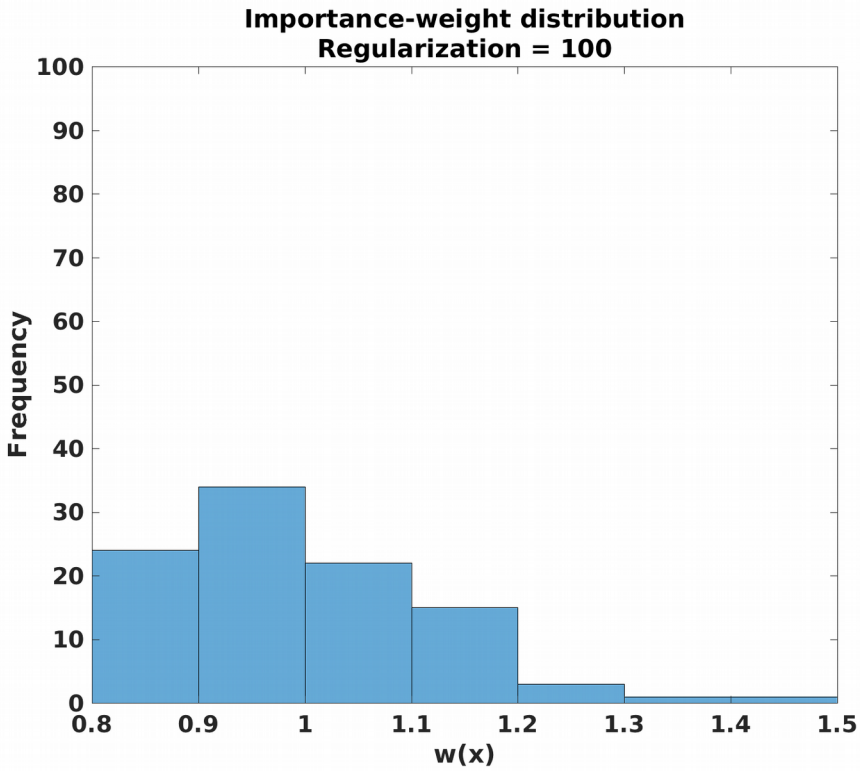




Kernel Mean Matching

– **Increasing the amount of regularization:**





Kernel Mean Matching

- **Note that the kernel hyperparameters introduce a bias in the weights.**
  - But one could argue that trading off variance for bias actually benefits the weighted classifier later on.
- **Also, the nonparametric estimator only matches the distribution approximately.**
  - But it still converges to the true weighting as more samples become available.

Recap

- **Covariate shift is a supervised learning setting with training and test data coming from different distributions.**
- **A classifier can be adapted by importance-weighing.**
- **Importance-weights can be problematic.**
- **One could design nonparametric weight estimators through distribution divergences.**
- **MMD measures discrepancy between two sets of samples.**
- **Kernel Mean Matching uses MMD to find weights.**