# Support vector comparison machines

Toby Dylan Hocking
toby.hocking@mail.mcgill.ca
joint work with David Venuto, Lakjaree Sphanurattana, and Masashi Sugiyama

2 March 2018

# Motivating example: learning to compare sushi

salmon is better than eel

fatty tuna is as good as crab liver

If I give you another sushi pair,
can you tell me which one is better,
or if they are equally good?

# Motivating example: learning to compare sushi

salmon is better than eel

fatty tuna is as good as crab liver

If I give you another sushi pair,
can you tell me which one is better,
or if they are equally good?

# Learning a comparison function

We are given $n$ training pairs $(\mathbf{x}_i, \mathbf{x}'_i, y_i)$

- Input: a pair of feature vectors $\mathbf{x}_i, \mathbf{x}'_i \in \mathbb{R}^p$
  e.g. sushi fattiness, taster birthplace.
- Output: a label $y_i = \begin{cases} -1 & \text{if } \mathbf{x}_i \text{ is better} \\ 0 & \text{if } \mathbf{x}_i \text{ is as good as } \mathbf{x}'_i \\ 1 & \text{if } \mathbf{x}'_i \text{ is better.} \end{cases}$

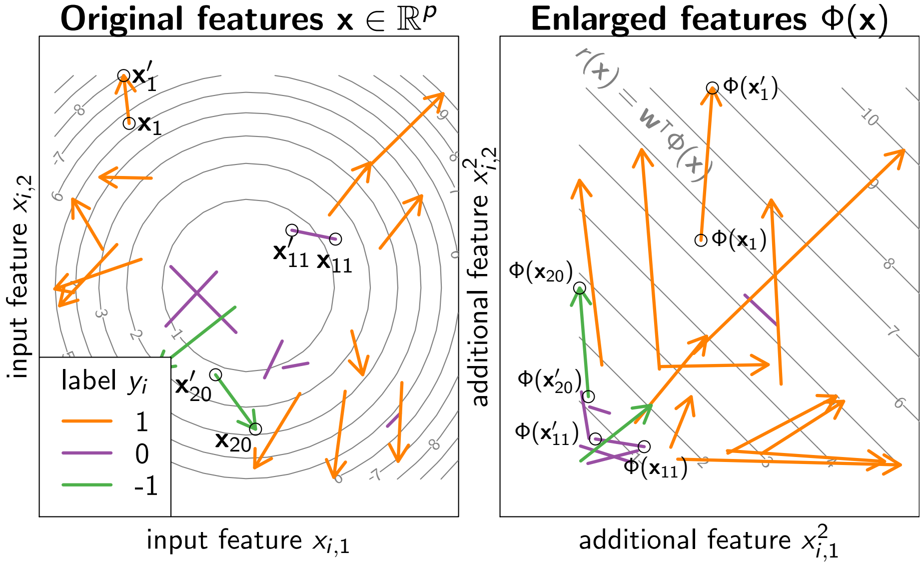
Goal: find a comparison function $c : \mathbb{R}^p \times \mathbb{R}^p \to \{-1, 0, 1\}$

- Good prediction with respect to the zero-one loss:

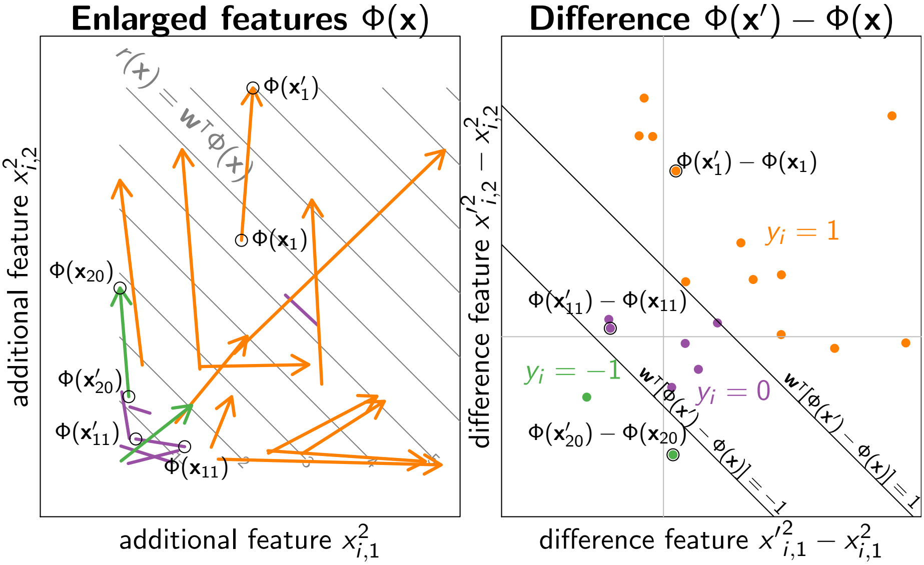
$$\underset{c}{\text{minimize}} \sum_{i \in \text{test}} I\left[y_i \neq c(\mathbf{x}_i, \mathbf{x}'_i)\right]$$

- Symmetry: $c(\mathbf{x}, \mathbf{x}') = -c(\mathbf{x}', \mathbf{x})$.

# Geometric interpretation when $r(\mathbf{x}) = ||\mathbf{x}||_2^2$



# Geometric interpretation when $r(\mathbf{x}) = ||\mathbf{x}||_2^2$



# Related work: rank and rate

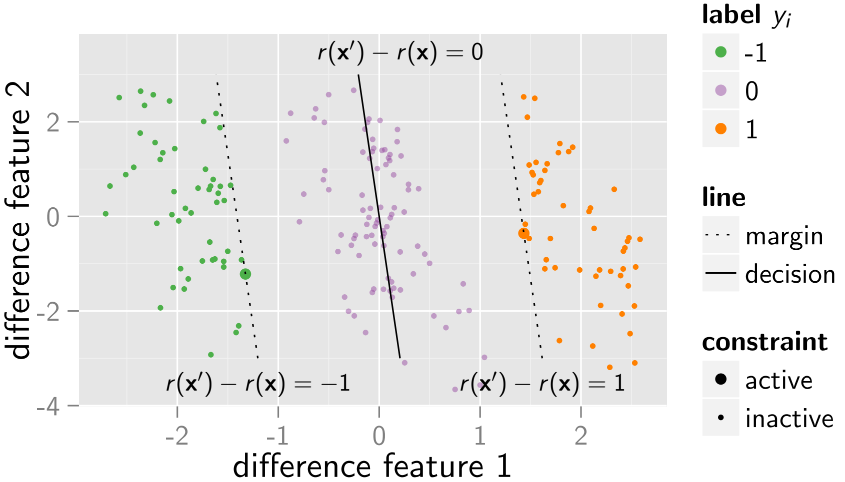
| Inputs / Outputs | single items $\mathbf{x}$ | pairs of items $\mathbf{x}, \mathbf{x}'$ |
|---|---|---|
| $y \in \{-1, 1\}$ | SVM | SVMrank |
| $y \in \{-1, 0, 1\}$ | | this work |

- T Joachims. Optimizing search engines using clickthrough data. KDD 2002. (SVMrank)
- K Zhou *et al.* Learning to rank with ties. SIGIR 2008. (boosting, ties are more effective with more output values)
- R Herbrich *et al.* TrueSkill: a Bayesian skill rating system. NIPS 2006. (generalization of Elo for chess)

# SVMrank ignores equality $y_i = 0$ pairs

Linear $r(\mathbf{x}) = \mathbf{w}^\intercal \mathbf{x}$.

$$\begin{aligned} \underset{\mathbf{w}\in\mathbb{R}^p}{\text{minimize}} \quad & \mathbf{w}^\intercal \mathbf{w} \\ \text{subject to} \quad & \mathbf{w}^\intercal(\mathbf{x}_i' - \mathbf{x}_i) y_i \geq 1, \ \forall i \text{ such that } y_i \in \{-1, 1\}. \end{aligned}$$



Introduction and related work

Learning a max-margin comparison function

Results and conclusions

# Learning to rank and compare

We will learn a

- Ranking function $r : \mathbb{R}^p \to \mathbb{R}$. Bigger is better.
- Threshold $\tau \in \mathbb{R}^+$.
  A small difference $|r(\mathbf{x}') - r(\mathbf{x})| \leq \tau$ is not significant.
- Comparison function $c_\tau(\mathbf{x}, \mathbf{x}') = \begin{cases} -1 & \text{if } r(\mathbf{x}') - r(\mathbf{x}) < -\tau \\ 0 & \text{if } |r(\mathbf{x}') - r(\mathbf{x})| \leq \tau \\ 1 & \text{if } r(\mathbf{x}') - r(\mathbf{x}) > \tau. \end{cases}$

Fix the threshold $\tau = 1$. The problem becomes

$$\underset{r}{\text{minimize}} \sum_{i=1}^{n} I\left[y_i \neq c_1(\mathbf{x}_i, \mathbf{x}_i')\right].$$

If there are several $r$ that achieve 0 error,
then the data are separable.

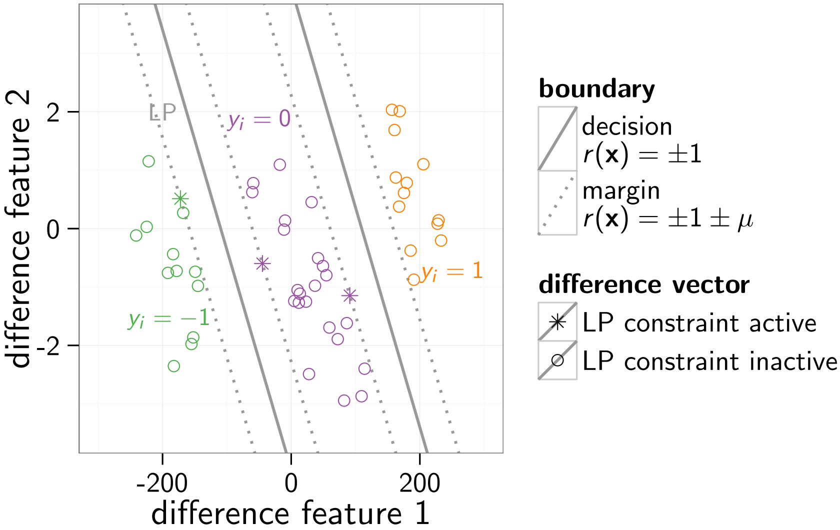
# Max margin LP for separable data

Linear Program (LP) measures ranking function values:

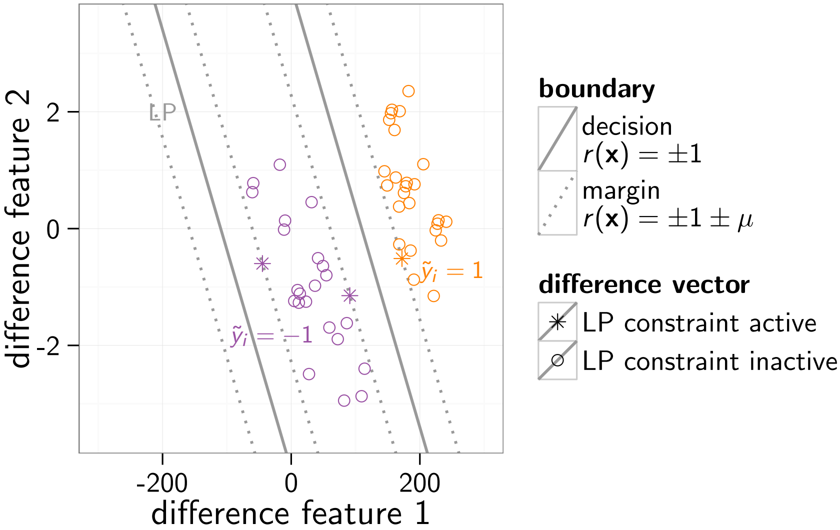
$$\begin{aligned} \underset{\mu\in\mathbb{R},\mathbf{w}\in\mathbb{R}^p}{\text{maximize}}\ & \mu \\ \text{subject to}\ & \mu \leq 1 - |\mathbf{w}^\intercal(\mathbf{x}_i' - \mathbf{x}_i)|,\ \forall\ i \text{ such that } y_i = 0, \\ & \mu \leq -1 + \mathbf{w}^\intercal(\mathbf{x}_i' - \mathbf{x}_i)y_i,\ \forall\ i \text{ such that } y_i \in \{-1, 1\}. \end{aligned}$$

- If the optimal margin $\mu > 0$ then the data are separable.
- Ranking function $r(\mathbf{x}) = \mathbf{w}^\intercal\mathbf{x}$.
- Comparison function $c_1(\mathbf{x}, \mathbf{x}')$.

# Geometric interpretation of max margin LP



# Equivalent: $(\mathbf{x}_i, \mathbf{x}_i', y_i = -1)$ flipped to $(\mathbf{x}_i', \mathbf{x}_i, \tilde{y}_i = 1)$



# Max margin SVM QP for separable data

Change of variables "flipped data"

$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}'_{-1} \\ \mathbf{X}_0 \\ \mathbf{X}'_0 \end{bmatrix}, \quad \tilde{\mathbf{X}}' = \begin{bmatrix} \mathbf{X}'_1 \\ \mathbf{X}_{-1} \\ \mathbf{X}'_0 \\ \mathbf{X}_0 \end{bmatrix}, \quad \tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{1}_{|\mathcal{I}_1|} \\ \mathbf{1}_{|\mathcal{I}_{-1}|} \\ -\mathbf{1}_{|\mathcal{I}_0|} \\ -\mathbf{1}_{|\mathcal{I}_0|} \end{bmatrix},$$

- $\tilde{y}_i = -1$ implies no significant difference between $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}'_i$,
- $\tilde{y}_i = 1$ implies that $\tilde{\mathbf{x}}'_i$ is better than $\tilde{\mathbf{x}}_i$.

Quadratic Program measures weight vector size (SVM QP)

$$\begin{aligned} \underset{\mathbf{u}\in\mathbb{R}^p,\beta\in\mathbb{R}}{\text{minimize}} \quad & \mathbf{u}^\intercal \mathbf{u} \\ \text{subject to} \quad & \tilde{y}_i(\beta + \mathbf{u}^\intercal(\tilde{\mathbf{x}}'_i - \tilde{\mathbf{x}}_i)) \geq 1, \ \forall i \in \{1, \dots, m\}. \end{aligned}$$

Same as SVM: learn affine function $f(\mathbf{x}) = \beta + \mathbf{u}^\intercal \mathbf{x}$.
**Lemma:** $\hat{\mu} = -1/\beta$, $\hat{\mathbf{w}} = -\mathbf{u}/\beta$ are feasible for the LP.

# Max margin SVM QP for separable data

Change of variables "flipped data"

$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}'_{-1} \\ \mathbf{X}_0 \\ \mathbf{X}'_0 \end{bmatrix}, \quad \tilde{\mathbf{X}}' = \begin{bmatrix} \mathbf{X}'_1 \\ \mathbf{X}_{-1} \\ \mathbf{X}'_0 \\ \mathbf{X}_0 \end{bmatrix}, \quad \tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{1}_{|\mathcal{I}_1|} \\ \mathbf{1}_{|\mathcal{I}_{-1}|} \\ -\mathbf{1}_{|\mathcal{I}_0|} \\ -\mathbf{1}_{|\mathcal{I}_0|} \end{bmatrix},$$

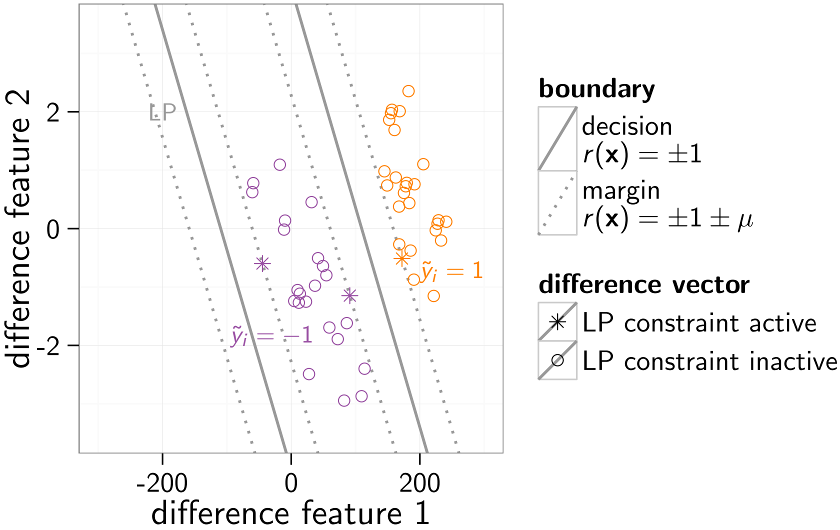
- $\tilde{y}_i = -1$ implies no significant difference between $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}'_i$,
- $\tilde{y}_i = 1$ implies that $\tilde{\mathbf{x}}'_i$ is better than $\tilde{\mathbf{x}}_i$.

Quadratic Program measures weight vector size (SVM QP)

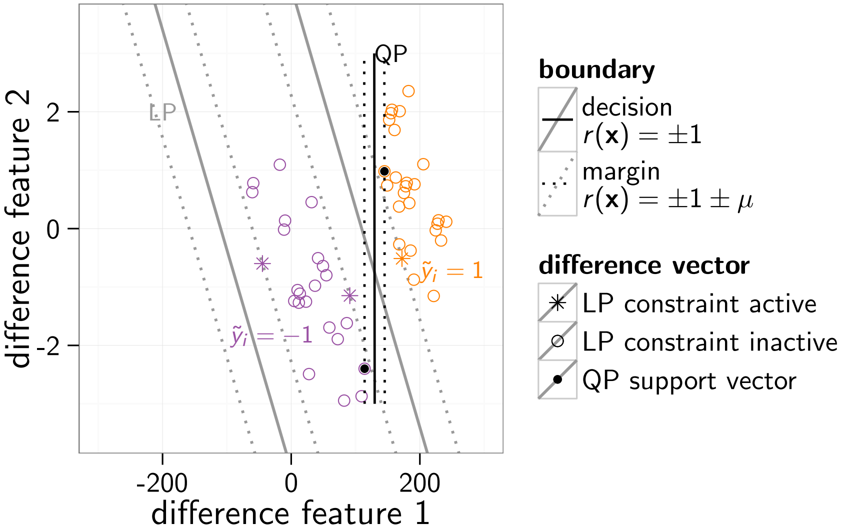
$$\begin{aligned} \underset{\mathbf{u}\in\mathbb{R}^p, \beta\in\mathbb{R}}{\text{minimize}} \quad & \mathbf{u}^\intercal \mathbf{u} \\ \text{subject to} \quad & \tilde{y}_i(\beta + \mathbf{u}^\intercal(\tilde{\mathbf{x}}'_i - \tilde{\mathbf{x}}_i)) \geq 1, \ \forall i \in \{1, \dots, m\}. \end{aligned}$$

Same as SVM: learn affine function $f(\mathbf{x}) = \beta + \mathbf{u}^\intercal \mathbf{x}$.
**Lemma:** $\hat{\mu} = -1/\beta$, $\hat{\mathbf{w}} = -\mathbf{u}/\beta$ are feasible for the LP.

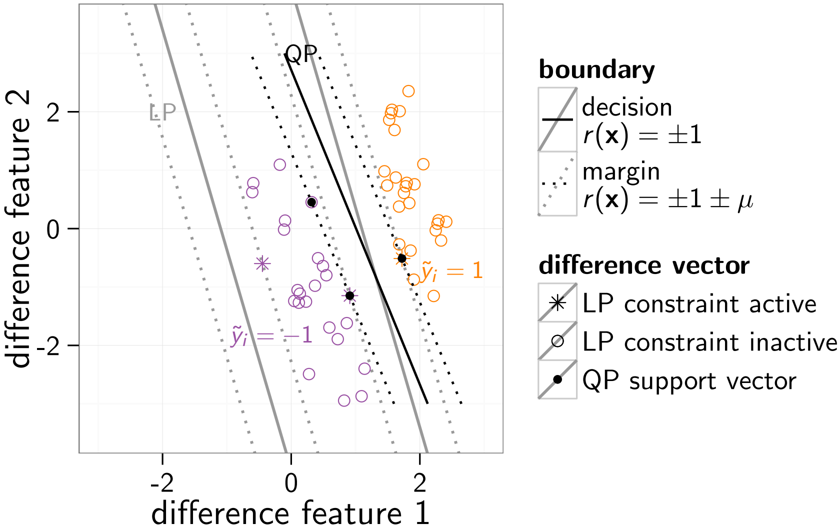
## Equivalent: $(\mathbf{x}_i, \mathbf{x}'_i, y_i = -1)$ flipped to $(\mathbf{x}'_i, \mathbf{x}_i, \tilde{y}_i = 1)$



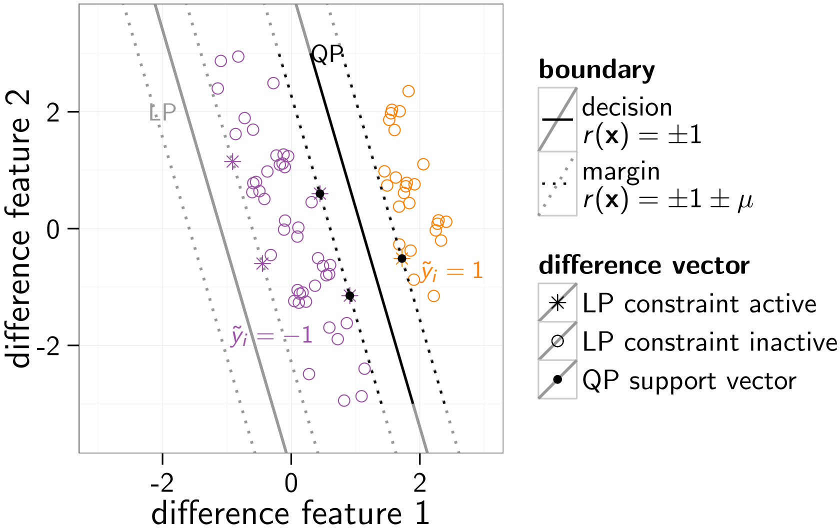
# QP sensitive to feature scale



# $(\mathbf{x}_i, \mathbf{x}'_i, y_i = 0)$ flipped to $(\mathbf{x}_i, \mathbf{x}'_i, \tilde{y}_i = -1)$



## $(\mathbf{x}_i, \mathbf{x}'_i, y_i = 0)$ flipped to $(\mathbf{x}_i, \mathbf{x}'_i, \tilde{y}_i = -1)$, $(\mathbf{x}'_i, \mathbf{x}_i, \tilde{y}_i = -1)$



# Max margin LP and QP for separable data

Linear Program (LP) measures function values:

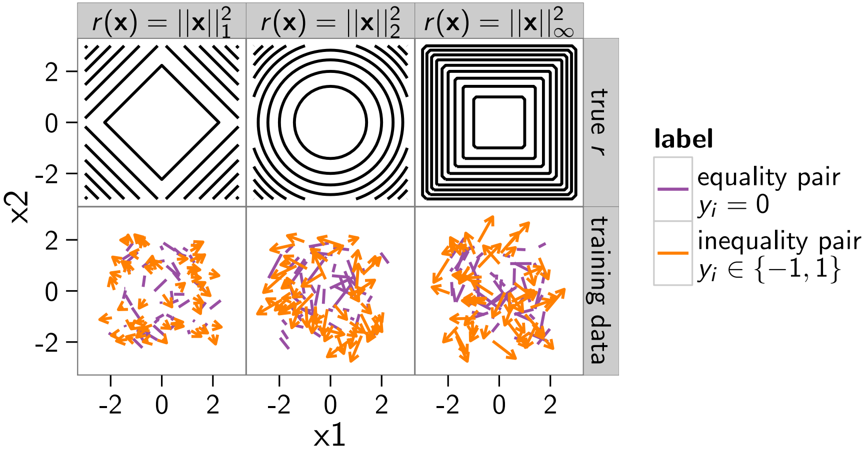
$$\begin{aligned} \operatorname*{maximize}_{\mu\in\mathbb{R},\mathbf{w}\in\mathbb{R}^p} \ & \mu \\ \text{subject to } & \mu \leq 1 - |\mathbf{w}^\intercal(\mathbf{x}_i' - \mathbf{x}_i)|, \ \forall\ i \text{ such that } y_i = 0, \\ & \mu \leq -1 + \mathbf{w}^\intercal(\mathbf{x}_i' - \mathbf{x}_i)y_i, \ \forall\ i \text{ such that } y_i \in \{-1, 1\}. \end{aligned}$$

Quadratic Program (QP) measures weight vector size:

$$\begin{aligned} \operatorname*{minimize}_{\mathbf{u}\in\mathbb{R}^p,\beta\in\mathbb{R}} \quad & \mathbf{u}^\intercal\mathbf{u} \\ \text{subject to} \quad & \tilde{y}_i(\beta + \mathbf{u}^\intercal(\tilde{\mathbf{x}}_i' - \tilde{\mathbf{x}}_i)) \geq 1, \ \forall i \in \{1, \dots, m\}. \end{aligned}$$

- **Lemma:** $\hat{\mu} = -1/\beta$, $\hat{\mathbf{w}} = -\mathbf{u}/\beta$ are feasible for the LP.
- Ranking functions $r_{\text{LP}}(\mathbf{x}) = \mathbf{w}^\intercal\mathbf{x}$, $r_{\text{QP}}(\mathbf{x}) = \hat{\mathbf{w}}^\intercal\mathbf{x}$.
- Comparison function $c_1(\mathbf{x}, \mathbf{x}')$.

# Simulation: true patterns $r$ and noisy training pairs




Validation and test data have the same number of pairs $n$ and the same proportion of equality pairs $\rho$.

# Details of simulation setup

- Inputs $\mathbf{x}_i, \mathbf{x}'_i \in [-3, 3]^2$.
- True ranking function $r(\mathbf{x}) = ||\mathbf{x}||_j^2$ for $j \in \{1, 2, \infty\}$.
- Noisy labels $y_i = t_1[r(\mathbf{x}'_i) - r(\mathbf{x}_i) + \epsilon_i]$.
- Threshold function $t_1(x) = \begin{cases} -1 & \text{if } x < -1, \\ 0 & \text{if } |x| \leq 1, \\ 1 & \text{if } x > 1. \end{cases}$
- Noise $\epsilon_i \sim N(0, \sigma)$ with standard deviation $\sigma = 1/4$.
- Train, validation, and test sets with
  - same number of training pairs $n$, and
  - same proportion of equality pairs $\rho$.
- Fit a $10 \times 10$ grid of models to the training set:
  - Cost parameter $C = 10^{-3}, \dots, 10^3$,
  - Gaussian kernel width $2^{-7}, \dots, 2^4$.
- Select the model with minimal zero-one loss on the validation set.

# We ran 3 different algorithms on each data set

| Input: | equality pairs $\lvert\mathcal{I}_0\rvert$ | — | inequality pairs $\lvert\mathcal{I}_1\rvert + \lvert\mathcal{I}_{-1}\rvert$ | $\rightarrow$ | code |
|---|---|---|---|---|---|
| rank | 0 | | $\lvert\mathcal{I}_1\rvert + \lvert\mathcal{I}_{-1}\rvert$ | $\rightarrow$ | SVMrank |
| rank2 | $2\lvert\mathcal{I}_0\rvert$ | $\leftarrow\rightarrow$ | $2(\lvert\mathcal{I}_1\rvert + \lvert\mathcal{I}_{-1}\rvert)$ | $\rightarrow\rightarrow$ | SVMrank |
| compare | $2\lvert\mathcal{I}_0\rvert$ | — — | $\lvert\mathcal{I}_1\rvert + \lvert\mathcal{I}_{-1}\rvert$ | $\rightarrow$ | proposed |

Equality $y_i = 0$ pairs are shown as — segments.

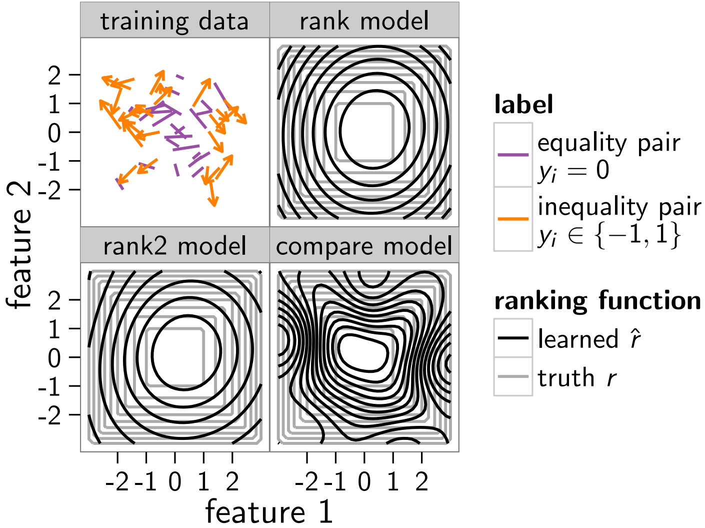
Inequality $y_i \in \{-1, 1\}$ pairs are shown as $\rightarrow$ arrows.

rank ignores each input equality pair.

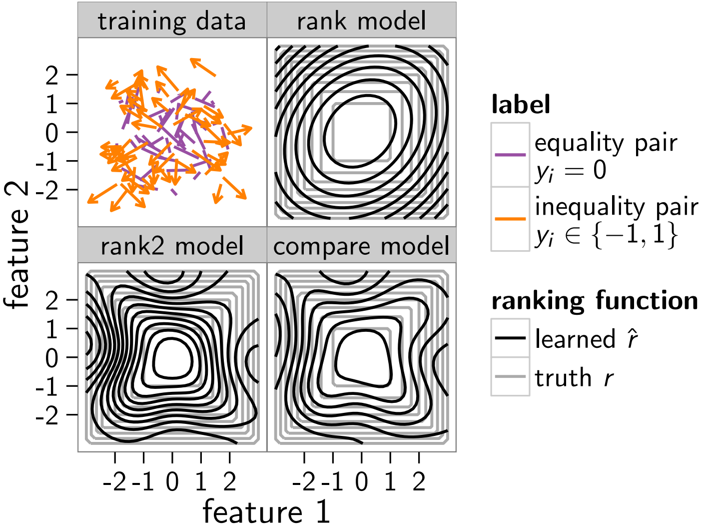
rank2 converts each input equality pair to two contradictory inequality pairs.

compare directly models the equality pairs.

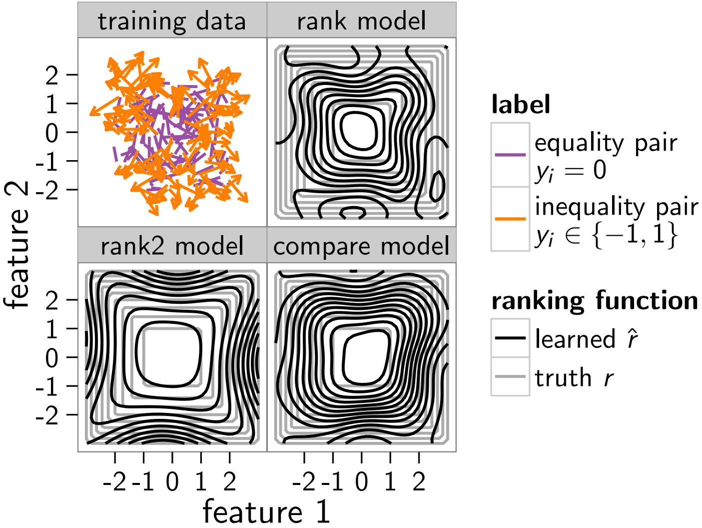
$n = 50$ training pairs and learned $\hat{r} : \mathbb{R}^2 \rightarrow \mathbb{R}$ for simulated square-shaped $r(\mathbf{x}) = ||\mathbf{x}||_\infty^2$ pattern



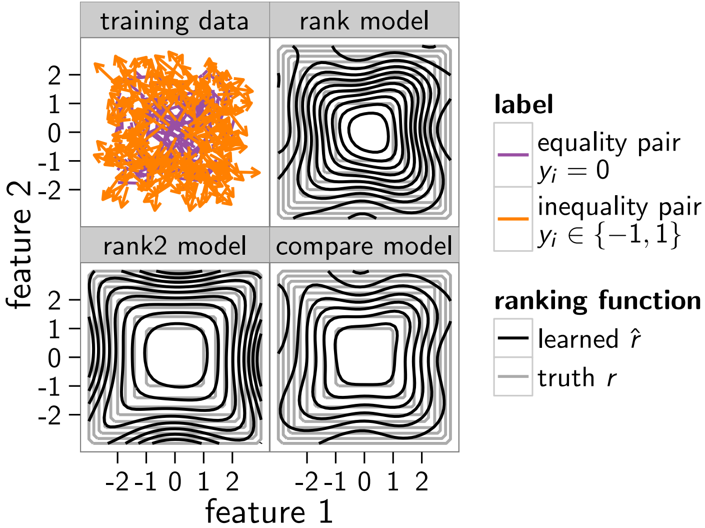
## $n = 100$ training pairs and learned $\hat{r} : \mathbb{R}^2 \rightarrow \mathbb{R}$ for simulated square-shaped $r(\mathbf{x}) = ||\mathbf{x}||_\infty^2$ pattern



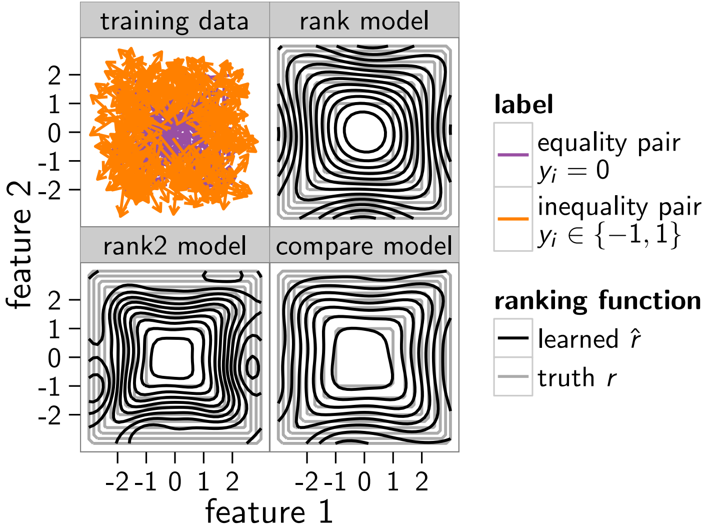
## $n = 200$ training pairs and learned $\hat{r} : \mathbb{R}^2 \to \mathbb{R}$ for simulated square-shaped $r(\mathbf{x}) = ||\mathbf{x}||_\infty^2$ pattern



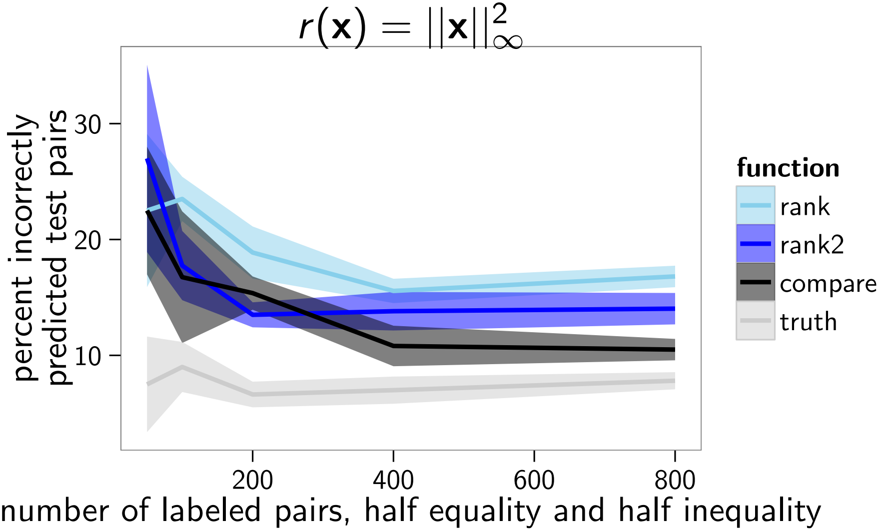
## $n = 400$ training pairs and learned $\hat{r} : \mathbb{R}^2 \rightarrow \mathbb{R}$ for simulated square-shaped $r(\mathbf{x}) = ||\mathbf{x}||_\infty^2$ pattern



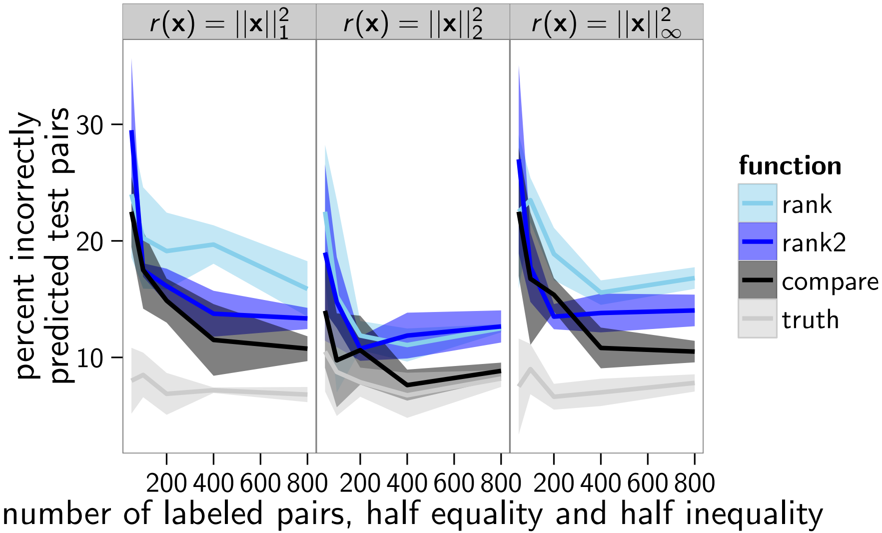
## $n = 800$ training pairs and learned $\hat{r} : \mathbb{R}^2 \rightarrow \mathbb{R}$ for simulated square-shaped $r(\mathbf{x}) = ||\mathbf{x}||_\infty^2$ pattern



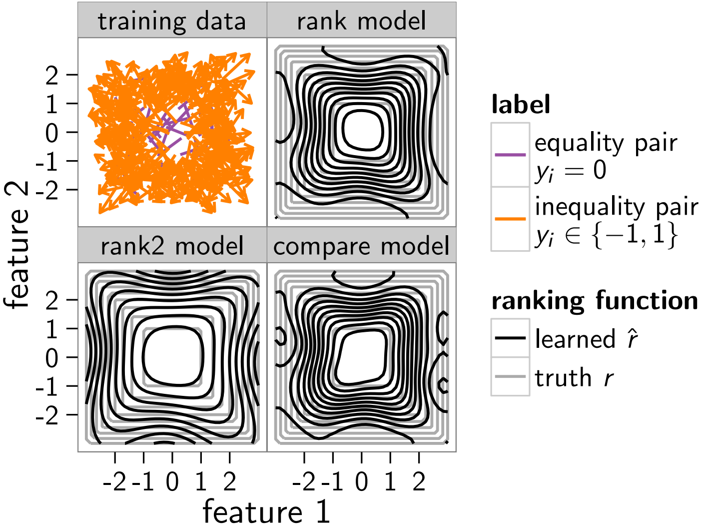
# Test error lowest for proposed SVMcompare model



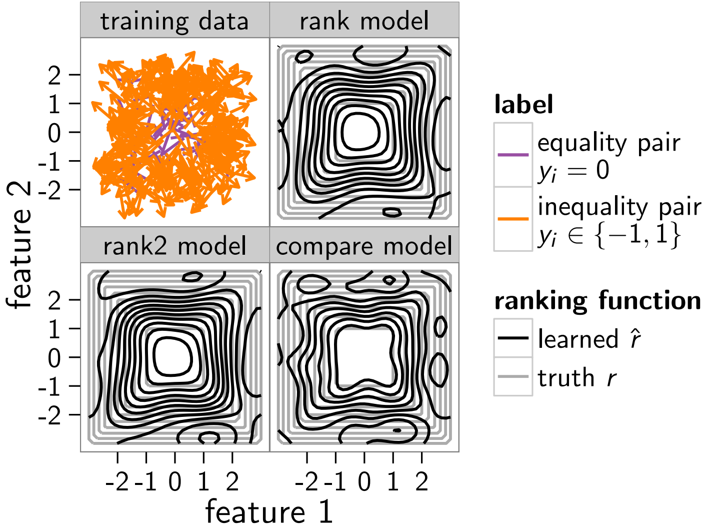
# Test error lowest for proposed SVMcompare model



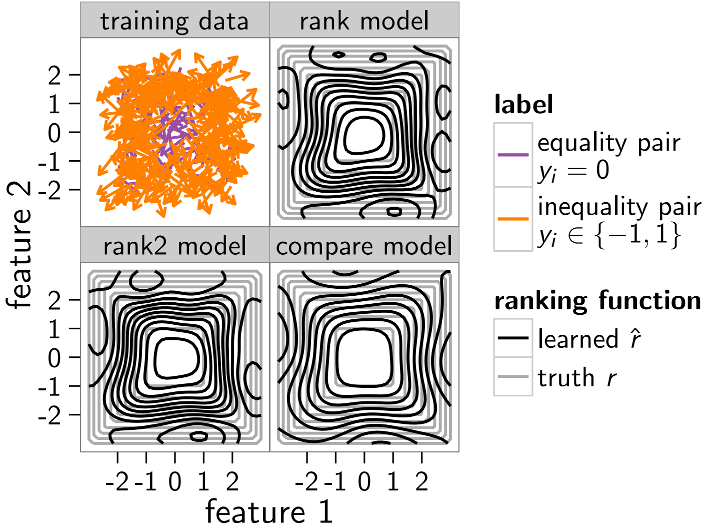
# 10% equality pairs (400 total) and learned $\hat{r} : \mathbb{R}^2 \to \mathbb{R}$ for simulated square-shaped $r(\mathbf{x}) = ||\mathbf{x}||_\infty^2$ pattern



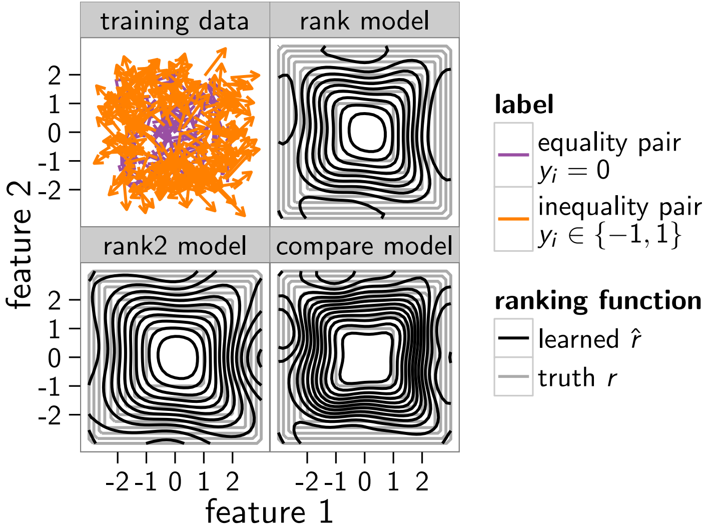
# 20% equality pairs (400 total) and learned $\hat{r} : \mathbb{R}^2 \rightarrow \mathbb{R}$ for simulated square-shaped $r(\mathbf{x}) = ||\mathbf{x}||_\infty^2$ pattern



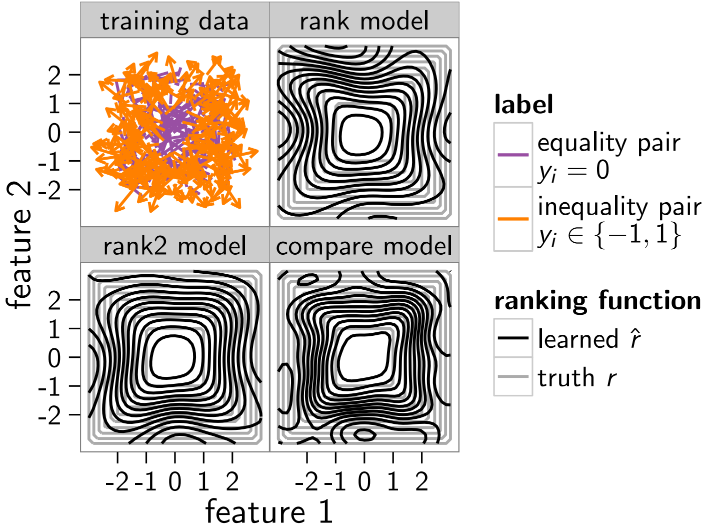
# 30% equality pairs (400 total) and learned $\hat{r} : \mathbb{R}^2 \to \mathbb{R}$ for simulated square-shaped $r(\mathbf{x}) = ||\mathbf{x}||_\infty^2$ pattern



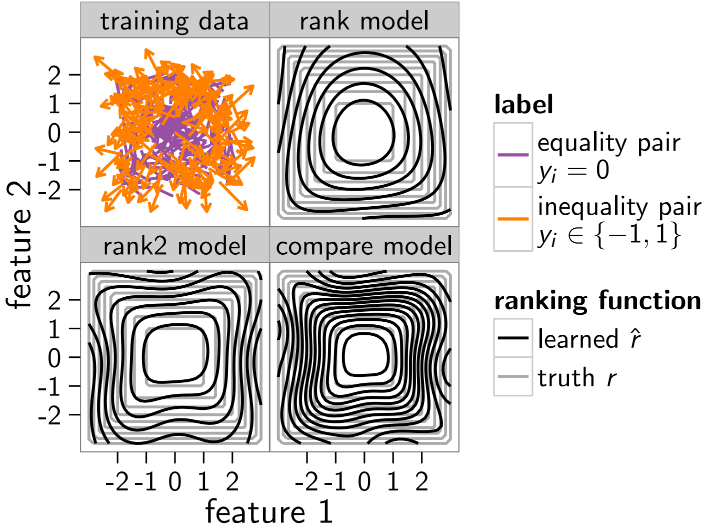
# 40% equality pairs (400 total) and learned $\hat{r} : \mathbb{R}^2 \rightarrow \mathbb{R}$ for simulated square-shaped $r(\mathbf{x}) = ||\mathbf{x}||_\infty^2$ pattern



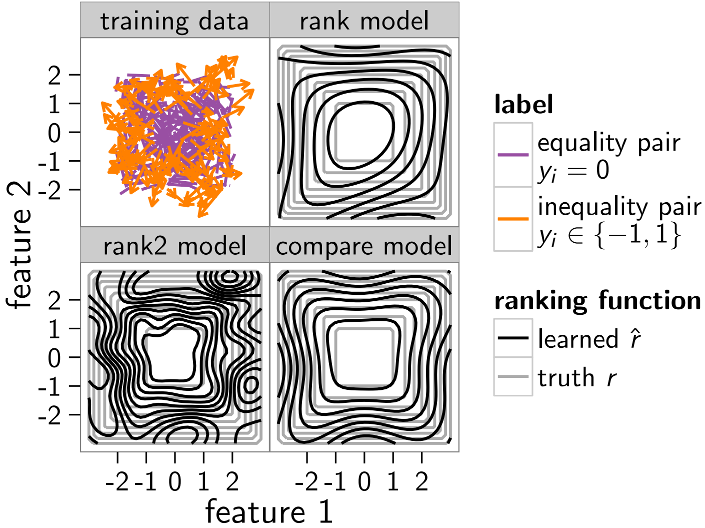
# 50% equality pairs (400 total) and learned $\hat{r} : \mathbb{R}^2 \rightarrow \mathbb{R}$ for simulated square-shaped $r(\mathbf{x}) = ||\mathbf{x}||_\infty^2$ pattern



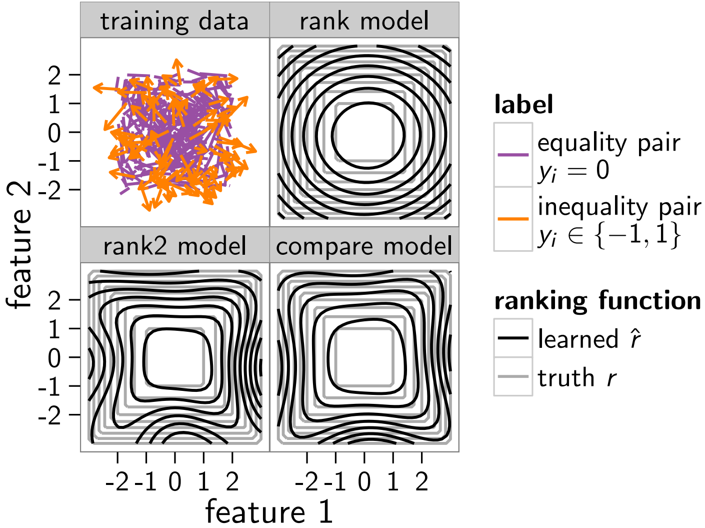
# 60% equality pairs (400 total) and learned $\hat{r} : \mathbb{R}^2 \rightarrow \mathbb{R}$ for simulated square-shaped $r(\mathbf{x}) = ||\mathbf{x}||_\infty^2$ pattern



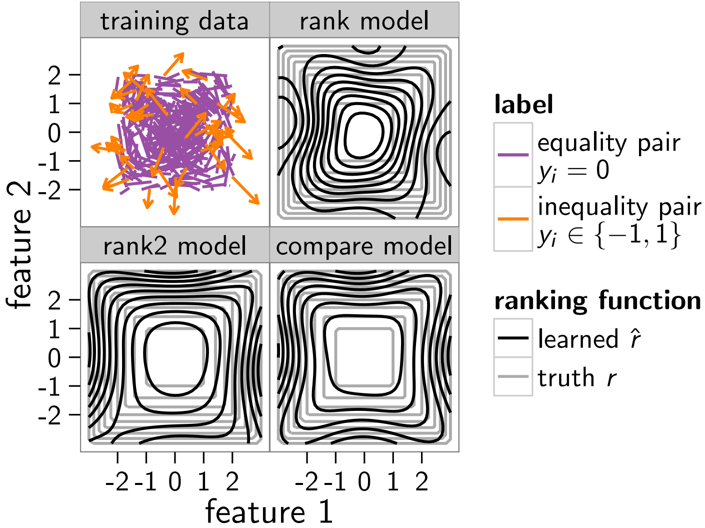
# 70% equality pairs (400 total) and learned $\hat{r} : \mathbb{R}^2 \rightarrow \mathbb{R}$ for simulated square-shaped $r(\mathbf{x}) = ||\mathbf{x}||_\infty^2$ pattern



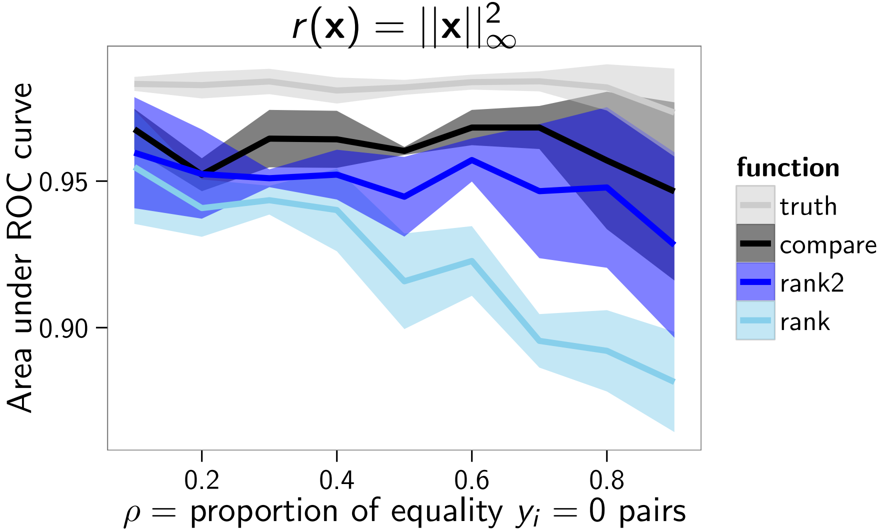
# 80% equality pairs (400 total) and learned $\hat{r} : \mathbb{R}^2 \rightarrow \mathbb{R}$ for simulated square-shaped $r(\mathbf{x}) = ||\mathbf{x}||_\infty^2$ pattern



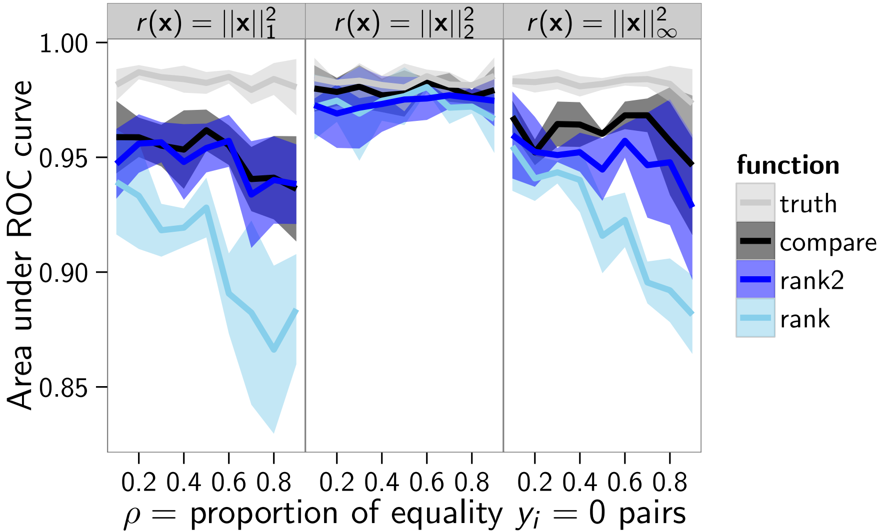
## 90% equality pairs (400 total) and learned $\hat{r} : \mathbb{R}^2 \rightarrow \mathbb{R}$ for simulated square-shaped $r(\mathbf{x}) = ||\mathbf{x}||_\infty^2$ pattern



## No difference for few equality pairs, rank worse when there are many equality pairs



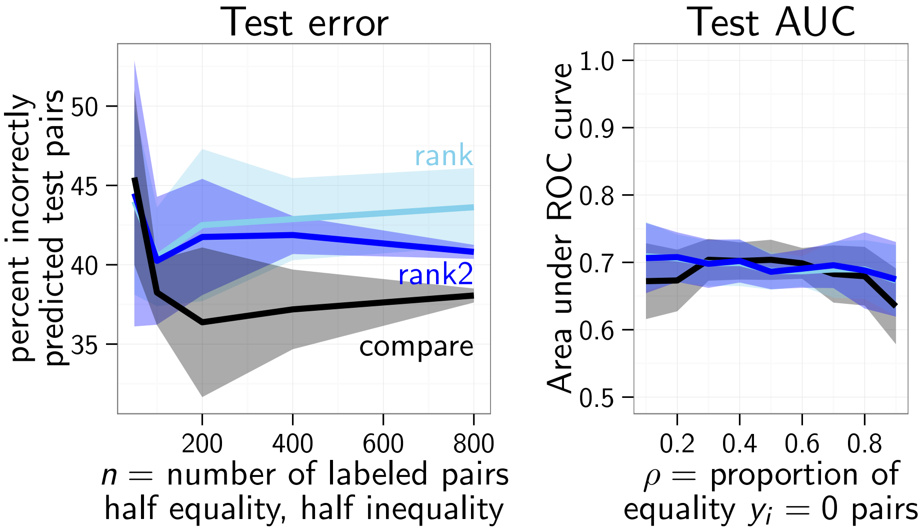
# No difference for few equality pairs, rank worse when there are many equality pairs



# Sushi data of Kamishima et al.

- `http://www.kamishima.net/sushi/`
- 100 different sushis rated by 5000 different people.
- Each person rated 10 sushis on a 5 point scale.
- Convert 10 ratings to 5 preference pairs.
- 17,832 equality $y_i = 0$ pairs and
- 7,168 inequality $y_i \in \{-1, 1\}$ pairs.
- Feature pairs $\mathbf{x}_i, \mathbf{x}'_i \in \mathbb{R}^{14}$.
- 7 sushi features: style, major, minor, oily, eating frequency, price, and selling frequency.
- 7 taster/person features: Sushi gender, age, time, birthplace and current home (we converted Japanese prefecture codes to latitude/longitude coordinates).

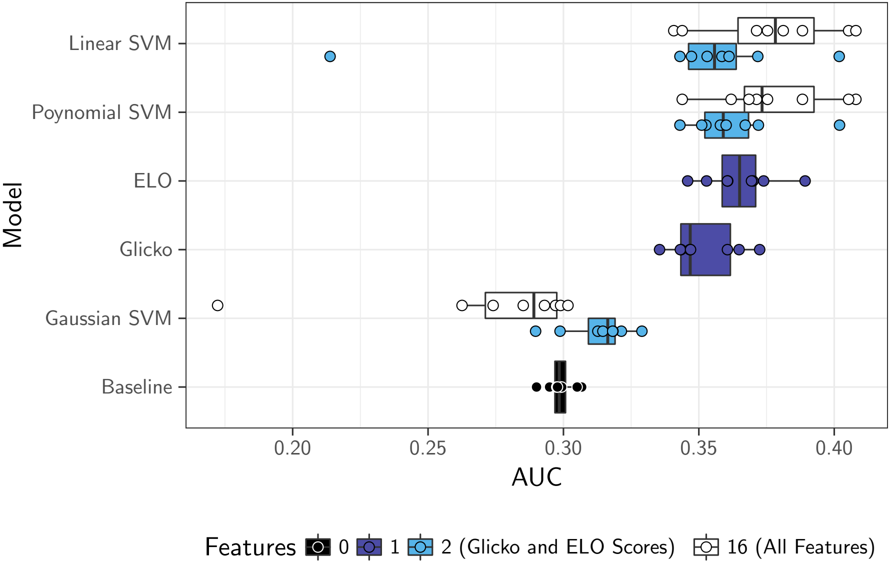
# Sushi data are harder, but SVMcompare still has lowest test error



# Chess data description

- http://www.chessmetrics.com, 1999–2006 (eight years).
- For each year, train on first four months (Jan–Apr), test on other months (May–Dec).
- 44.7% draws ($y_i = 0$) – predicting them is important!
- 16 features computed for each player and game: ELO score, Glicko score, initial move, loss/wins to a lower/higher ranked player, the average score difference of opponents, win/loss/draw/games played raw values and percentages, etc.

# Chess data result



# Conclusions and future work

- Learned a nonlinear ranking function $r(\mathbf{x}) \in \mathbb{R}$, and
- a comparison function $c(\mathbf{x}, \mathbf{x}') \in \{-1, 0, 1\}$.
- Results in simulation/sushi: rank $<$ rank2 $<$ compare.
- Results in chess: linear SVM improves over ELO/Glicko.
- Directly learning from $y_i = 0$ equality pairs (draws) is important,
  when they are present!
- `https://github.com/tdhock/rankSVMcompare`
- Future work: algorithms for large data and online setting.

# Thank you!

Supplementary slides appear after this one.