Support Vector Machines as Probabilistic Models

Vojtěch Franc [1], Alex Zien [2], Bernhard Schölkopf [3]

[1] Czech Technical University in Prague
[2] LIFE Biosystems GmbH, Germany
[3] Max Planck Institute for Intelligent Systems, Tübingen, Germany

November 13, 2013

The 32th Pattern Recognition and Computer Vision Colloquium
In Prague, April 23, 2013

Introduction

Task: Given training examples $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\} \in (\mathbb{R}^n \times \{+1, -1\})$ i.i.d. from unknown $p^*(\boldsymbol{x}, y)$, the goal is to learn Bayes classifier $q\colon \mathbb{R}^n \to \{+1, -1\}$ which minimizes the expected classification error $E_{p^*}[\![y \neq q(\boldsymbol{x})]\!]$.

Introduction

Task: Given training examples $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\} \in (\mathbb{R}^n \times \{+1, -1\})$ i.i.d. from unknown $p^*(\boldsymbol{x}, y)$, the goal is to learn Bayes classifier $q\colon \mathbb{R}^n \to \{+1, -1\}$ which minimizes the expected classification error $E_{p^*}[\![y \neq q(\boldsymbol{x})]\!]$.

Maximum Likelihood learning (Generative paradigm)

- Assume parametric distribution $p(\boldsymbol{x}, y; \theta)$ well approximates $p^*(\boldsymbol{x}, y)$.
- Construct the plug-in Bayes classifier $\hat{y} \in \operatorname{argmax}_{y \in \{+1,-1\}} p(\boldsymbol{x}, y; \theta_{\mathrm{ML}})$ where θ_{ML} is the ML estimate of θ.

Introduction

Task: Given training examples $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\} \in (\mathbb{R}^n \times \{+1, -1\})$ i.i.d. from unknown $p^*(\boldsymbol{x}, y)$, the goal is to learn Bayes classifier $q\colon \mathbb{R}^n \to \{+1, -1\}$ which minimizes the expected classification error $E_{p^*}[\![y \neq q(\boldsymbol{x})]\!]$.

Maximum Likelihood learning (Generative paradigm)

- Assume parametric distribution $p(\boldsymbol{x}, y; \theta)$ well approximates $p^*(\boldsymbol{x}, y)$.
- Construct the plug-in Bayes classifier $\hat{y} \in \operatorname{argmax}_{y \in \{+1,-1\}} p(\boldsymbol{x}, y; \theta_{\mathrm{ML}})$ where θ_{ML} is the ML estimate of θ.

Support Vector Machines learning (Discriminative paradigm)

- Assume a linear classifier well approximates the Bayes classifier.
- Find the linear classifier which minimizes (an approximation of) the empirical risk.

Introduction

Task: Given training examples $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\} \in (\mathbb{R}^n \times \{+1, -1\})$ i.i.d. from unknown $p^*(\boldsymbol{x}, y)$, the goal is to learn Bayes classifier $q\colon \mathbb{R}^n \to \{+1, -1\}$ which minimizes the expected classification error $E_{p^*}[\![y \neq q(\boldsymbol{x})]\!]$.

Maximum Likelihood learning (Generative paradigm)

- Assume parametric distribution $p(\boldsymbol{x}, y; \theta)$ well approximates $p^*(\boldsymbol{x}, y)$.
- Construct the plug-in Bayes classifier $\hat{y} \in \operatorname{argmax}_{y \in \{+1,-1\}} p(\boldsymbol{x}, y; \theta_{\mathrm{ML}})$ where θ_{ML} is the ML estimate of θ.

Support Vector Machines learning (Discriminative paradigm)

- Assume a linear classifier well approximates the Bayes classifier.
- Find the linear classifier which minimizes (an approximation of) the empirical risk.

Our contribution to the problem:

1. We construct a probabilistic model under which the ML learning and the SVM learning become equivalent.
2. We illustrate the benefits of this new view by giving a probabilistic interpretation to established SVM-related heuristics.

Linear Support Vector Machine classifier without bias

- For technical reasons, we consider the linear SVM without bias.
- We seek a linear classifier without bias (=separating hyperplane passing via the origin)

$$q_{\mathrm{SVM}}(\boldsymbol{x};\boldsymbol{w}) = \begin{cases} +1 & \text{if} \quad \langle \boldsymbol{x}, \boldsymbol{w} \rangle \geq 0\,, \\ -1 & \text{if} \quad \langle \boldsymbol{x}, \boldsymbol{w} \rangle < 0\,. \end{cases}$$

Linear Support Vector Machine classifier without bias

- For technical reasons, we consider the linear SVM without bias.
- We seek a linear classifier without bias (=separating hyperplane passing via the origin)

$$q_{\mathrm{SVM}}(\boldsymbol{x};\boldsymbol{w}) = \begin{cases} +1 & \text{if} \quad \langle \boldsymbol{x}, \boldsymbol{w} \rangle \geq 0\,, \\ -1 & \text{if} \quad \langle \boldsymbol{x}, \boldsymbol{w} \rangle < 0\,. \end{cases}$$

- For fixed hyper-parameters $\lambda \in \mathbb{R}^{++}$ and $\omega \in (0,1)$ the parameter vector $\boldsymbol{w}$ is computed by solving

$$\boldsymbol{w}_{\mathrm{SVM}}(\lambda,\omega) = \operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^n} \left[\frac{\lambda}{2}\|\boldsymbol{w}\|^2 + R_{SVM}(\boldsymbol{w};\omega) \right]$$

where

$$R_{SVM}(\boldsymbol{w};\omega) = \sum_{i=1}^{m} \omega^{y_i}\, \ell(y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)$$

is the convex risk given by the hinge loss $\ell(t) = \max\{0, 1-t\}$ and cost-factors $\omega^+ = \omega$ and $\omega^- = 1-\omega$ for the positive and negative class where $\omega \in (0,1)$.

Proposed semi-paramateric probabilistic model

$$p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u}) = \underbrace{Z(\tau, \omega)}_{\text{normalization constant}} \cdot \underbrace{\exp\Big(- \omega^y \ell\big(y\langle \boldsymbol{x}, \tau \boldsymbol{u}\rangle\big)\Big)}_{\text{exponent of hinge loss}} \cdot \underbrace{h(\boldsymbol{x})}_{\text{non-parametric term}}$$

Proposed semi-paramateric probabilistic model

$$p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u}) = \underbrace{Z(\tau, \omega)}_{\text{normalization constant}} \cdot \underbrace{\exp\Big(-\omega^y \ell\big(y\langle \boldsymbol{x}, \tau \boldsymbol{u}\rangle\big)\Big)}_{\text{exponent of hinge loss}} \cdot \underbrace{h(\boldsymbol{x})}_{\text{non-parametric term}}$$

The parameters of the distribution:

1. strictly positive scalar $\tau \in \mathbb{R}^{++}$
2. scalar $\omega \in (0, 1)$ defining $\omega^+ = \omega$ and $\omega^- = 1 - \omega$
3. unit vector $\boldsymbol{u} \in \mathcal{U} = \{\boldsymbol{u}' \in \mathbb{R}^n \mid \|\boldsymbol{u}'\| = 1\}$

Related SVM parameters

regularization const. $\lambda \in \mathbb{R}^{++}$

cost-factor $\omega \in (0, 1)$

parameters $\boldsymbol{w} \in \mathbb{R}^n$

Proposed semi-paramateric probabilistic model

$$p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u}) = \underbrace{Z(\tau, \omega)}_{\text{normalization constant}} \cdot \underbrace{\exp\Big(-\omega^y \ell\big(y\langle \boldsymbol{x}, \tau \boldsymbol{u}\rangle\big)\Big)}_{\text{exponent of hinge loss}} \cdot \underbrace{h(\boldsymbol{x})}_{\text{non-parametric term}}$$

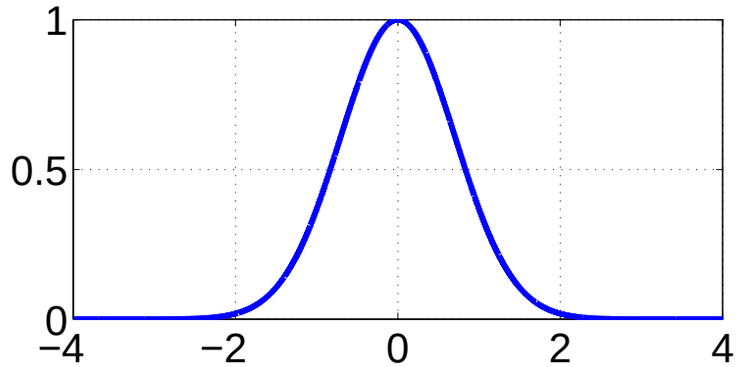
The parameters of the distribution:

1. strictly positive scalar $\tau \in \mathbb{R}^{++}$
2. scalar $\omega \in (0, 1)$ defining $\omega^+ = \omega$ and $\omega^- = 1 - \omega$
3. unit vector $\boldsymbol{u} \in \mathcal{U} = \{\boldsymbol{u}' \in \mathbb{R}^n \mid \|\boldsymbol{u}'\| = 1\}$

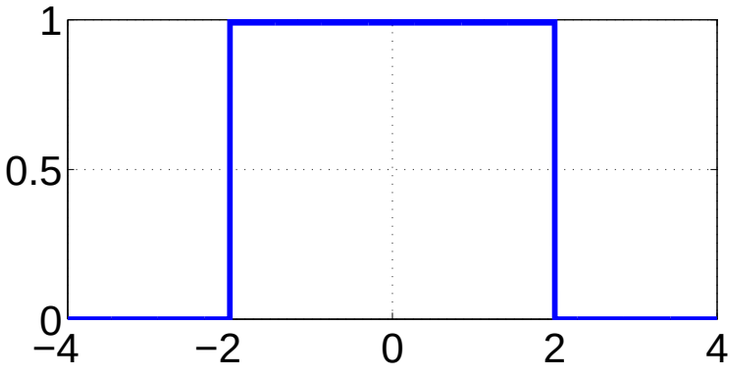
Related SVM parameters

regularization const. $\lambda \in \mathbb{R}^{++}$
cost-factor $\omega \in (0, 1)$
parameters $\boldsymbol{w} \in \mathbb{R}^n$

The non-parametric term $h\colon \mathbb{R}^n \to \mathbb{R}$ is an arbitrary positive and integrable radial basis function ($h(\boldsymbol{x}) = h(\|\boldsymbol{x}\|)$). For example:



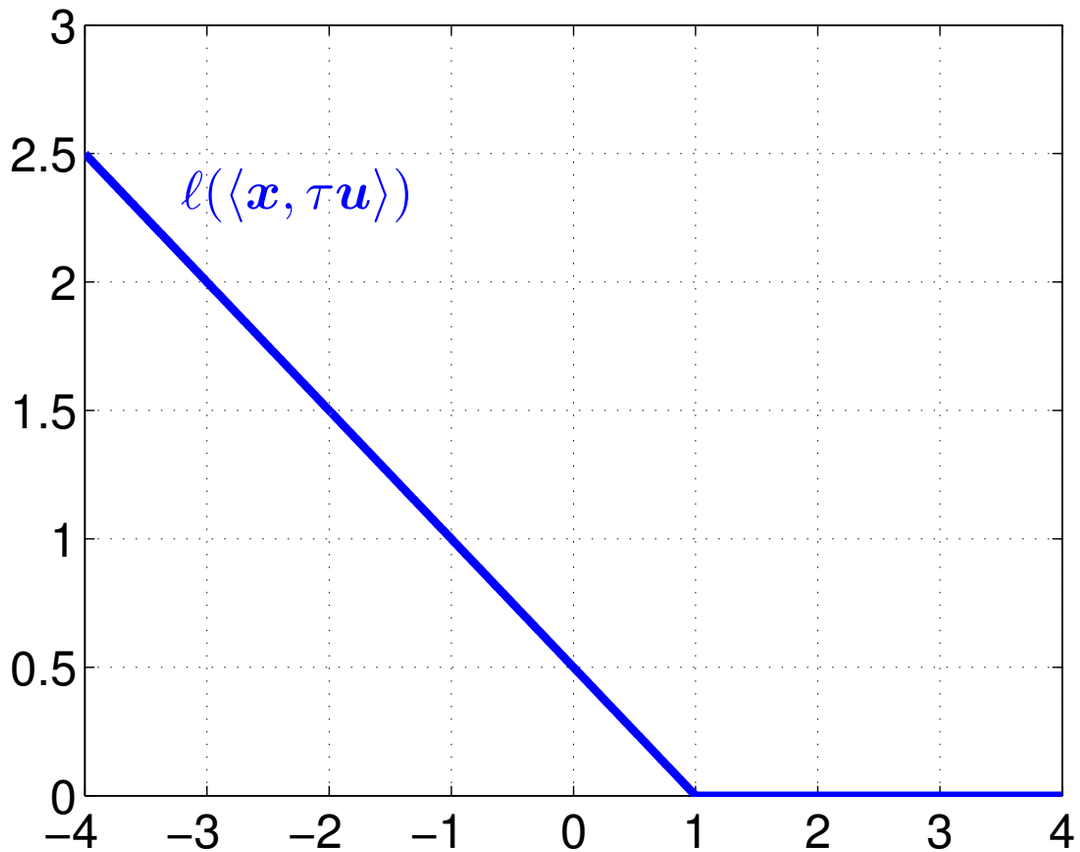
$h_1(\boldsymbol{x}) = \exp(-\langle \boldsymbol{x}, c_1 \mathbf{E}\boldsymbol{x}\rangle)$



$h_2(\boldsymbol{x}) = c_2[\![\|\boldsymbol{x}\| \leq c_3]\!]$

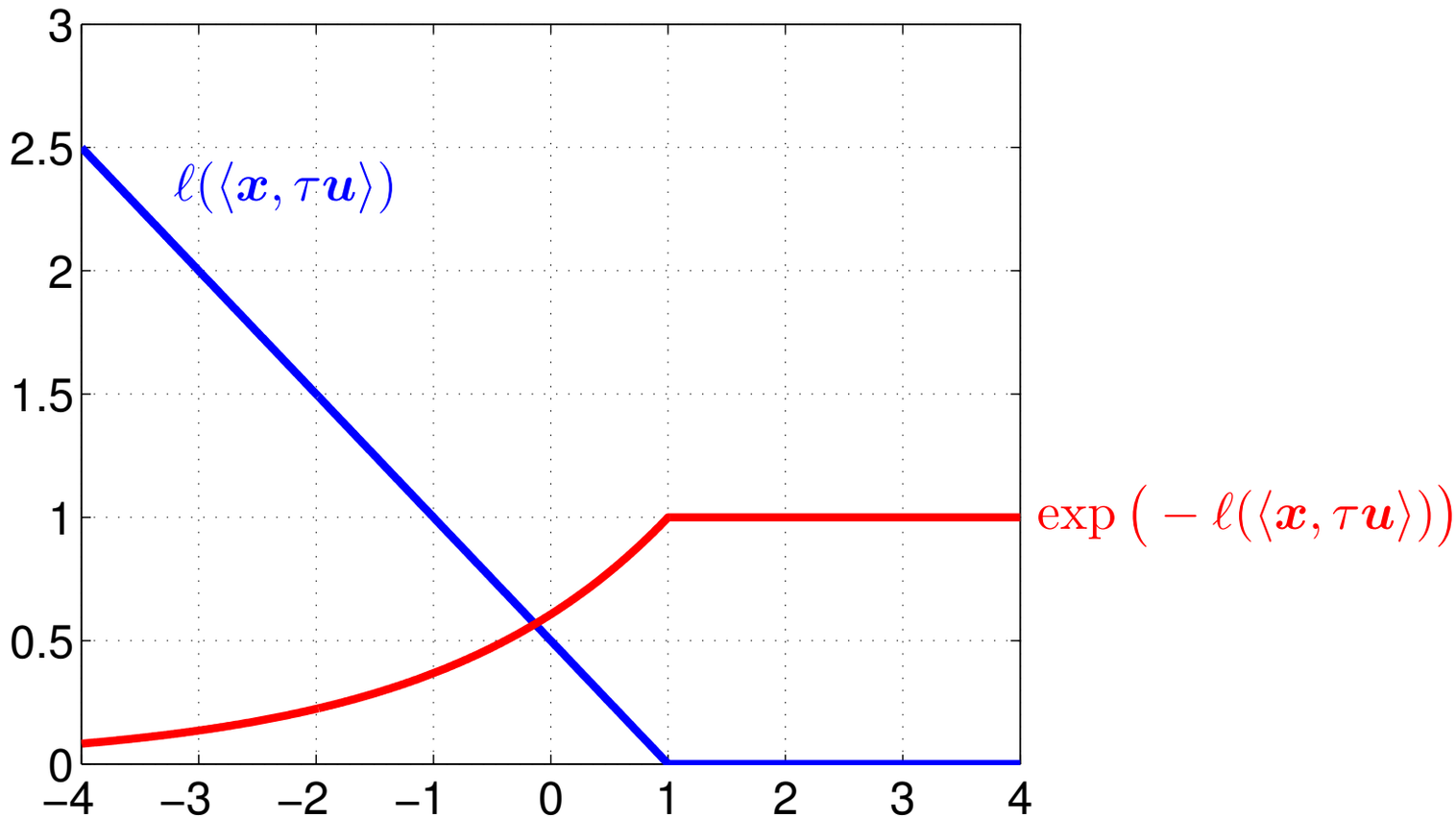
Proposed semi-paramateric probabilistic model

$$p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u}) = \underbrace{Z(\tau, \omega)}_{\text{normalization constant}} \cdot \underbrace{\exp\Big(- \omega^y \ell\big(y\langle \boldsymbol{x}, \tau \boldsymbol{u}\rangle\big)\Big)}_{\text{exponent of hinge loss}} \cdot \underbrace{h(\boldsymbol{x})}_{\text{non-parametric term}}$$



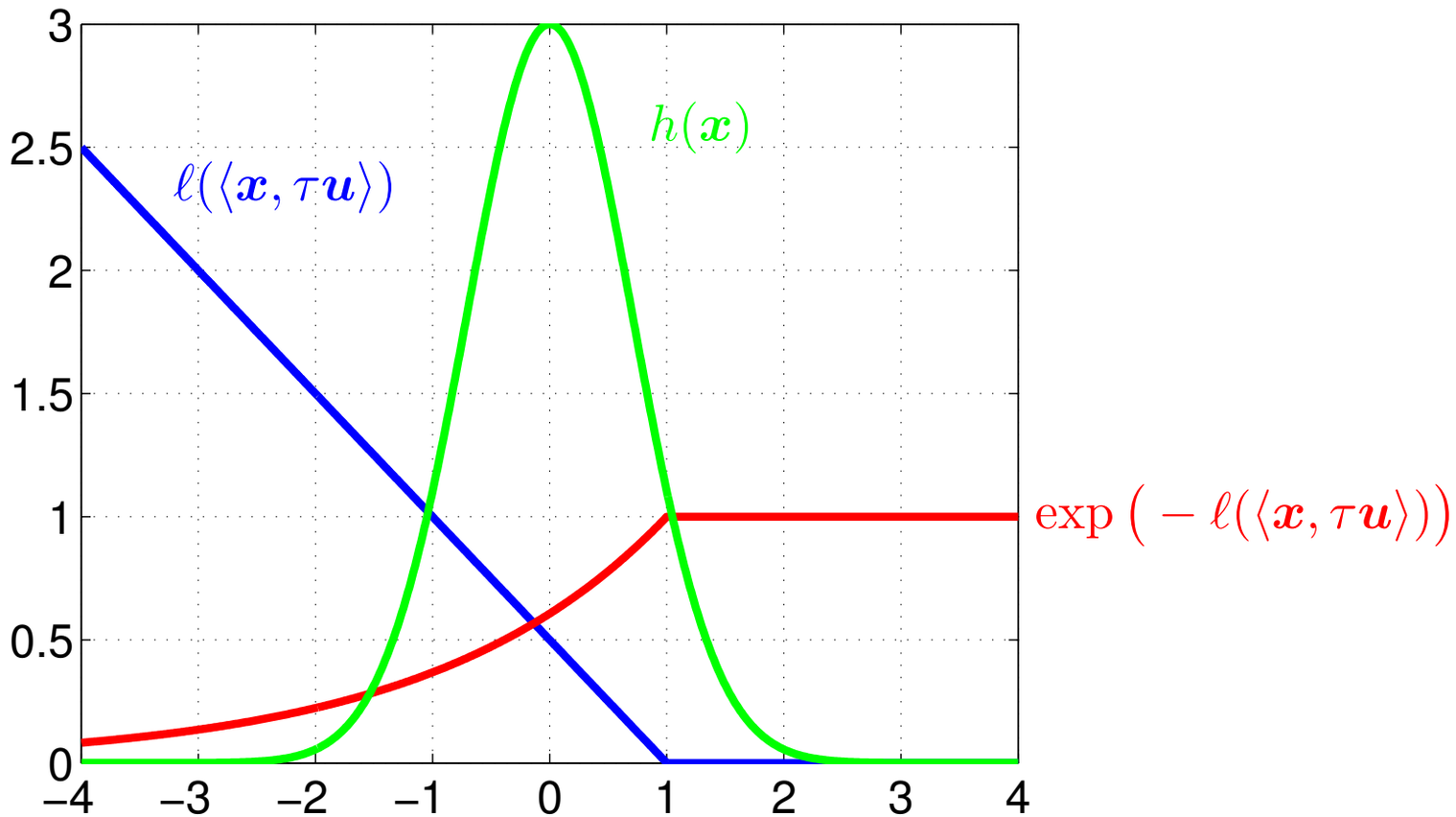
Proposed semi-paramateric probabilistic model

$$p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u}) = \underbrace{Z(\tau, \omega)}_{\text{normalization constant}} \cdot \underbrace{\exp\Big(-\omega^y \ell\big(y\langle \boldsymbol{x}, \tau \boldsymbol{u}\rangle\big)\Big)}_{\text{exponent of hinge loss}} \cdot \underbrace{h(\boldsymbol{x})}_{\text{non-parametric term}}$$



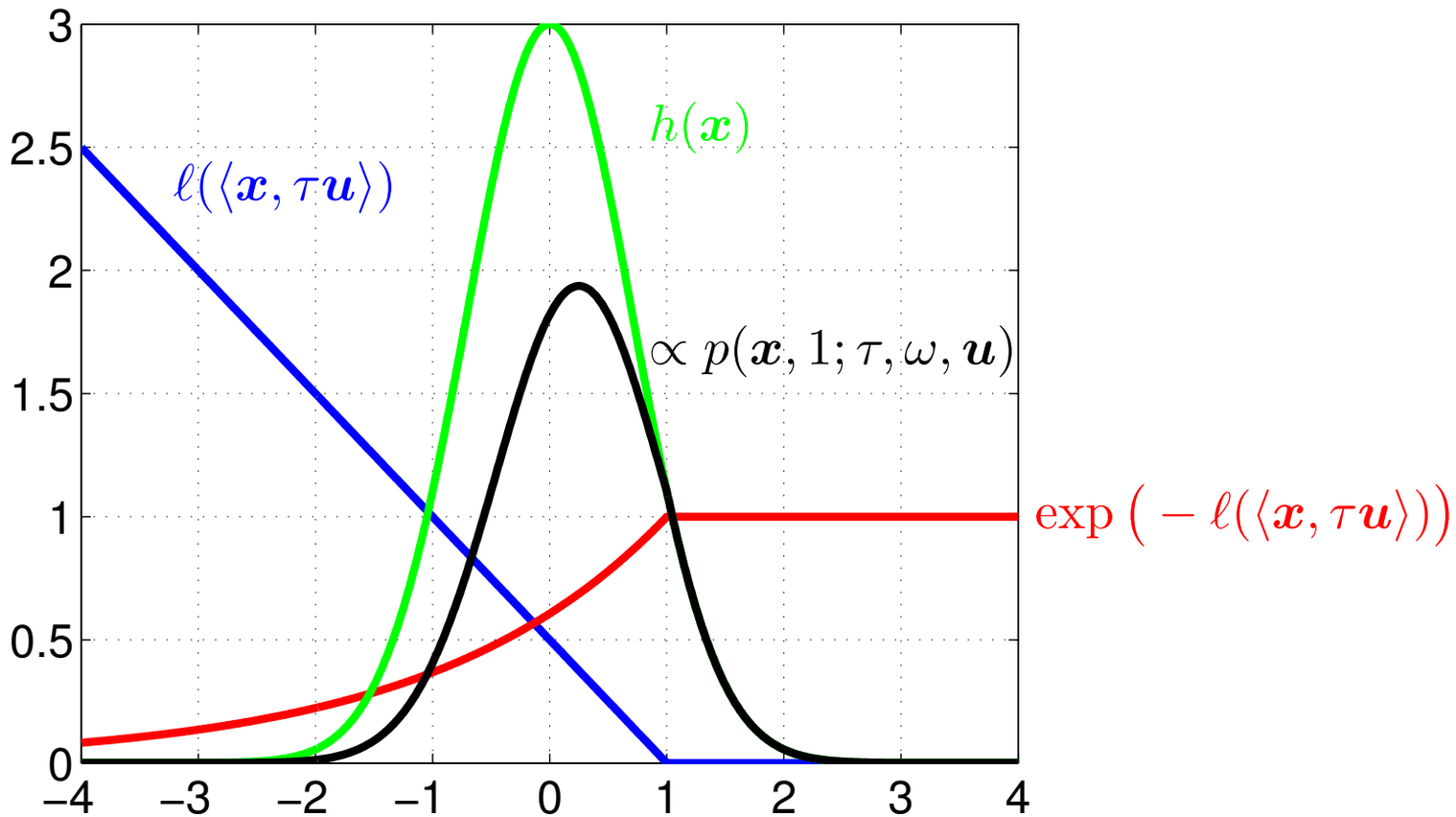
Proposed semi-paramateric probabilistic model

$$p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u}) = \underbrace{Z(\tau, \omega)}_{\text{normalization constant}} \cdot \underbrace{\exp\Big(-\omega^y \ell\big(y\langle \boldsymbol{x}, \tau \boldsymbol{u}\rangle\big)\Big)}_{\text{exponent of hinge loss}} \cdot \underbrace{h(\boldsymbol{x})}_{\text{non-parametric term}}$$



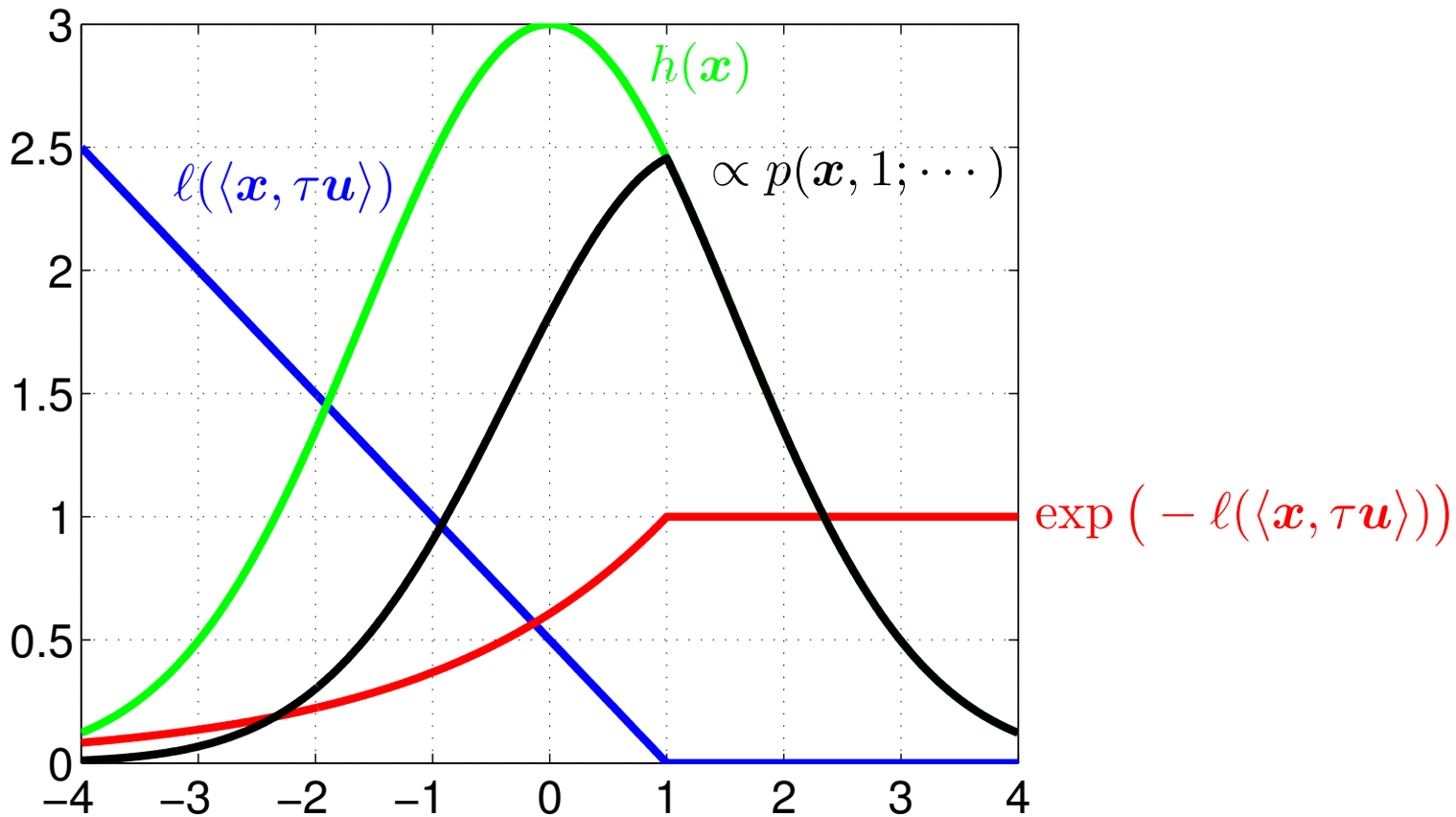
Proposed semi-paramateric probabilistic model

$$p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u}) = \underbrace{Z(\tau, \omega)}_{\text{normalization constant}} \cdot \underbrace{\exp\Big(-\omega^y \ell\big(y\langle \boldsymbol{x}, \tau \boldsymbol{u}\rangle\big)\Big)}_{\text{exponent of hinge loss}} \cdot \underbrace{h(\boldsymbol{x})}_{\text{non-parametric term}}$$



Proposed semi-paramateric probabilistic model

$$p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u}) = \underbrace{Z(\tau, \omega)}_{\text{normalization constant}} \cdot \underbrace{\exp\Big(-\omega^y \ell\big(y\langle \boldsymbol{x}, \tau \boldsymbol{u}\rangle\big)\Big)}_{\text{exponent of hinge loss}} \cdot \underbrace{h(\boldsymbol{x})}_{\text{non-parametric term}}$$



Marginals, posteriors and comparison to Logistic Regression

Marginal distribution

$$p(\boldsymbol{x};\tau,\omega,\boldsymbol{u}) = Z(\tau,\omega)\cdot h(\boldsymbol{x})\cdot f(\boldsymbol{x};\tau,\omega,\boldsymbol{u})$$

with the parametric term (RED)

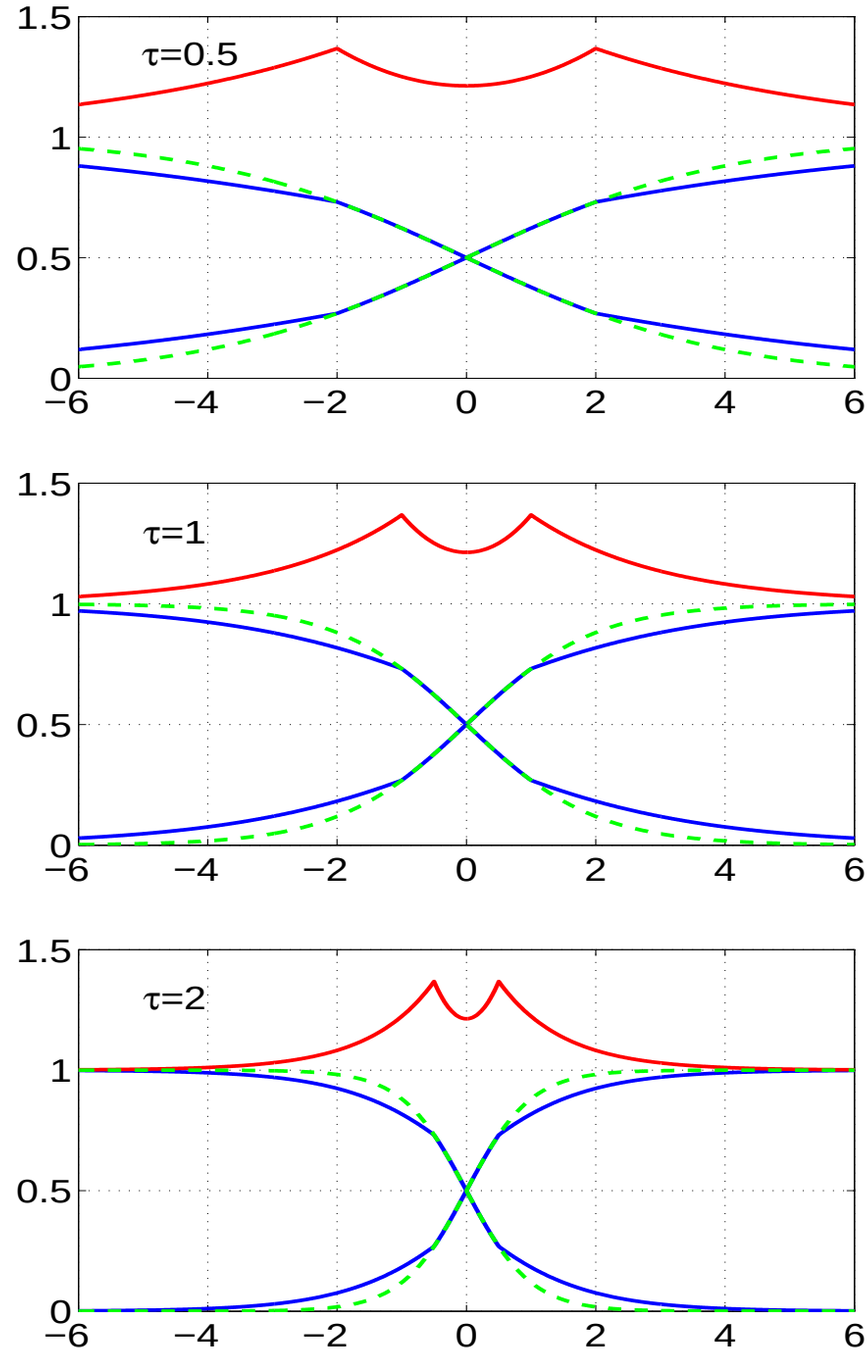
$$f(\boldsymbol{x};\tau,\omega,\boldsymbol{u}) = \sum_{y\in\{1,-1\}} \exp(-\omega^y \ell(y\langle\tau\boldsymbol{u},\boldsymbol{x}\rangle))$$

Posterior probability (BLUE)

$$p(y \mid \boldsymbol{x};\tau,\omega,\boldsymbol{u}) = \frac{\exp(-\omega^y \ell(y\langle\tau\boldsymbol{u},\boldsymbol{x}\rangle))}{f(\boldsymbol{x};\tau,\omega,\boldsymbol{u})}$$

Logistic Regression model (GREEN)

$$p_{\mathrm{LR}}(y \mid \boldsymbol{x};\tau\boldsymbol{u}) = \frac{1}{1+\exp(-y\langle\tau\boldsymbol{u},\boldsymbol{x}\rangle)}$$



ML estimation compared to SVM learning

Given examples $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$ the negative log-likelihood reads

$$L(\tau, \omega, \boldsymbol{u}) = -\sum_{i=1}^{m} \log p(\boldsymbol{x}_i, y_i; \tau, \omega, \boldsymbol{u}) = R_{SVM}(\tau \boldsymbol{u}; \omega) - m \log Z(\tau, \omega) - \sum_{i=1}^{m} \log h(\boldsymbol{x}_i)$$

ML estimation compared to SVM learning

Given examples $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$ the negative log-likelihood reads

$$L(\tau, \omega, \boldsymbol{u}) = -\sum_{i=1}^{m} \log p(\boldsymbol{x}_i, y_i; \tau, \omega, \boldsymbol{u}) = R_{SVM}(\tau \boldsymbol{u}; \omega) - m \log Z(\tau, \omega) - \sum_{i=1}^{m} \log h(\boldsymbol{x}_i)$$

ML estimation: given hyper-parameters τ and ω we compute

$$\boldsymbol{u}_{\mathrm{ML}}(\tau, \omega) \in \operatorname*{argmin}_{\boldsymbol{u} \in \mathcal{U}} L(\tau, \omega, \boldsymbol{u}) = \operatorname*{argmin}_{\boldsymbol{u} \in \mathcal{U}} R_{SVM}(\tau \boldsymbol{u}; \omega)$$

ML estimation compared to SVM learning

Given examples $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$ the negative log-likelihood reads

$$L(\tau, \omega, \boldsymbol{u}) = -\sum_{i=1}^{m} \log p(\boldsymbol{x}_i, y_i; \tau, \omega, \boldsymbol{u}) = R_{SVM}(\tau \boldsymbol{u}; \omega) - m \log Z(\tau, \omega) - \sum_{i=1}^{m} \log h(\boldsymbol{x}_i)$$

ML estimation: given hyper-parameters τ and ω we compute

$$\boldsymbol{u}_{\mathrm{ML}}(\tau, \omega) \in \operatorname*{argmin}_{\boldsymbol{u} \in \mathcal{U}} L(\tau, \omega, \boldsymbol{u}) = \operatorname*{argmin}_{\boldsymbol{u} \in \mathcal{U}} R_{SVM}(\tau \boldsymbol{u}; \omega)$$

SVM learning: given hyper-parameters λ and ω we compute

$$\boldsymbol{w}_{SVM}(\lambda, \omega) = \operatorname*{argmin}_{\boldsymbol{w} \in \mathbb{R}^n} \left[\frac{\lambda}{2} \|\boldsymbol{w}\|^2 + R_{SVM}(\boldsymbol{w}; \omega) \right]$$

ML estimate and SVM learning give the same parameters

Theorem: For any $\omega \in (0,1)$, $\lambda \in \mathbb{R}^{++}$ and $\tau = \|\boldsymbol{w}_{SVM}(\lambda,\omega)\|$ it holds

$$\tau \cdot \boldsymbol{u}_{ML}(\tau,\omega) = \boldsymbol{w}_{SVM}(\lambda,\omega)$$

ML estimate and SVM learning give the same parameters

Theorem: For any $\omega \in (0,1)$, $\lambda \in \mathbb{R}^{++}$ and $\tau = \|\boldsymbol{w}_{SVM}(\lambda,\omega)\|$ it holds

$$\tau \cdot \boldsymbol{u}_{ML}(\tau,\omega) = \boldsymbol{w}_{SVM}(\lambda,\omega)$$

ML estimate can be seen as re-parametrization of the SVM learning:

	hyper-parameter 1		hyper-parameter 2	
SVM learning	λ	regularization constant	ω	cost-factor of examples
ML estimate	τ	reciprocal to margin	ω	proportional to class prior

Plug-in Bayes classifier = SVM classifier

Bayes classifier minimizing the expected classification error is based on comparing the posterior probabilities $p(y = +1 \mid \boldsymbol{x}) \gtrless p(y = -1 \mid \boldsymbol{x})$.

Plug-in Bayes classifier derived from the model $p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u})$ reads

$$q_{Bayes}(\boldsymbol{x}; \tau, \omega, \boldsymbol{u}) = \begin{cases} +1 & \text{if} \quad \langle \tau \boldsymbol{u}, \boldsymbol{x} \rangle \geq b \\ -1 & \text{if} \quad \langle \tau \boldsymbol{u}, \boldsymbol{x} \rangle < b \end{cases} \qquad \text{where} \qquad b = 2\omega - 1$$

Plug-in Bayes classifier = SVM classifier

Bayes classifier minimizing the expected classification error is based on comparing the posterior probabilities $p(y = +1 \mid \boldsymbol{x}) \gtrless p(y = -1 \mid \boldsymbol{x})$.

Plug-in Bayes classifier derived from the model $p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u})$ reads

$$q_{Bayes}(\boldsymbol{x}; \tau, \omega, \boldsymbol{u}) = \begin{cases} +1 & \text{if} \quad \langle \tau \boldsymbol{u}, \boldsymbol{x} \rangle \geq b \\ -1 & \text{if} \quad \langle \tau \boldsymbol{u}, \boldsymbol{x} \rangle < b \end{cases} \qquad \text{where} \qquad b = 2\omega - 1$$

SVM classifier reads

$$q_{SVM}(\boldsymbol{x}; \boldsymbol{w}) = \begin{cases} +1 & \text{if} \quad \langle \boldsymbol{w}, \boldsymbol{x} \rangle \geq 0 \\ -1 & \text{if} \quad \langle \boldsymbol{w}, \boldsymbol{x} \rangle < 0 \end{cases}$$

Plug-in Bayes classifier = SVM classifier

Bayes classifier minimizing the expected classification error is based on comparing the posterior probabilities $p(y=+1 \mid \boldsymbol{x}) \gtreqless p(y=-1 \mid \boldsymbol{x})$.

Plug-in Bayes classifier derived from the model $p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u})$ reads

$$q_{Bayes}(\boldsymbol{x}; \tau, \omega, \boldsymbol{u}) = \begin{cases} +1 & \text{if} \quad \langle \tau \boldsymbol{u}, \boldsymbol{x} \rangle \geq b \\ -1 & \text{if} \quad \langle \tau \boldsymbol{u}, \boldsymbol{x} \rangle < b \end{cases} \qquad \text{where} \qquad b = 2\omega - 1$$

SVM classifier reads

$$q_{SVM}(\boldsymbol{x}; \boldsymbol{w}) = \begin{cases} +1 & \text{if} \quad \langle \boldsymbol{w}, \boldsymbol{x} \rangle \geq 0 \\ -1 & \text{if} \quad \langle \boldsymbol{w}, \boldsymbol{x} \rangle < 0 \end{cases}$$

Theorem: Let $\omega = 0.5$. Then, for any linear SVM classifier there exists an equivalent plug-in Bayes classifier whose parameters are estimated by the ML principle, i.e., the equality

$$q_{\mathrm{SVM}}(\boldsymbol{x}; \boldsymbol{w}_{\mathrm{SVM}}(\lambda, \omega)) = q_{\mathrm{Bayes}}(\boldsymbol{x}; \tau, \omega, \boldsymbol{u}_{\mathrm{ML}}(\tau, \omega))$$

holds for any $\boldsymbol{x} \in \mathbb{R}^n$, $\lambda \in \mathbb{R}^{++}$ and $\tau = \|\boldsymbol{w}_{\mathrm{SVM}}(\lambda, \omega)\|$.

Application 1: SVM classifier with different cost factors

In the case of unbalanced training data the following heuristics are common:

1. Set a higher cost-factor for the class which is less represented in the data. E.g., if the first class is the smaller one, then set $\omega > 0.5$, that is, $\omega^+ > \omega^-$.
2. After learning the vector $\boldsymbol{w}$ tune only the bias b of the linear classifier.

Application 1: SVM classifier with different cost factors

In the case of unbalanced training data the following heuristics are common:

1. Set a higher cost-factor for the class which is less represented in the data. E.g., if the first class is the smaller one, then set $\omega > 0.5$, that is, $\omega^+ > \omega^-$.
2. After learning the vector $\boldsymbol{w}$ tune only the bias b of the linear classifier.

Interpretation of the heuristics using the proposed probabilistic model:

1. **Theorem:** For any $\tau \in \mathbb{R}^{++}$, the prior probability derived from $p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u})$ is

$$\begin{array}{rcll} p(y=+1;\tau,\omega) & = & 0.5 & \text{if} \quad \omega = 0.5 \\ p(y=+1;\tau,\omega) & < & 0.5 & \text{if} \quad \omega > 0.5 \\ p(y=+1;\tau,\omega) & > & 0.5 & \text{if} \quad \omega < 0.5 \end{array}$$

Application 1: SVM classifier with different cost factors

In the case of unbalanced training data the following heuristics are common:

1. Set a higher cost-factor for the class which is less represented in the data. E.g., if the first class is the smaller one, then set $\omega > 0.5$, that is, $\omega^+ > \omega^-$.
2. After learning the vector $\boldsymbol{w}$ tune only the bias b of the linear classifier.

Interpretation of the heuristics using the proposed probabilistic model:

1. **Theorem:** For any $\tau \in \mathbb{R}^{++}$, the prior probability derived from $p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u})$ is

$$\begin{array}{rcll} p(y=+1;\tau,\omega) & = & 0.5 & \text{if} \quad \omega = 0.5 \\ p(y=+1;\tau,\omega) & < & 0.5 & \text{if} \quad \omega > 0.5 \\ p(y=+1;\tau,\omega) & > & 0.5 & \text{if} \quad \omega < 0.5 \end{array}$$

2. The plug-in Bayes classifier under the model $p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u})$ leads to biased linear rule

$$q_{Bayes}(\boldsymbol{x}; \tau, \omega, \boldsymbol{u}) = \begin{cases} +1 & \text{if} \quad \langle \tau \boldsymbol{u}, \boldsymbol{x} \rangle \geq b \\ -1 & \text{if} \quad \langle \tau \boldsymbol{u}, \boldsymbol{x} \rangle < b \end{cases} \qquad b = 2\omega - 1 = \begin{cases} = 0 & \text{if} \quad \omega = 0.5 \\ < 0 & \text{if} \quad \omega > 0.5 \\ > 0 & \text{if} \quad \omega < 0.5 \end{cases}$$

Application 2: Probabilistic outputs

On top of recovering the SVM decision function, the probabilistic model also naturally provides **probabilistic outputs**.

Easy-to-compute posterior probability:

$$p(y \mid \boldsymbol{x}; \tau, \omega, \boldsymbol{u}) = \frac{\exp(-\omega^y \ell(y\langle \tau \boldsymbol{u}, \boldsymbol{x}\rangle))}{\sum_{y\in\{1,-1\}} \exp(-\omega^y \ell(y\langle \tau \boldsymbol{u}, \boldsymbol{x}\rangle))} \qquad (\text{or } \boldsymbol{w} = \tau \boldsymbol{u})$$

Application 2: Probabilistic outputs

On top of recovering the SVM decision function, the probabilistic model also naturally provides **probabilistic outputs**.

Easy-to-compute posterior probability:

$$p(y \mid \boldsymbol{x}; \tau, \omega, \boldsymbol{u}) = \frac{\exp(-\omega^y \ell(y\langle \tau \boldsymbol{u}, \boldsymbol{x}\rangle))}{\sum_{y\in\{1,-1\}} \exp(-\omega^y \ell(y\langle \tau \boldsymbol{u}, \boldsymbol{x}\rangle))} \qquad (\text{or } \boldsymbol{w} = \tau \boldsymbol{u})$$

This provides a convenient alternative to the commonly used Platt's approach which is based on retrospectively fitting the sigmoid function to the SVM outputs.

The shape of SVM posterior is very similar to the Logistic model (sigmoid) used by Plat.

Application 3: Maximum Margin Clustering = Classification Maximum Likelihood

Task: Let $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\} \in (\mathbb{R}^n \times \{+1, -1\})^m$ be generated by i.i.d. random variables distributed according to $p^*(\boldsymbol{x}, y)$. Given only the observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ the goal is to recover the hidden labels $\{y_1, \ldots, y_m\}$.

Application 3: Maximum Margin Clustering = Classification Maximum Likelihood

Task: Let $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\} \in (\mathbb{R}^n \times \{+1, -1\})^m$ be generated by i.i.d. random variables distributed according to $p^*(\boldsymbol{x}, y)$. Given only the observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ the goal is to recover the hidden labels $\{y_1, \ldots, y_m\}$.

Maximum Margin Clustering finds the labels by (approximately) solving

$$\boldsymbol{y}_{\mathrm{MMC}}(\lambda) = \operatorname*{Argmin}_{\boldsymbol{y} \in \{1,-1\}^m} \min_{\boldsymbol{w} \in \mathbb{R}^n} \left[\frac{\lambda}{2} \|\boldsymbol{w}\|^2 + \sum_{i=1}^{m} \ell(y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle) \right]$$

Application 3: Maximum Margin Clustering = Classification Maximum Likelihood

Task: Let $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\} \in (\mathbb{R}^n \times \{+1, -1\})^m$ be generated by i.i.d. random variables distributed according to $p^*(\boldsymbol{x}, y)$. Given only the observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ the goal is to recover the hidden labels $\{y_1, \ldots, y_m\}$.

Maximum Margin Clustering finds the labels by (approximately) solving

$$\boldsymbol{y}_{\mathrm{MMC}}(\lambda) = \operatorname*{Argmin}_{\boldsymbol{y} \in \{1,-1\}^m} \min_{\boldsymbol{w} \in \mathbb{R}^n} \left[\frac{\lambda}{2} \|\boldsymbol{w}\|^2 + \sum_{i=1}^{m} \ell(y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle) \right]$$

Classification Maximum Likelihood principle applied to the model $p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u})$ finds the labels by (approximately) solving

$$\boldsymbol{y}_{CML}(\tau) = \operatorname*{Argmin}_{\boldsymbol{y} \in \{1,-1\}^m} \min_{\boldsymbol{u} \in \mathcal{U}} \left[-\sum_{i=1}^{m} \log p(\boldsymbol{x}_i \mid y_i; \tau, \omega = 0.5, \boldsymbol{u}) \right]$$

Application 3: Maximum Margin Clustering = Classification Maximum Likelihood

Task: Let $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\} \in (\mathbb{R}^n \times \{+1, -1\})^m$ be generated by i.i.d. random variables distributed according to $p^*(\boldsymbol{x}, y)$. Given only the observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ the goal is to recover the hidden labels $\{y_1, \ldots, y_m\}$.

Maximum Margin Clustering finds the labels by (approximately) solving

$$\boldsymbol{y}_{\mathrm{MMC}}(\lambda) = \operatorname*{Argmin}_{\boldsymbol{y} \in \{1,-1\}^m} \min_{\boldsymbol{w} \in \mathbb{R}^n} \left[\frac{\lambda}{2} \|\boldsymbol{w}\|^2 + \sum_{i=1}^{m} \ell(y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle) \right]$$

Classification Maximum Likelihood principle applied to the model $p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u})$ finds the labels by (approximately) solving

$$\boldsymbol{y}_{CML}(\tau) = \operatorname*{Argmin}_{\boldsymbol{y} \in \{1,-1\}^m} \min_{\boldsymbol{u} \in \mathcal{U}} \left[-\sum_{i=1}^{m} \log p(\boldsymbol{x}_i \mid y_i; \tau, \omega = 0.5, \boldsymbol{u}) \right]$$

Theorem: Let $\boldsymbol{y}_{\mathrm{MMC}}(\lambda)$ be a set of minimizers of the MMC problem for some $\lambda \in \mathbb{R}^{++}$. Then, for any labeling $\boldsymbol{y}^* \in \boldsymbol{y}_{\mathrm{MMC}}(\lambda)$ there exists $\tau \in \mathbb{R}^+$ such that $\boldsymbol{y}^*$ is a minimizer of the CML problem, i.e., $\boldsymbol{y}^* \in \boldsymbol{y}_{\mathrm{CML}}(\tau)$ holds.

Conclusions

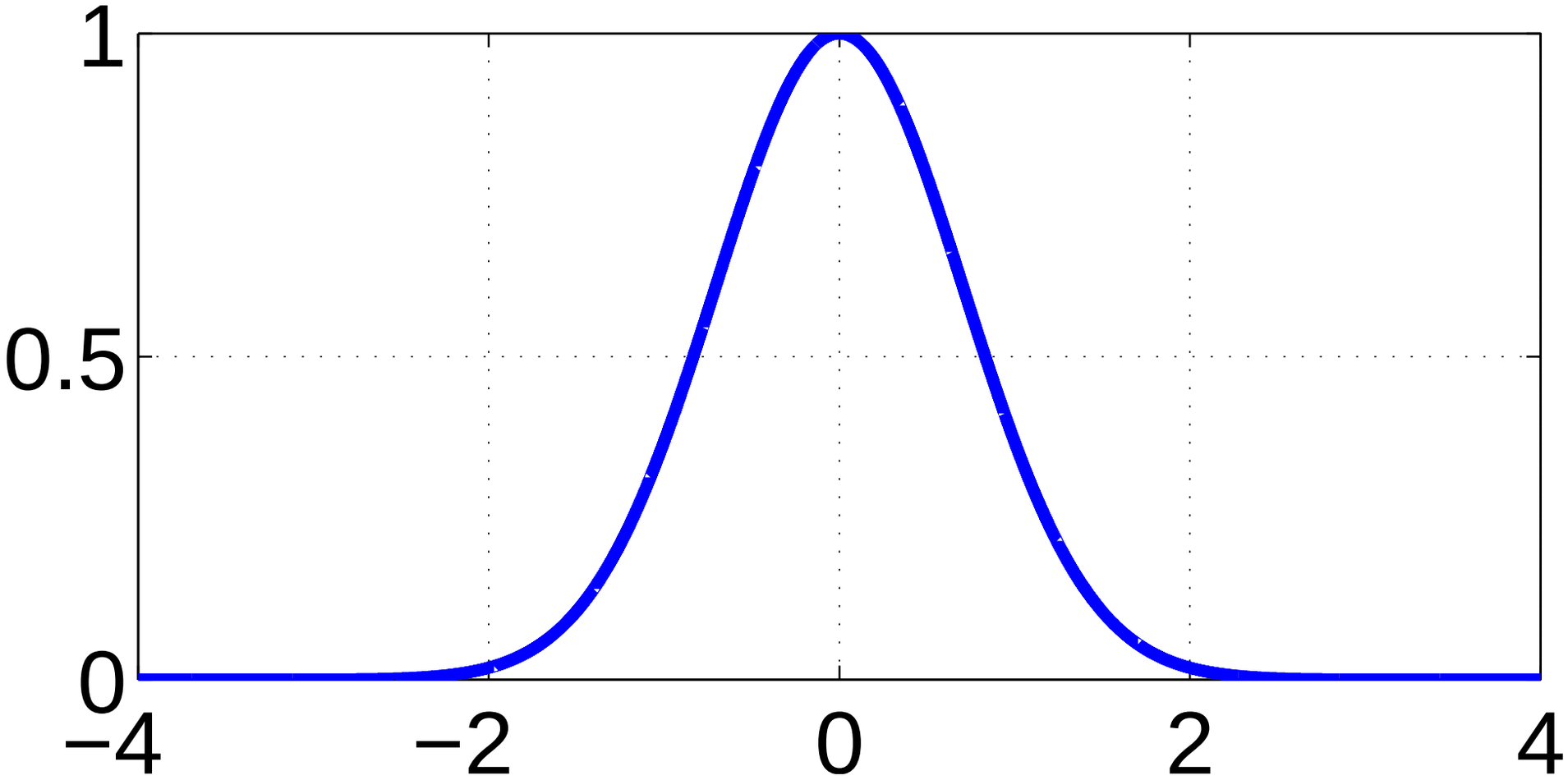
Contributions:

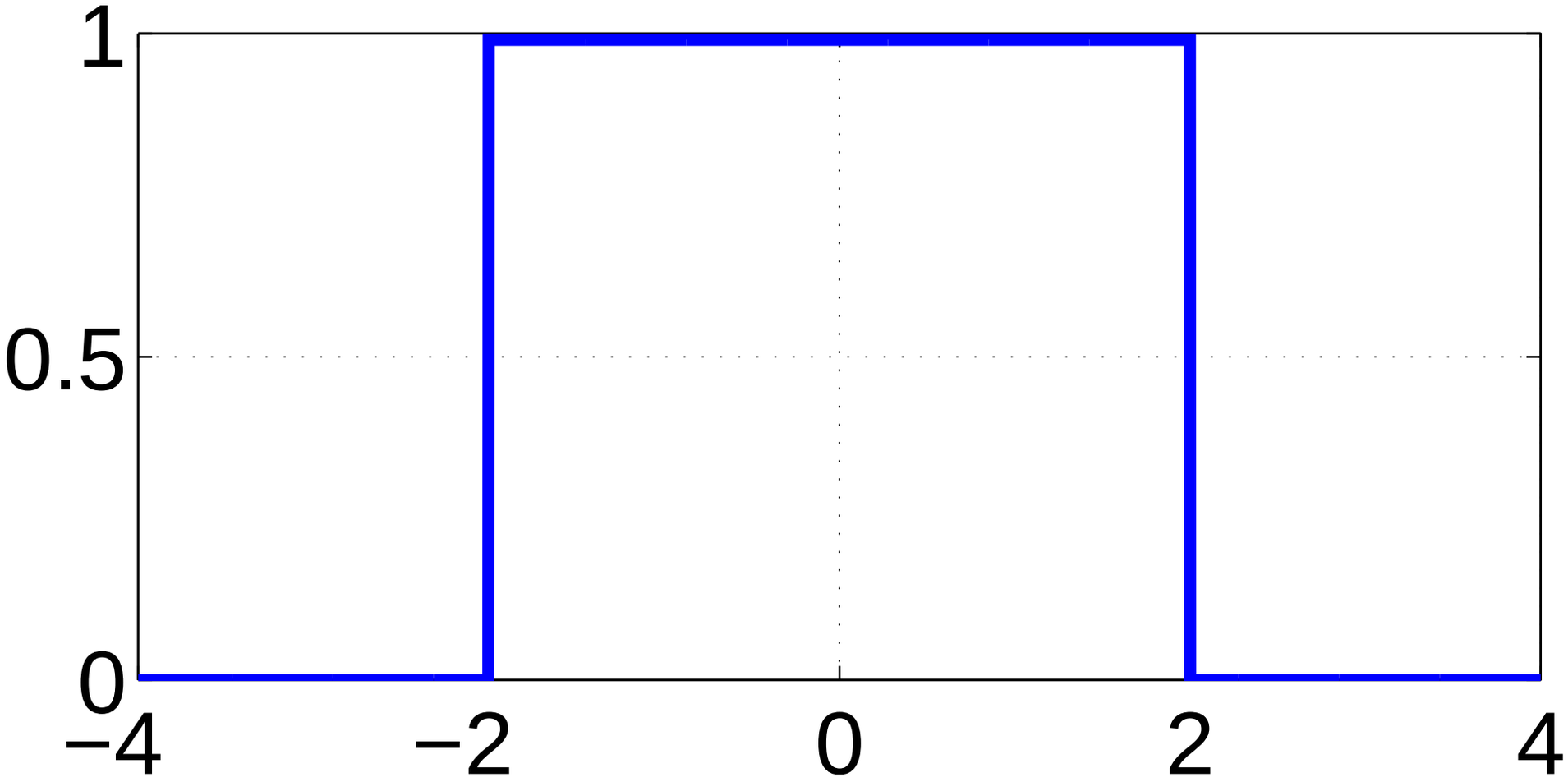
- We set up a probabilistic model which shows that SVMs have a generative flavor.
- The probabilistic model provides simple probabilistic outputs.
- The probabilistic model allows principled engineering of new algorithms in probabilistic framework, e.g.
  - balancing the data;
  - the maximum margin clustering.

Outlook:

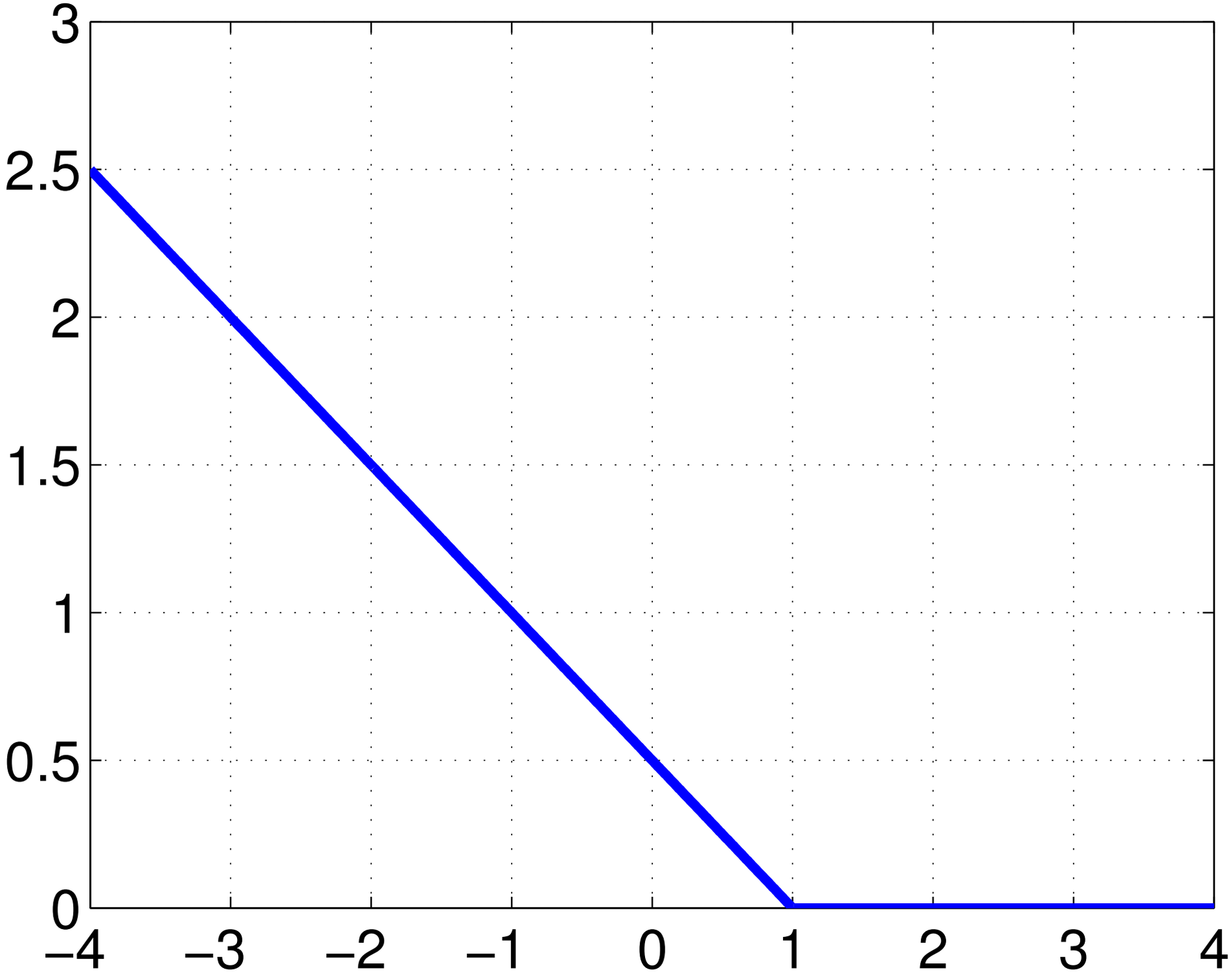
- Does it work with bias?
- Does it work with kernel?
- Can we use Maximum-Likelihood to estimate the remaining hyper-parameters ?
- EM algorithm based treatment of missing data, e.g. semi-supevised learning, missing features etc.
- Different loss functions can be used, e.g. reject option.
- ...

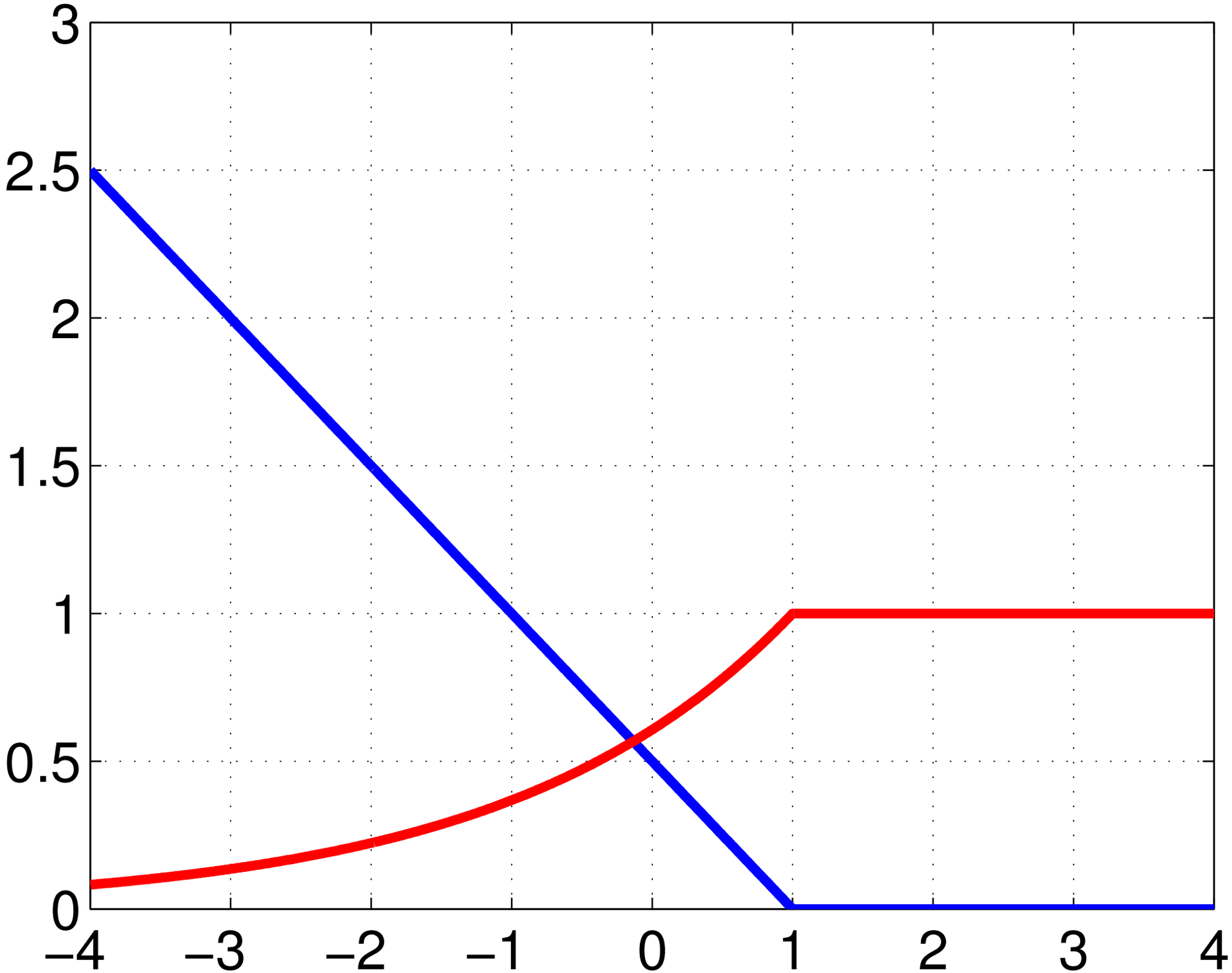
$$p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u}) = \underbrace{Z(\tau, \omega)}_{\text{normalization constant}} \cdot \underbrace{\exp\Big(- \omega^y \ell\big(y\langle \boldsymbol{x}, \tau \boldsymbol{u}\rangle\big)\Big)}_{\text{exponent of hinge loss}} \cdot \underbrace{h(\boldsymbol{x})}_{\text{non-parametric term}}$$

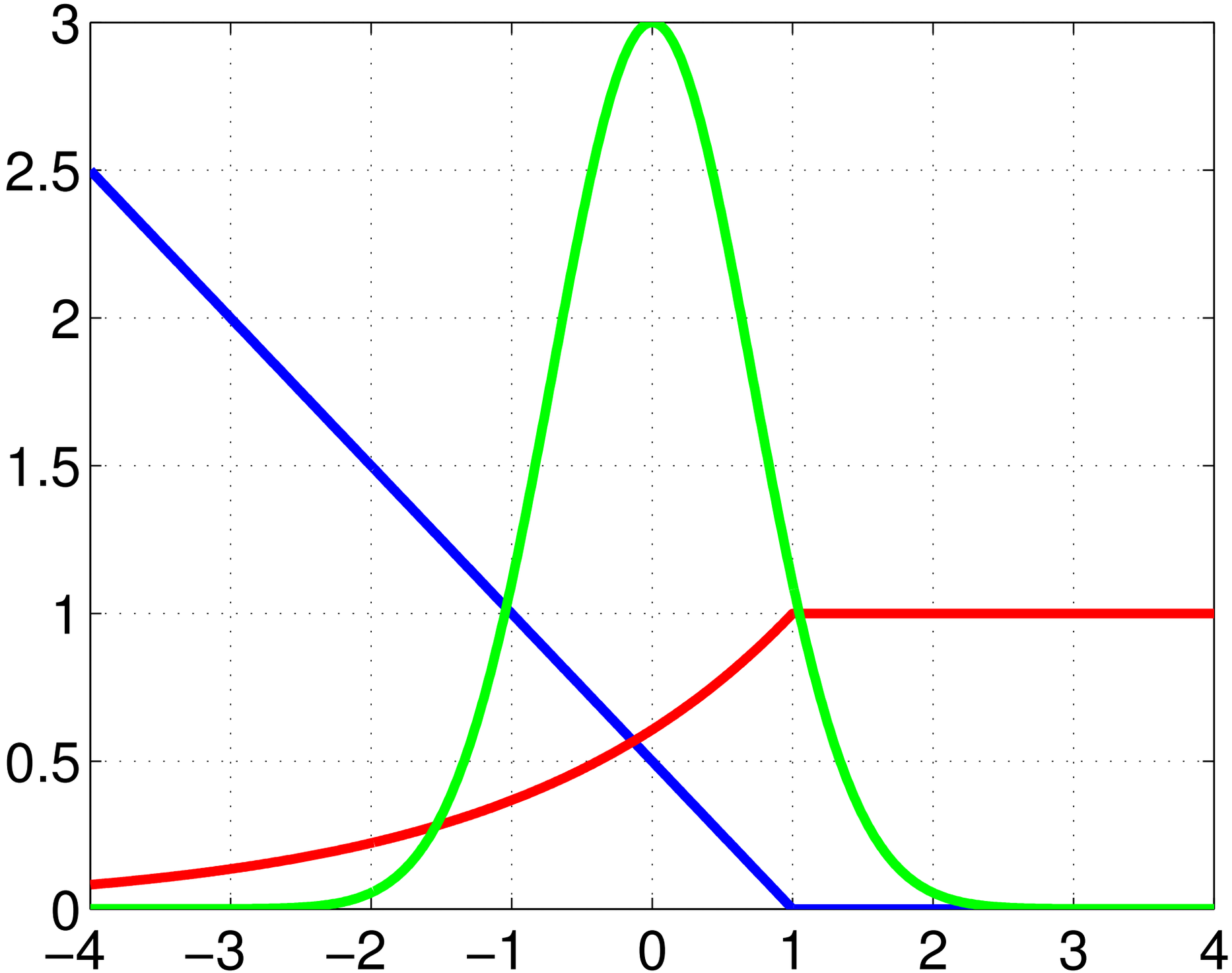

1
0.5
0
−4
−2
0
2
4

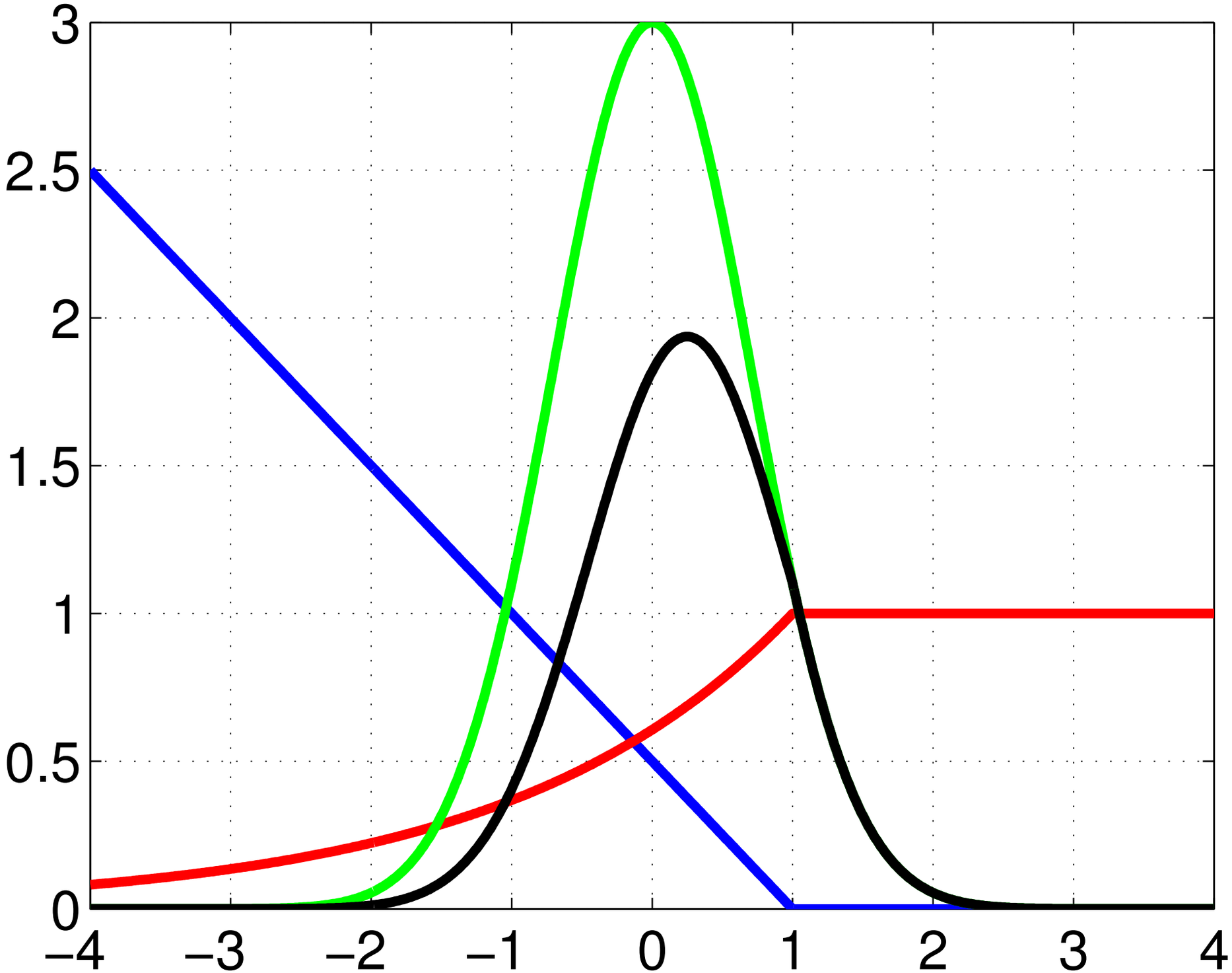

1
0.5
0
−4
−2
0
2
4

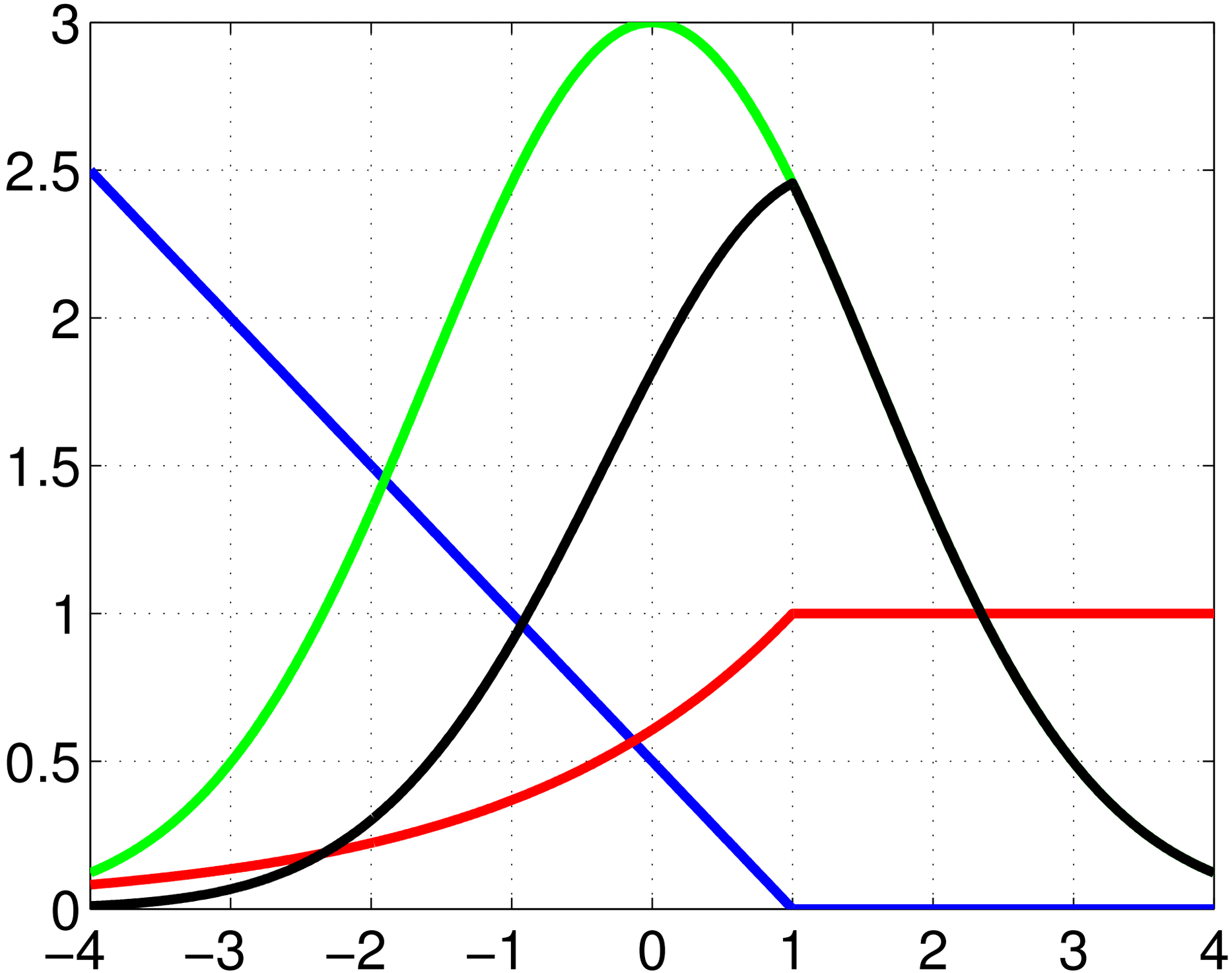
$$p(\boldsymbol{x}, y; \tau, \omega, \boldsymbol{u}) = \underbrace{Z(\tau, \omega)}_{\text{normalization constant}} \cdot \underbrace{\exp\Big(- \omega^y \ell\big(y\langle \boldsymbol{x}, \tau \boldsymbol{u}\rangle\big)\Big)}_{\text{exponent of hinge loss}} \cdot \underbrace{h(\boldsymbol{x})}_{\text{non-parametric term}}$$

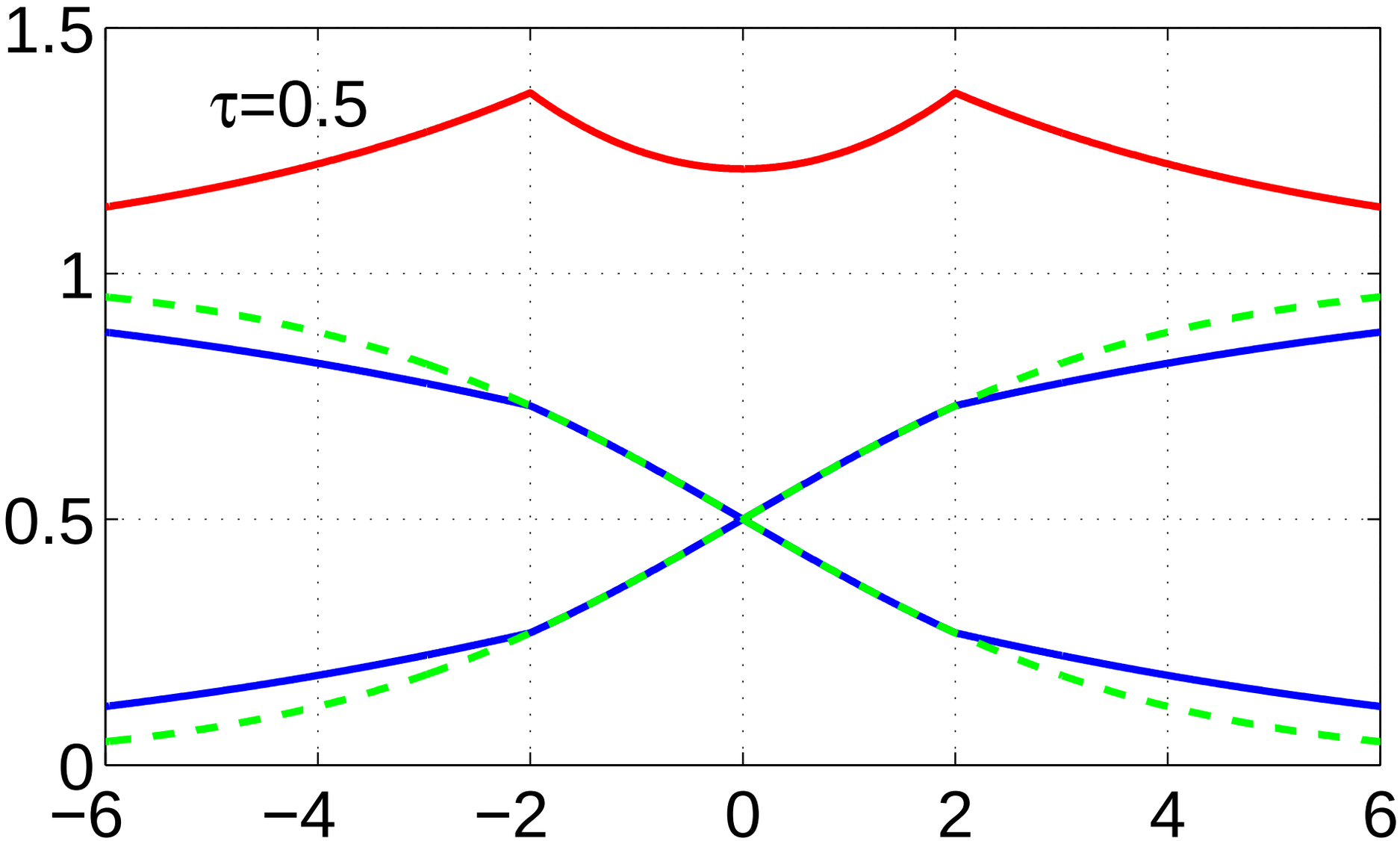

3
2.5
2
1.5
1
0.5
0
−4
−3
−2
−1
0
1
2
3
4

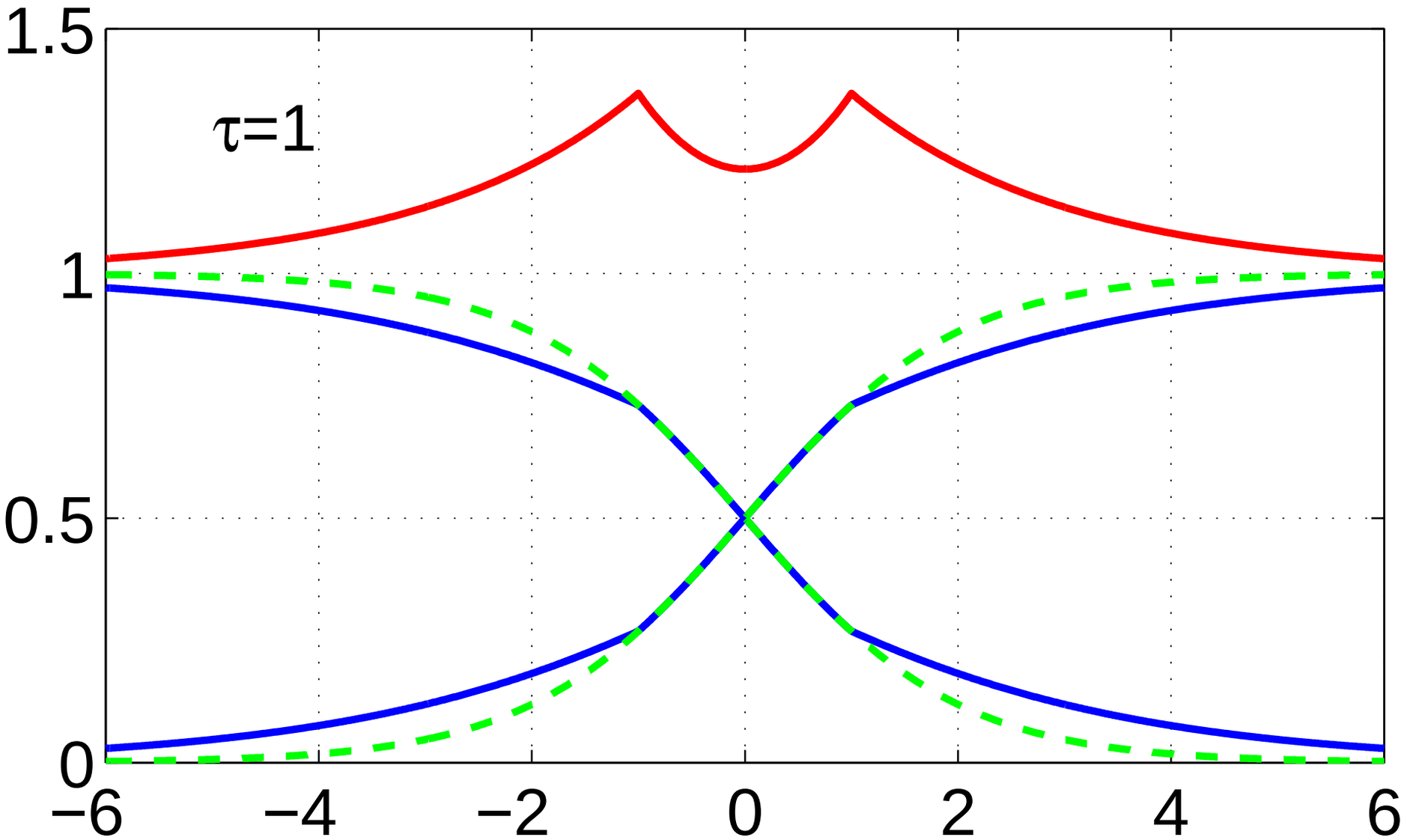

3
2.5
2
1.5
1
0.5
0
−4
−3
−2
−1
0
1
2
3
4

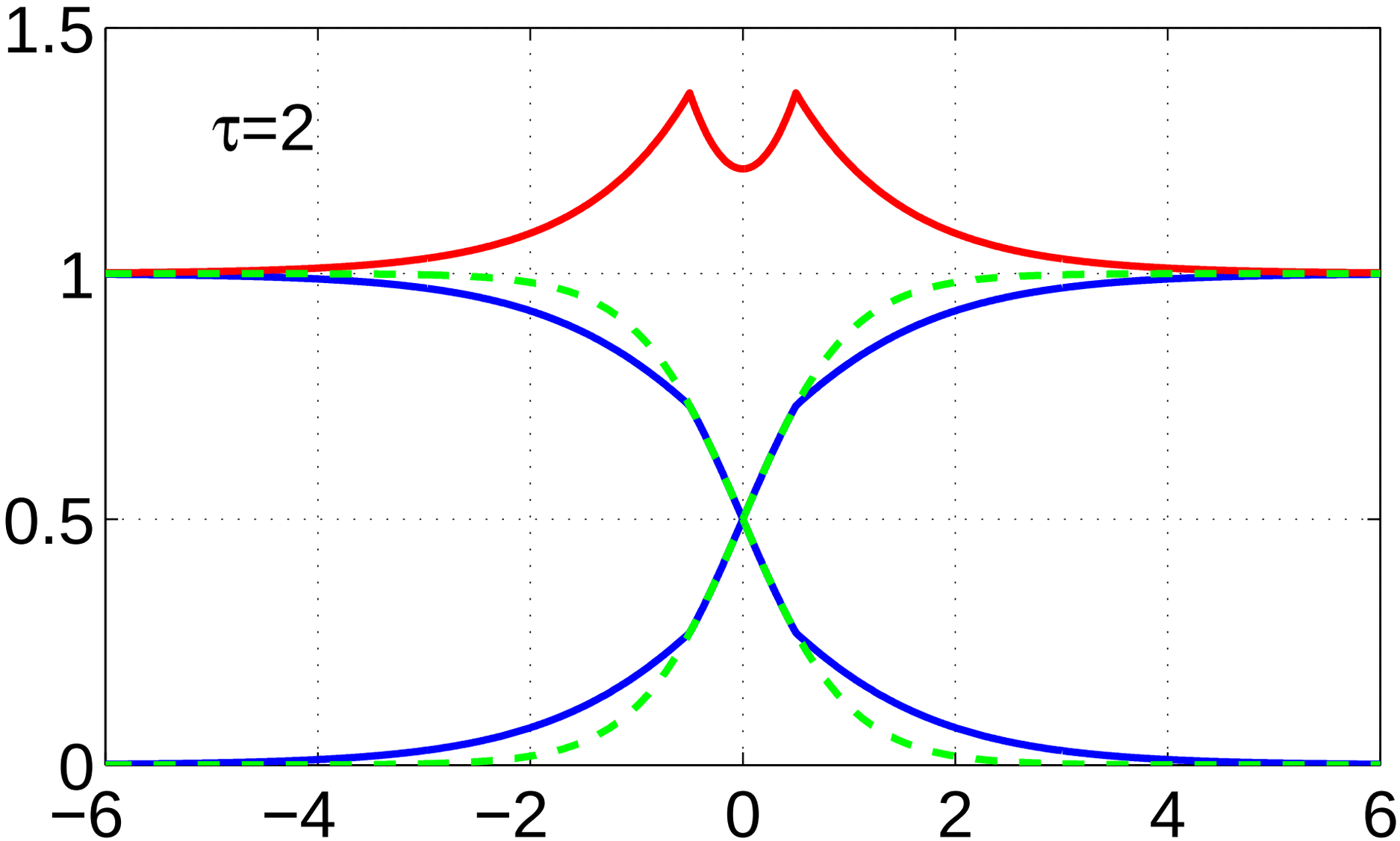

3
2.5
2
1.5
1
0.5
0
−4
−3
−2
−1
0
1
2
3
4


3
2.5
2
1.5
1
0.5
0
−4
−3
−2
−1
0
1
2
3
4


3
2.5
2
1.5
1
0.5
0
−4
−3
−2
−1
0
1
2
3
4


τ=0.5
1.5
1
0.5
0
−6
−4
−2
0
2
4
6


τ=1
1.5
1
0.5
0
−6
−4
−2
0
2
4
6


τ=2
1.5
1
0.5
0
−6
−4
−2
0
2
4
6