PAC-Bayesian Learning of Linear Classifiers

Laval University, Quebec city, Canada

P. Germain A. Lacasse F. Laviolette **M. Marchand**

June 17, 2009

In search of an optimization problem for learning

- The goal of the **learner** is to try to find a **classifier** h with the smallest possible **risk** $R(h)$

$$R(h) \stackrel{\text{def}}{=} \Pr_{(\mathbf{x},y)\sim D} \left\{ h(\mathbf{x}) \neq y \right\} = \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} I\big(h(\mathbf{x}) \neq y\big).$$

- However, we do not know the data-generating distribution D.
- We have only access to $S \stackrel{\text{def}}{=} \{(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_m, y_m)\}$: a training set of m examples, each generated according to D.
- **What should the learner optimize on S to obtain a classifier h having the smallest possible risk $R(h)$?**

In search of an optimization problem for learning

- The goal of the **learner** is to try to find a **classifier** h with the smallest possible **risk** $R(h)$

$$R(h) \stackrel{\text{def}}{=} \Pr_{(\mathbf{x},y)\sim D}\left\{h(\mathbf{x}) \neq y\right\} = \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} I\left(h(\mathbf{x}) \neq y\right).$$

- However, we do not know the data-generating distribution D.
- We have only access to $S \stackrel{\text{def}}{=} \{(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_m, y_m)\}$: a training set of m examples, each generated according to D.
- **What should the learner optimize on S to obtain a classifier h having the smallest possible risk $R(h)$?**

In search of an optimization problem for learning

- The goal of the **learner** is to try to find a **classifier** h with the smallest possible **risk** $R(h)$

$$R(h) \stackrel{\text{def}}{=} \Pr_{(\mathbf{x},y)\sim D} \left\{ h(\mathbf{x}) \neq y \right\} = \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} I(h(\mathbf{x}) \neq y) .$$

- However, we do not know the data-generating distribution D.
- We have only access to $S \stackrel{\text{def}}{=} \{(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_m, y_m)\}$: a training set of m examples, each generated according to D.
- What should the learner optimize on S to obtain a classifier h having the smallest possible risk $R(h)$?

In search of an optimization problem for learning

- The goal of the **learner** is to try to find a **classifier** h with the smallest possible **risk** $R(h)$

$$R(h) \stackrel{\text{def}}{=} \Pr_{(\mathbf{x},y)\sim D} \left\{ h(\mathbf{x}) \neq y \right\} = \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} I\big(h(\mathbf{x}) \neq y\big) .$$

- However, we do not know the data-generating distribution D.
- We have only access to $S \stackrel{\text{def}}{=} \{(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_m, y_m)\}$: a training set of m examples, each generated according to D.
- **What should the learner optimize on S to obtain a classifier h having the smallest possible risk $R(h)$?**

PAC-Bayesian bound minimization

- Several objective functions (to minimize on S) have been proposed.
  - The soft-margin SVM, AdaBoost, ridge regression...
- But the final rigorous (and accepted) guarantee is always a **risk bound** that holds uniformly over a space of classifiers.
- **Let us try to design an efficient learning algorithm that minimizes the PAC-Bayes bound.**

PAC-Bayesian bound minimization

- Several objective functions (to minimize on S) have been proposed.
  - The soft-margin SVM, AdaBoost, ridge regression...
- But the final rigorous (and accepted) guarantee is always a **risk bound** that holds uniformly over a space of classifiers.
- Let us try to design an efficient learning algorithm that minimizes the PAC-Bayes bound.

PAC-Bayesian bound minimization

- Several objective functions (to minimize on S) have been proposed.
    - The soft-margin SVM, AdaBoost, ridge regression...
- But the final rigorous (and accepted) guarantee is always a **risk bound** that holds uniformly over a space of classifiers.
- **Let us try to design an efficient learning algorithm that minimizes the PAC-Bayes bound.**

Summary

- A simple and general theorem from which all other known PAC-Bayes bound can be simply derived.
- Specialization to linear classifiers.
- Gradient descent of the PAC-Bayes bound: three different learning algorithms.
- Extensive numerical results and comparison with AdaBoost and the SVM.

Summary

- A simple and general theorem from which all other known PAC-Bayes bound can be simply derived.
- Specialization to linear classifiers.
- Gradient descent of the PAC-Bayes bound: three different learning algorithms.
- Extensive numerical results and comparison with AdaBoost and the SVM.

Summary

- A simple and general theorem from which all other known PAC-Bayes bound can be simply derived.
- Specialization to linear classifiers.
- Gradient descent of the PAC-Bayes bound: three different learning algorithms.
- Extensive numerical results and comparison with AdaBoost and the SVM.

Summary

- A simple and general theorem from which all other known PAC-Bayes bound can be simply derived.
- Specialization to linear classifiers.
- Gradient descent of the PAC-Bayes bound: three different learning algorithms.
- Extensive numerical results and comparison with AdaBoost and the SVM.

Definitions

- The (true) risk $R(h)$ and training error $R_S(h)$ are defined as:

$$R(h) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} I(h(\mathbf{x}) \neq y) \quad ; \quad R_S(h) \stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} I(h(\mathbf{x}_i) \neq y_i)\,.$$

- The learner's goal is to choose a **posterior distribution** Q on a space $\mathcal{H}$ of classifiers such that the risk of the Q-weighted **majority vote** B_Q is as small as possible.
- PAC-Bayes bounds the risk of the **Gibbs classifier** G_Q. To predict the label of $\mathbf{x}$, G_Q draws h from $\mathcal{H}$ and predicts $h(\mathbf{x})$.
- The risk and the training error of G_Q are thus defined as:

$$R(G_Q) = \mathop{\mathbf{E}}_{h\sim Q} R(h) \quad ; \quad R_S(G_Q) = \mathop{\mathbf{E}}_{h\sim Q} R_S(h)\,.$$

Definitions

- The (true) risk $R(h)$ and training error $R_S(h)$ are defined as:

$$R(h) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} I(h(\mathbf{x}) \neq y) \quad ; \quad R_S(h) \stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} I(h(\mathbf{x}_i) \neq y_i)\,.$$

- The learner's goal is to choose a **posterior distribution** Q on a space $\mathcal{H}$ of classifiers such that the risk of the Q-weighted **majority vote** B_Q is as small as possible.
- PAC-Bayes bounds the risk of the **Gibbs classifier** G_Q. To predict the label of $\mathbf{x}$, G_Q draws h from $\mathcal{H}$ and predicts $h(\mathbf{x})$.
- The risk and the training error of G_Q are thus defined as:

$$R(G_Q) = \mathop{\mathbf{E}}_{h\sim Q} R(h) \quad ; \quad R_S(G_Q) = \mathop{\mathbf{E}}_{h\sim Q} R_S(h)\,.$$

Definitions

- The (true) risk $R(h)$ and training error $R_S(h)$ are defined as:

$$R(h) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} I(h(\mathbf{x}) \neq y) \quad ; \quad R_S(h) \stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} I(h(\mathbf{x}_i) \neq y_i)\,.$$

- The learner's goal is to choose a **posterior distribution** Q on a space $\mathcal{H}$ of classifiers such that the risk of the Q-weighted **majority vote** B_Q is as small as possible.
- PAC-Bayes bounds the risk of the **Gibbs classifier** G_Q. To predict the label of $\mathbf{x}$, G_Q draws h from $\mathcal{H}$ and predicts $h(\mathbf{x})$.
- The risk and the training error of G_Q are thus defined as:

$$R(G_Q) = \mathop{\mathbf{E}}_{h\sim Q} R(h) \quad ; \quad R_S(G_Q) = \mathop{\mathbf{E}}_{h\sim Q} R_S(h)\,.$$

Definitions

- The (true) risk $R(h)$ and training error $R_S(h)$ are defined as:

$$R(h) \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} I(h(\mathbf{x}) \neq y) \quad ; \quad R_S(h) \stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} I(h(\mathbf{x}_i) \neq y_i)\,.$$

- The learner's goal is to choose a **posterior distribution** Q on a space $\mathcal{H}$ of classifiers such that the risk of the Q-weighted **majority vote** B_Q is as small as possible.
- PAC-Bayes bounds the risk of the **Gibbs classifier** G_Q. To predict the label of $\mathbf{x}$, G_Q draws h from $\mathcal{H}$ and predicts $h(\mathbf{x})$.
- The risk and the training error of G_Q are thus defined as:

$$R(G_Q) = \mathop{\mathbf{E}}_{h\sim Q} R(h) \quad ; \quad R_S(G_Q) = \mathop{\mathbf{E}}_{h\sim Q} R_S(h)\,.$$

G_Q, B_Q, and $\mathrm{KL}(Q\|P)$

- If B_Q misclassifies $\mathbf{x}$, then at least half of the classifiers (under measure Q) err on $\mathbf{x}$. Then $R(B_Q) \leq 2R(G_Q)$: **An upper bound on $R(G_Q)$ also gives an upper bound on $R(B_Q)$.**
- PAC-Bayes makes use of a **prior distribution** P on $\mathcal{H}$.
- The risk bound depends on the **Kullback-Leibler divergence** $\mathrm{KL}(Q\|P)$:

$$\mathrm{KL}(Q\|P) \stackrel{\mathrm{def}}{=} \mathop{\mathbf{E}}_{h\sim Q} \ln \frac{Q(h)}{P(h)} .$$

- We will also make use of the of the KL divergence between Bernoulli distributions of probability of success p and q:

$$\mathrm{kl}(q,p) \stackrel{\mathrm{def}}{=} q \ln \frac{q}{p} + (1-q) \ln \frac{1-q}{1-p} .$$

G_Q, B_Q, and $\mathrm{KL}(Q\|P)$

- If B_Q misclassifies $\mathbf{x}$, then at least half of the classifiers (under measure Q) err on $\mathbf{x}$. Then $R(B_Q) \leq 2R(G_Q)$: **An upper bound on $R(G_Q)$ also gives an upper bound on $R(B_Q)$.**
- PAC-Bayes makes use of a **prior distribution** P on $\mathcal{H}$.
- The risk bound depends on the **Kullback-Leibler divergence** $\mathrm{KL}(Q\|P)$:

$$\mathrm{KL}(Q\|P) \stackrel{\mathrm{def}}{=} \mathop{\mathbf{E}}_{h\sim Q} \ln \frac{Q(h)}{P(h)} .$$

- We will also make use of the of the KL divergence between Bernoulli distributions of probability of success p and q:

$$\mathrm{kl}(q,p) \stackrel{\mathrm{def}}{=} q \ln \frac{q}{p} + (1-q) \ln \frac{1-q}{1-p} .$$

G_Q, B_Q, and $\mathrm{KL}(Q\|P)$

- If B_Q misclassifies $\mathbf{x}$, then at least half of the classifiers (under measure Q) err on $\mathbf{x}$. Then $R(B_Q) \leq 2R(G_Q)$: **An upper bound on $R(G_Q)$ also gives an upper bound on $R(B_Q)$.**
- PAC-Bayes makes use of a **prior distribution** P on $\mathcal{H}$.
- The risk bound depends on the **Kullback-Leibler divergence** $\mathrm{KL}(Q\|P)$:

$$\mathrm{KL}(Q\|P) \stackrel{\mathrm{def}}{=} \mathop{\mathbf{E}}_{h\sim Q} \ln \frac{Q(h)}{P(h)}\,.$$

- We will also make use of the of the KL divergence between Bernoulli distributions of probability of success p and q:

$$\mathrm{kl}(q,p) \stackrel{\mathrm{def}}{=} q\ln\frac{q}{p} + (1-q)\ln\frac{1-q}{1-p}\,.$$

G_Q, B_Q, and $\mathrm{KL}(Q\|P)$

- If B_Q misclassifies $\mathbf{x}$, then at least half of the classifiers (under measure Q) err on $\mathbf{x}$. Then $R(B_Q) \leq 2R(G_Q)$: **An upper bound on $R(G_Q)$ also gives an upper bound on $R(B_Q)$.**
- PAC-Bayes makes use of a **prior distribution** P on $\mathcal{H}$.
- The risk bound depends on the **Kullback-Leibler divergence** $\mathrm{KL}(Q\|P)$:

$$\mathrm{KL}(Q\|P) \stackrel{\mathrm{def}}{=} \mathop{\mathbf{E}}_{h\sim Q} \ln \frac{Q(h)}{P(h)}.$$

- We will also make use of the of the KL divergence between Bernoulli distributions of probability of success p and q:

$$\mathrm{kl}(q,p) \stackrel{\mathrm{def}}{=} q \ln \frac{q}{p} + (1-q) \ln \frac{1-q}{1-p}.$$

The General PAC-Bayes Theorem

Theorem 1

For any distribution D, for any set $\mathcal{H}$ of classifiers, for any prior distribution P of support $\mathcal{H}$, for any $\delta \in (0,1]$, and for any convex function $\mathcal{D} : [0,1] \times [0,1] \to \mathbb{R}$, we have

$$\Pr_{S \sim D^m}\left(\forall\, Q \text{ on } \mathcal{H}\colon\; \mathcal{D}(R_S(G_Q), R(G_Q)) \leq \frac{1}{m}\left[\mathrm{KL}(Q\|P) + \ln\left(\frac{1}{\delta} \mathop{\mathbf{E}}_{S \sim D^m} \mathop{\mathbf{E}}_{h \sim P} e^{m\mathcal{D}(R_S(h), R(h))}\right)\right]\right) \geq 1 - \delta\,.$$

The Langford (2005) and Seeger (2002) bound

We recover a slightly tighter version if $\mathcal{D}(q,p) = \mathrm{kl}(q,p)$ and

$$\mathop{\mathbf{E}}_{S\sim D^m} \mathop{\mathbf{E}}_{h\sim P} e^{m\,\mathrm{kl}(R_S(h),R(h))} = \sum_{k=0}^{m} \binom{m}{k} (k/m)^k (1-k/m)^{m-k} \stackrel{\text{def}}{=} \xi(m) \in \Theta(\sqrt{m})\,.$$

Corollary 2.1

For any D, any $\mathcal{H}$, any P of support $\mathcal{H}$, any $\delta \in (0,1]$, we have

$$\Pr_{S\sim D^m}\left(\forall\, Q \text{ on } \mathcal{H}\colon\ \mathrm{kl}(R_S(G_Q), R(G_Q)) \le \frac{1}{m}\left[\mathrm{KL}(Q\|P) + \ln\frac{\xi(m)}{\delta}\right]\right) \ge 1-\delta\,,$$

The Langford (2005) and Seeger (2002) bound

We recover a slightly tighter version if $\mathcal{D}(q,p) = \mathrm{kl}(q,p)$ and

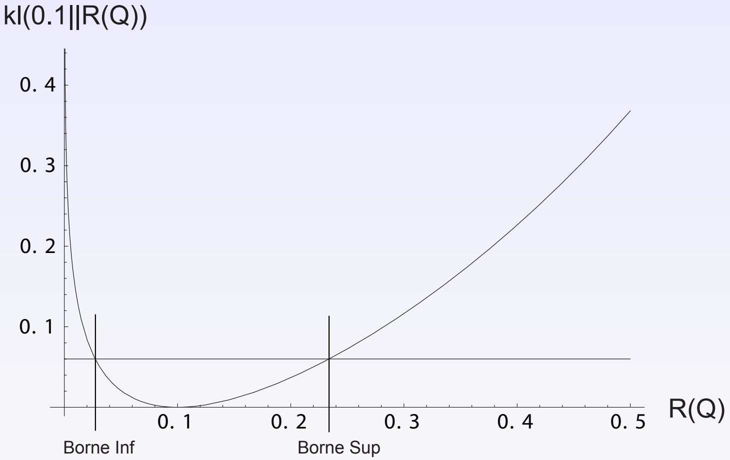
$$\mathop{\mathbf{E}}_{S\sim D^m} \mathop{\mathbf{E}}_{h\sim P} e^{m\,\mathrm{kl}(R_S(h),R(h))} = \sum_{k=0}^{m} \binom{m}{k} (k/m)^k (1-k/m)^{m-k} \stackrel{\mathrm{def}}{=} \xi(m) \in \Theta(\sqrt{m})\,.$$

Corollary 2.1

For any D, any $\mathcal{H}$, any P of support $\mathcal{H}$, any $\delta \in (0,1]$, we have

$$\Pr_{S\sim D^m}\left(\forall\, Q \text{ on } \mathcal{H}\colon\ \mathrm{kl}(R_S(G_Q), R(G_Q)) \le \frac{1}{m}\left[\mathrm{KL}(Q\|P) + \ln\frac{\xi(m)}{\delta}\right]\right) \ge 1-\delta\,,$$

Graphical illustration of the Langford-Seeger bound



A bound also found by Catoni (2007)

Let $\mathcal{D}(q,p) = \mathcal{F}(p) - C \cdot q$. Then

$$\mathop{\mathbf{E}}_{S\sim D^m} \mathop{\mathbf{E}}_{h\sim P} e^{m\,\mathcal{D}(R_S(h),R(h))} \quad = \quad \mathop{\mathbf{E}}_{h\sim P} e^{m\mathcal{F}(R(h))} \left(R(h)e^{-C} + (1-R(h))\right)^m .$$

Corollary 2.2

For any D, any $\mathcal{H}$, any P of support $\mathcal{H}$, any $\delta \in (0,1]$, and any positive real number C, we have

$$\Pr_{S\sim D^m}\left(\begin{array}{l} \forall\, Q \text{ on } \mathcal{H}: \\ R(G_Q) \le \frac{1}{1-e^{-C}}\left\{1-\exp\left[-\left(C\cdot R_S(G_Q)\right.\right.\right. \\ \qquad\qquad \left.\left.\left. +\frac{1}{m}\left[\mathrm{KL}(Q\|P)+\ln\frac{1}{\delta}\right]\right)\right]\right\} \end{array}\right) \quad \ge \quad 1-\delta.$$

A bound also found by Catoni (2007)

Let $\mathcal{D}(q,p) = \mathcal{F}(p) - C \cdot q$. Then

$$\mathop{\mathbf{E}}_{S\sim D^m} \mathop{\mathbf{E}}_{h\sim P} e^{m\mathcal{D}(R_S(h),R(h))} = \mathop{\mathbf{E}}_{h\sim P} e^{m\mathcal{F}(R(h))} \left(R(h)e^{-C} + (1-R(h))\right)^m .$$

Corollary 2.2

For any D, any $\mathcal{H}$, any P of support $\mathcal{H}$, any $\delta \in (0,1]$, and any positive real number C, we have

$$\Pr_{S\sim D^m}\left(\begin{array}{l} \forall\, Q \text{ on } \mathcal{H}: \\ R(G_Q) \le \frac{1}{1-e^{-C}}\Big\{1 - \exp\Big[-\Big(C\cdot R_S(G_Q) \\ \qquad\qquad + \frac{1}{m}\big[\mathrm{KL}(Q\|P) + \ln\frac{1}{\delta}\big]\Big)\Big]\Big\} \end{array}\right) \ge 1 - \delta.$$

Observations about Corollary 2.2

- G_Q is minimizing the bound of Corollary 2.2 iff it minimizes the following cost function (linear in $R_S(G_Q)$):

$$C\, m\, R_S(G_Q) + \mathrm{KL}(Q \| P)$$

- However, we have **hyperparameter** C to tune (in contrast with the bound of Corollary 2.1).
- Corollary 2.1 gives a bound which is always tighter except for a narrow range of C values.
- In fact, if we would replace $\xi(m)$ by one, Corollary 2.1 would always give a tighter bound.

Observations about Corollary 2.2

- G_Q is minimizing the bound of Corollary 2.2 iff it minimizes the following cost function (linear in $R_S(G_Q)$):

$$C\, m\, R_S(G_Q) + \mathrm{KL}(Q\|P)$$

- However, we have **hyperparameter** C to tune (in contrast with the bound of Corollary 2.1).
- Corollary 2.1 gives a bound which is always tighter except for a narrow range of C values.
- In fact, if we would replace $\xi(m)$ by one, Corollary 2.1 would always give a tighter bound.

Observations about Corollary 2.2

- G_Q is minimizing the bound of Corollary 2.2 iff it minimizes the following cost function (linear in $R_S(G_Q)$):

$$C\, m\, R_S(G_Q) + \mathrm{KL}(Q\|P)$$

- However, we have **hyperparameter** C to tune (in contrast with the bound of Corollary 2.1).
- Corollary 2.1 gives a bound which is always tighter except for a narrow range of C values.
- In fact, if we would replace $\xi(m)$ by one, Corollary 2.1 would always give a tighter bound.

Linear classifiers

- Each $\mathbf{x}$ is mapped to a high-dimensional feature vector $\boldsymbol{\phi}(\mathbf{x})$:

$$\boldsymbol{\phi}(\mathbf{x}) \stackrel{\text{def}}{=} (\phi_1(\mathbf{x}), \ldots, \phi_N(\mathbf{x}))\,.$$

- $\boldsymbol{\phi}$ is often implicitly given by a Mercer kernel

$$k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}')\,.$$

- The output $h_{\mathbf{v}}(\mathbf{x})$ of linear classifier $h_{\mathbf{v}}$ with weight vector $\mathbf{v}$ is given by

$$h_{\mathbf{v}}(\mathbf{x}) \;=\; \operatorname{sgn}\left(\mathbf{v} \cdot \boldsymbol{\phi}(\mathbf{x})\right)\,.$$

- Each posterior $Q_{\mathbf{w}}$ is an isotropic Gaussian centered on $\mathbf{w}$:

$$Q_{\mathbf{w}}(\mathbf{v}) \;=\; \left(\frac{1}{\sqrt{2\pi}}\right)^{N} \exp\left(-\frac{1}{2}\|\mathbf{v}-\mathbf{w}\|^2\right)$$

Linear classifiers

- Each $\mathbf{x}$ is mapped to a high-dimensional feature vector $\boldsymbol{\phi}(\mathbf{x})$:

$$\boldsymbol{\phi}(\mathbf{x}) \stackrel{\text{def}}{=} (\phi_1(\mathbf{x}), \ldots, \phi_N(\mathbf{x}))\,.$$

- $\boldsymbol{\phi}$ is often implicitly given by a Mercer kernel

$$k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}')\,.$$

- The output $h_{\mathbf{v}}(\mathbf{x})$ of linear classifier $h_{\mathbf{v}}$ with weight vector $\mathbf{v}$ is given by

$$h_{\mathbf{v}}(\mathbf{x}) = \operatorname{sgn}(\mathbf{v} \cdot \boldsymbol{\phi}(\mathbf{x}))\,.$$

- Each posterior $Q_{\mathbf{w}}$ is an isotropic Gaussian centered on $\mathbf{w}$:

$$Q_{\mathbf{w}}(\mathbf{v}) = \left(\frac{1}{\sqrt{2\pi}}\right)^N \exp\left(-\frac{1}{2}\|\mathbf{v} - \mathbf{w}\|^2\right)$$

Linear classifiers

- Each $\mathbf{x}$ is mapped to a high-dimensional feature vector $\boldsymbol{\phi}(\mathbf{x})$:

$$\boldsymbol{\phi}(\mathbf{x}) \stackrel{\text{def}}{=} (\phi_1(\mathbf{x}), \ldots, \phi_N(\mathbf{x}))\,.$$

- $\boldsymbol{\phi}$ is often implicitly given by a Mercer kernel

$$k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}')\,.$$

- The output $h_{\mathbf{v}}(\mathbf{x})$ of linear classifier $h_{\mathbf{v}}$ with weight vector $\mathbf{v}$ is given by

$$h_{\mathbf{v}}(\mathbf{x}) \;=\; \operatorname{sgn}\left(\mathbf{v} \cdot \boldsymbol{\phi}(\mathbf{x})\right)\,.$$

- Each posterior $Q_{\mathbf{w}}$ is an isotropic Gaussian centered on $\mathbf{w}$:

$$Q_{\mathbf{w}}(\mathbf{v}) \;=\; \left(\frac{1}{\sqrt{2\pi}}\right)^N \exp\left(-\frac{1}{2}\|\mathbf{v}-\mathbf{w}\|^2\right)$$

Linear classifiers

- Each $\mathbf{x}$ is mapped to a high-dimensional feature vector $\boldsymbol{\phi}(\mathbf{x})$:

$$\boldsymbol{\phi}(\mathbf{x}) \stackrel{\text{def}}{=} (\phi_1(\mathbf{x}), \ldots, \phi_N(\mathbf{x}))\,.$$

- $\boldsymbol{\phi}$ is often implicitly given by a Mercer kernel

$$k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}')\,.$$

- The output $h_{\mathbf{v}}(\mathbf{x})$ of linear classifier $h_{\mathbf{v}}$ with weight vector $\mathbf{v}$ is given by

$$h_{\mathbf{v}}(\mathbf{x}) \;=\; \operatorname{sgn}\left(\mathbf{v} \cdot \boldsymbol{\phi}(\mathbf{x})\right)\,.$$

- Each posterior $Q_{\mathbf{w}}$ is an isotropic Gaussian centered on $\mathbf{w}$:

$$Q_{\mathbf{w}}(\mathbf{v}) \;=\; \left(\frac{1}{\sqrt{2\pi}}\right)^N \exp\left(-\frac{1}{2}\|\mathbf{v}-\mathbf{w}\|^2\right)$$

Bayes-equivalent classifiers

- With this choice for $Q_{\mathbf{w}}$, the majority vote $B_{Q_{\mathbf{w}}}$ is the same classifier as $h_{\mathbf{w}}$ since:

$$B_{Q_{\mathbf{w}}}(\mathbf{x}) \;=\; \operatorname{sgn}\left(\mathop{\mathbf{E}}_{\mathbf{v}\sim Q_{\mathbf{w}}} \operatorname{sgn}\left(\mathbf{v}\cdot\phi(\mathbf{x})\right)\right) \;=\; \operatorname{sgn}\left(\mathbf{w}\cdot\phi(\mathbf{x})\right) \;=\; h_{\mathbf{w}}(\mathbf{x})\,.$$

- Thus $R(h_{\mathbf{w}}) = R(B_{Q_{\mathbf{w}}}) \leq 2R(G_{Q_{\mathbf{w}}})$: an upper bound on $R(G_{Q_{\mathbf{w}}})$ also provides an upper bound on $R(h_{\mathbf{w}})$.
- The prior $P_{\mathbf{w}_p}$ is also an isotropic Gaussian centered on $\mathbf{w}_p$. Consequently:

$$\mathrm{KL}(Q_{\mathbf{w}} \| P_{\mathbf{w}_p}) \;=\; \frac{1}{2}\|\mathbf{w}-\mathbf{w}_p\|^2\,.$$

Bayes-equivalent classifiers

- With this choice for $Q_{\mathbf{w}}$, the majority vote $B_{Q_{\mathbf{w}}}$ is the same classifier as $h_{\mathbf{w}}$ since:

$$B_{Q_{\mathbf{w}}}(\mathbf{x}) \;=\; \mathrm{sgn}\left(\mathop{\mathbf{E}}_{\mathbf{v}\sim Q_{\mathbf{w}}} \mathrm{sgn}\left(\mathbf{v}\cdot\phi(\mathbf{x})\right)\right) \;=\; \mathrm{sgn}\left(\mathbf{w}\cdot\phi(\mathbf{x})\right) \;=\; h_{\mathbf{w}}(\mathbf{x})\,.$$

- Thus $R(h_{\mathbf{w}}) = R(B_{Q_{\mathbf{w}}}) \leq 2R(G_{Q_{\mathbf{w}}})$: an upper bound on $R(G_{Q_{\mathbf{w}}})$ also provides an upper bound on $R(h_{\mathbf{w}})$.
- The prior $P_{\mathbf{w}_p}$ is also an isotropic Gaussian centered on $\mathbf{w}_p$. Consequently:

$$\mathrm{KL}(Q_{\mathbf{w}}\|P_{\mathbf{w}_p}) \;=\; \frac{1}{2}\|\mathbf{w}-\mathbf{w}_p\|^2\,.$$

Bayes-equivalent classifiers

- With this choice for $Q_{\mathbf{w}}$, the majority vote $B_{Q_{\mathbf{w}}}$ is the same classifier as $h_{\mathbf{w}}$ since:

$$B_{Q_{\mathbf{w}}}(\mathbf{x}) = \mathrm{sgn}\left(\mathop{\mathbf{E}}_{\mathbf{v}\sim Q_{\mathbf{w}}} \mathrm{sgn}\left(\mathbf{v}\cdot\phi(\mathbf{x})\right)\right) = \mathrm{sgn}\left(\mathbf{w}\cdot\phi(\mathbf{x})\right) = h_{\mathbf{w}}(\mathbf{x})\,.$$

- Thus $R(h_{\mathbf{w}}) = R(B_{Q_{\mathbf{w}}}) \le 2R(G_{Q_{\mathbf{w}}})$: an upper bound on $R(G_{Q_{\mathbf{w}}})$ also provides an upper bound on $R(h_{\mathbf{w}})$.
- The prior $P_{\mathbf{w}_p}$ is also an isotropic Gaussian centered on $\mathbf{w}_p$. Consequently:

$$\mathrm{KL}(Q_{\mathbf{w}} \| P_{\mathbf{w}_p}) = \frac{1}{2}\|\mathbf{w} - \mathbf{w}_p\|^2\,.$$

Gibbs' risk

We need to compute Gibb's risk $R_{(\mathbf{x},y)}(G_{Q_\mathbf{w}})$ on $(\mathbf{x}, y)$ since:

$$R_{(\mathbf{x},y)}(G_{Q_\mathbf{w}}) \stackrel{\text{def}}{=} \int_{\mathbb{R}^N} d\mathbf{v} Q_\mathbf{w}(\mathbf{v}) I(y\mathbf{v} \cdot \phi(\mathbf{x}) < 0)$$

$$R(G_{Q_\mathbf{w}}) = \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} R_{(\mathbf{x},y)}(G_{Q_\mathbf{w}}) \quad ; \quad R_S(G_{Q_\mathbf{w}}) = \frac{1}{m}\sum_{i=1}^{m} R_{(\mathbf{x}_i,y_i)}(G_{Q_\mathbf{w}}) \,.$$

As in Langford (2005), the Gaussian integral gives:

$$R_{(\mathbf{x},y)}(G_{Q_\mathbf{w}}) = \Phi\left(\|\mathbf{w}\|\,\Gamma_\mathbf{w}(\mathbf{x},y)\right) \quad ; \quad \text{where:}$$

$$\Gamma_\mathbf{w}(\mathbf{x},y) \stackrel{\text{def}}{=} \frac{y\mathbf{w}\cdot\phi(\mathbf{x})}{\|\mathbf{w}\|\,\|\phi(\mathbf{x})\|} \quad ; \quad \Phi(a) \stackrel{\text{def}}{=} \frac{1}{\sqrt{2\pi}}\int_a^\infty \exp\left(-\frac{1}{2}x^2\right) dx \,.$$

Gibbs' risk

We need to compute Gibb's risk $R_{(\mathbf{x},y)}(G_{Q_\mathbf{w}})$ on $(\mathbf{x},y)$ since:

$$R_{(\mathbf{x},y)}(G_{Q_\mathbf{w}}) \stackrel{\text{def}}{=} \int_{\mathbb{R}^N} d\mathbf{v}\, Q_\mathbf{w}(\mathbf{v}) I(y\mathbf{v}\cdot\phi(\mathbf{x}) < 0)$$

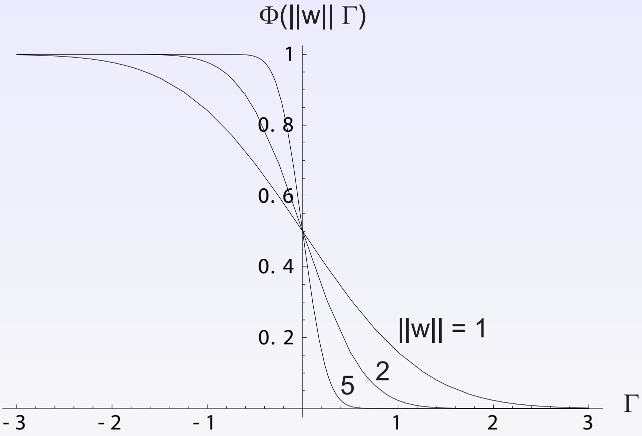
$$R(G_{Q_\mathbf{w}}) = \mathop{\mathbf{E}}_{(\mathbf{x},y)\sim D} R_{(\mathbf{x},y)}(G_{Q_\mathbf{w}}) \quad ; \quad R_S(G_{Q_\mathbf{w}}) = \frac{1}{m}\sum_{i=1}^{m} R_{(\mathbf{x}_i,y_i)}(G_{Q_\mathbf{w}}) \,.$$

As in Langford (2005), the Gaussian integral gives:

$$R_{(\mathbf{x},y)}(G_{Q_\mathbf{w}}) = \Phi\Big(\|\mathbf{w}\|\,\Gamma_\mathbf{w}(\mathbf{x},y)\Big) \quad ; \quad \text{where:}$$

$$\Gamma_\mathbf{w}(\mathbf{x},y) \stackrel{\text{def}}{=} \frac{y\mathbf{w}\cdot\phi(\mathbf{x})}{\|\mathbf{w}\|\,\|\phi(\mathbf{x})\|} \quad ; \quad \Phi(a) \stackrel{\text{def}}{=} \frac{1}{\sqrt{2\pi}}\int_a^\infty \exp\left(-\frac{1}{2}x^2\right) dx \,.$$

Probit loss



Objective function from Corollary 2.1

From Corollary 2.1, we need to find **w** minimizing:

$$\mathcal{B}(S,\mathbf{w},\delta) \stackrel{\text{def}}{=} \sup\left\{\epsilon : \mathrm{kl}(R_S(G_{Q_\mathbf{w}})\|\epsilon) \le \frac{1}{m}\left[\mathrm{KL}(Q_\mathbf{w}\|P_{\mathbf{w}_p}) + \ln\frac{\xi(m)}{\delta}\right]\right\} \quad F(2.1)\,,$$

for a fixed δ (say $\delta = 0.05$). Hence we need to find **w** minimizing $\mathcal{B}$ subject to:

$$\mathrm{kl}\Big(R_S(G_{Q_\mathbf{w}})\Big\|\mathcal{B}\Big) = \frac{1}{m}\left[\mathrm{KL}(Q_\mathbf{w}\|P_{\mathbf{w}_p}) + \ln\frac{\xi(m)}{\delta}\right]$$
$$\mathcal{B} > R_S(G_{Q_\mathbf{w}})\,.$$

Objective function from Corollary 2.1

From Corollary 2.1, we need to find $\mathbf{w}$ minimizing:

$$\mathcal{B}(S, \mathbf{w}, \delta) \stackrel{\text{def}}{=} \sup\left\{\epsilon : \mathrm{kl}(R_S(G_{Q_\mathbf{w}})\|\epsilon) \leq \frac{1}{m}\left[\mathrm{KL}(Q_\mathbf{w}\|P_{\mathbf{w}_p}) + \ln\frac{\xi(m)}{\delta}\right]\right\} \quad F(2.1)\,,$$

for a fixed δ (say $\delta = 0.05$). Hence we need to find $\mathbf{w}$ minimizing $\mathcal{B}$ subject to:

$$\mathrm{kl}\Big(R_S(G_{Q_\mathbf{w}})\Big\|\mathcal{B}\Big) = \frac{1}{m}\left[\mathrm{KL}(Q_\mathbf{w}\|P_{\mathbf{w}_p}) + \ln\frac{\xi(m)}{\delta}\right]$$
$$\mathcal{B} > R_S(G_{Q_\mathbf{w}})\,.$$

Objective function from Corollary 2.2

From Corollary 2.2, for fixed C and $\mathbf{w}_p$, we need to find $\mathbf{w}$ minimizing:

$$CmR_S(G_{Q_\mathbf{w}}) + \mathrm{KL}(Q_\mathbf{w} \| P_{\mathbf{w}_p}) = C\sum_{i=1}^{m} \Phi\left(\frac{y_i \mathbf{w}\cdot\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}\right) + \frac{1}{2}\|\mathbf{w}-\mathbf{w}_p\|^2 \quad (F_{2.2}),$$

We have the same regularizer as the SVM when $\mathbf{w}_p = \mathbf{0}$ (absence of prior knowledge). Indeed, SVM minimizes:

$$C\sum_{i=1}^{m} \max\Big(0, 1 - y_i \mathbf{w}\cdot\phi(\mathbf{x}_i)\Big) + \frac{1}{2}\|\mathbf{w}\|^2,$$

(The probit loss is replaced by the convex hinge loss.)

Objective function from Corollary 2.2

From Corollary 2.2, for fixed C and $\mathbf{w}_p$, we need to find $\mathbf{w}$ minimizing:

$$C m R_S(G_{Q_\mathbf{w}}) + \mathrm{KL}(Q_\mathbf{w} \| P_{\mathbf{w}_p}) = C \sum_{i=1}^{m} \Phi\left(\frac{y_i \mathbf{w} \cdot \phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|} \right) + \frac{1}{2} \|\mathbf{w} - \mathbf{w}_p\|^2 \quad (F_{2.2}),$$

We have the same regularizer as the SVM when $\mathbf{w}_p = \mathbf{0}$ (absence of prior knowledge). Indeed, SVM minimizes:

$$C \sum_{i=1}^{m} \max\Big(0, 1 - y_i \mathbf{w} \cdot \phi(\mathbf{x}_i)\Big) + \frac{1}{2} \|\mathbf{w}\|^2,$$

(The probit loss is replaced by the convex hinge loss.)

Gradient of the PAC-Bayes bound

- We performed the Polak-Ribière conjugate gradient descent (GSL implementation).
- For this, we only need to compute the gradient $\nabla_{\mathbf{w}}\mathcal{B}$.

For $F_{2.1}$ and $F_{2.2}$, we find that $\nabla_{\mathbf{w}}\mathcal{B}$ is given, respectively, by:

$$\frac{1}{m}\frac{\mathcal{B}(1-\mathcal{B})}{\mathcal{B}-R_S}\left[\mathbf{w}-\mathbf{w}_p+\ln\left(\frac{\mathcal{B}(1-R_S)}{R_S(1-\mathcal{B})}\right)\cdot \sum_{i=1}^{m}\Phi'\left(\frac{y_i\mathbf{w}\cdot\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}\right)\frac{y_i\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}\right] \quad (\text{for } F_{2.1})$$

$$C\sum_{i=1}^{m}\Phi'\left(\frac{y_i\mathbf{w}\cdot\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}\right)\frac{y_i\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}+(\mathbf{w}-\mathbf{w}_p) \quad (\text{for } F_{2.2}).$$

Gradient of the PAC-Bayes bound

- We performed the Polak-Ribière conjugate gradient descent (GSL implementation).
- For this, we only need to compute the gradient $\nabla_{\mathbf{w}}\mathcal{B}$.

For $F_{2.1}$ and $F_{2.2}$, we find that $\nabla_{\mathbf{w}}\mathcal{B}$ is given, respectively, by:

$$\frac{1}{m}\,\frac{\mathcal{B}(1-\mathcal{B})}{\mathcal{B}-R_S}\left[\mathbf{w}-\mathbf{w}_p+\ln\left(\frac{\mathcal{B}(1-R_S)}{R_S(1-\mathcal{B})}\right)\cdot \sum_{i=1}^{m}\Phi'\left(\frac{y_i\mathbf{w}\cdot\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}\right)\frac{y_i\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}\right] \quad (\text{for } F_{2.1})$$

$$C\sum_{i=1}^{m}\Phi'\left(\frac{y_i\mathbf{w}\cdot\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}\right)\frac{y_i\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}+(\mathbf{w}-\mathbf{w}_p) \quad (\text{for } F_{2.2})\,.$$

Gradient of the PAC-Bayes bound

- We performed the Polak-Ribière conjugate gradient descent (GSL implementation).
- For this, we only need to compute the gradient $\nabla_{\mathbf{w}}\mathcal{B}$.

For $F_{2.1}$ and $F_{2.2}$, we find that $\nabla_{\mathbf{w}}\mathcal{B}$ is given, respectively, by:

$$\frac{1}{m}\frac{\mathcal{B}(1-\mathcal{B})}{\mathcal{B}-R_S}\left[\mathbf{w}-\mathbf{w}_p+\ln\left(\frac{\mathcal{B}(1-R_S)}{R_S(1-\mathcal{B})}\right)\cdot \sum_{i=1}^{m}\Phi'\left(\frac{y_i\mathbf{w}\cdot\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}\right)\frac{y_i\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}\right] \text{ (for } F_{2.1}\text{)}$$

$$C\sum_{i=1}^{m}\Phi'\left(\frac{y_i\mathbf{w}\cdot\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}\right)\frac{y_i\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}+(\mathbf{w}-\mathbf{w}_p) \text{ (for } F_{2.2}\text{)}.$$

Gradient of the PAC-Bayes bound

- We performed the Polak-Ribière conjugate gradient descent (GSL implementation).
- For this, we only need to compute the gradient $\nabla_{\mathbf{w}}\mathcal{B}$.

For $F_{2.1}$ and $F_{2.2}$, we find that $\nabla_{\mathbf{w}}\mathcal{B}$ is given, respectively, by:

$$\frac{1}{m}\frac{\mathcal{B}(1-\mathcal{B})}{\mathcal{B}-R_S}\left[\mathbf{w}-\mathbf{w}_p+\ln\left(\frac{\mathcal{B}(1-R_S)}{R_S(1-\mathcal{B})}\right)\cdot \sum_{i=1}^{m}\Phi'\left(\frac{y_i\mathbf{w}\cdot\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}\right)\frac{y_i\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}\right] \quad (\text{for } F_{2.1})$$

$$C\sum_{i=1}^{m}\Phi'\left(\frac{y_i\mathbf{w}\cdot\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}\right)\frac{y_i\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}+(\mathbf{w}-\mathbf{w}_p) \quad (\text{for } F_{2.2})\,.$$

Gradient of the PAC-Bayes bound

- We performed the Polak-Ribière conjugate gradient descent (GSL implementation).
- For this, we only need to compute the gradient $\nabla_{\mathbf{w}}\mathcal{B}$.

For $F_{2.1}$ and $F_{2.2}$, we find that $\nabla_{\mathbf{w}}\mathcal{B}$ is given, respectively, by:

$$\frac{1}{m}\frac{\mathcal{B}(1-\mathcal{B})}{\mathcal{B}-R_S}\left[\mathbf{w}-\mathbf{w}_p+\ln\left(\frac{\mathcal{B}(1-R_S)}{R_S(1-\mathcal{B})}\right)\cdot \sum_{i=1}^{m}\Phi'\left(\frac{y_i\mathbf{w}\cdot\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}\right)\frac{y_i\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}\right] \text{ (for } F_{2.1})$$

$$C\sum_{i=1}^{m}\Phi'\left(\frac{y_i\mathbf{w}\cdot\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}\right)\frac{y_i\phi(\mathbf{x}_i)}{\|\phi(\mathbf{x}_i)\|}+(\mathbf{w}-\mathbf{w}_p) \text{ (for } F_{2.2}).$$

Why gradient descent?

- Does there exists a more efficient learning algorithm?
- The probit loss $\Phi(a)$ is quasi-convex in a. **(Nice!)**
- The sum of two quasi-convex functions is (generally) not quasi-convex.
- Hence, $R_S(G_{Q_\mathbf{w}})$ is (generally) not quasi-convex. **(Not nice)**
- $\mathcal{B}(S, \mathbf{w}, \delta)$ does have several local minima on some data sets. This is especially true for $F_{2.2}$ when C is large.
- Each function minimization of $F_{2.2}$ consisted of k different gradient-descent runs. (We used $k = 10$ for $C \leq 10$, and $k = 100$ for $C > 10$.)

Why gradient descent?

- Does there exists a more efficient learning algorithm?
- The probit loss $\Phi(a)$ is quasi-convex in a. **(Nice!)**
- The sum of two quasi-convex functions is (generally) not quasi-convex.
- Hence, $R_S(G_{Q_{\mathbf{w}}})$ is (generally) not quasi-convex. **(Not nice)**
- $\mathcal{B}(S, \mathbf{w}, \delta)$ does have several local minima on some data sets. This is especially true for $F_{2.2}$ when C is large.
- Each function minimization of $F_{2.2}$ consisted of k different gradient-descent runs. (We used $k = 10$ for $C \leq 10$, and $k = 100$ for $C > 10$.)

Why gradient descent?

- Does there exists a more efficient learning algorithm?
- The probit loss $\Phi(a)$ is quasi-convex in a. **(Nice!)**
- The sum of two quasi-convex functions is (generally) not quasi-convex.
- Hence, $R_S(G_{Q_\mathbf{w}})$ is (generally) not quasi-convex. **(Not nice)**
- $\mathcal{B}(S, \mathbf{w}, \delta)$ does have several local minima on some data sets. This is especially true for $F_{2.2}$ when C is large.
- Each function minimization of $F_{2.2}$ consisted of k different gradient-descent runs. (We used $k = 10$ for $C \leq 10$, and $k = 100$ for $C > 10$.)

Why gradient descent?

- Does there exists a more efficient learning algorithm?
- The probit loss $\Phi(a)$ is quasi-convex in a. **(Nice!)**
- The sum of two quasi-convex functions is (generally) not quasi-convex.
- Hence, $R_S(G_{Q_{\mathbf{w}}})$ is (generally) not quasi-convex. **(Not nice)**
- $\mathcal{B}(S, \mathbf{w}, \delta)$ does have several local minima on some data sets. This is especially true for $F_{2.2}$ when C is large.
- Each function minimization of $F_{2.2}$ consisted of k different gradient-descent runs. (We used $k = 10$ for $C \leq 10$, and $k = 100$ for $C > 10$.)

Why gradient descent?

- Does there exists a more efficient learning algorithm?
- The probit loss $\Phi(a)$ is quasi-convex in a. **(Nice!)**
- The sum of two quasi-convex functions is (generally) not quasi-convex.
- Hence, $R_S(G_{Q_\mathbf{w}})$ is (generally) not quasi-convex. **(Not nice)**
- $\mathcal{B}(S, \mathbf{w}, \delta)$ does have several local minima on some data sets. This is especially true for $F_{2.2}$ when C is large.
- Each function minimization of $F_{2.2}$ consisted of k different gradient-descent runs. (We used $k = 10$ for $C \leq 10$, and $k = 100$ for $C > 10$.)

Why gradient descent?

- Does there exists a more efficient learning algorithm?
- The probit loss $\Phi(a)$ is quasi-convex in a. **(Nice!)**
- The sum of two quasi-convex functions is (generally) not quasi-convex.
- Hence, $R_S(G_{Q_\mathbf{w}})$ is (generally) not quasi-convex. **(Not nice)**
- $\mathcal{B}(S, \mathbf{w}, \delta)$ does have several local minima on some data sets. This is especially true for $F_{2.2}$ when C is large.
- Each function minimization of $F_{2.2}$ consisted of k different gradient-descent runs. (We used $k = 10$ for $C \leq 10$, and $k = 100$ for $C > 10$.)

Primal and Dual Versions

- Each proposed algorithm has a primal ($\{w_1, \ldots, w_N\}$) and a dual ($\{\alpha_1, \ldots, \alpha_m\}$) version

$$\mathbf{w} \;=\; \sum_{i=1}^{m} \alpha_i y_i \phi(\mathbf{x}_i) \quad ; \quad k(\mathbf{x}, \mathbf{x}') \;=\; \phi(\mathbf{x}) \cdot \phi(\mathbf{x}')$$

- Decision stumps was used for the primal versions (and compared with AdaBoost).
- The RBF kernel was used for the dual versions (and compared with the soft-margin SVM).

$$k(\mathbf{x}, \mathbf{x}') \;=\; \exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2/\sigma^2\right) .$$

Primal and Dual Versions

- Each proposed algorithm has a primal ($\{w_1, \ldots, w_N\}$) and a dual ($\{\alpha_1, \ldots, \alpha_m\}$) version

$$\mathbf{w} \;=\; \sum_{i=1}^{m} \alpha_i y_i \phi(\mathbf{x}_i) \quad ; \quad k(\mathbf{x}, \mathbf{x}') \;=\; \phi(\mathbf{x}) \cdot \phi(\mathbf{x}')$$

- Decision stumps was used for the primal versions (and compared with AdaBoost).
- The RBF kernel was used for the dual versions (and compared with the soft-margin SVM).

$$k(\mathbf{x}, \mathbf{x}') \;=\; \exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2/\sigma^2\right) .$$

Primal and Dual Versions

- Each proposed algorithm has a primal ($\{w_1, \ldots, w_N\}$) and a dual ($\{\alpha_1, \ldots, \alpha_m\}$) version

$$\mathbf{w} \;=\; \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i) \quad ; \quad k(\mathbf{x}, \mathbf{x}') \;=\; \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}')$$

- Decision stumps was used for the primal versions (and compared with AdaBoost).
- The RBF kernel was used for the dual versions (and compared with the soft-margin SVM).

$$k(\mathbf{x}, \mathbf{x}') \;=\; \exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2/\sigma^2\right) .$$

Three Learning Algorithms

- **PBGD1** uses $P_{\mathbf{0}}$ (*i.e.*, $\mathbf{w}_p = \mathbf{0}$) to learn $Q_{\mathbf{w}}$ by minimizing $F_{2.1}$ (with $\delta = .05$).
- **PBGD3** minimizes $F_{2.2}$ (with $P_{\mathbf{0}}$) but uses 10-fold CV to choose the value of C.
- **PBGD2** uses half of the training data to "learn a good prior".
  - Minimize $F_{2.2}$ (with $P_{\mathbf{0}}$) on first half of training data for $C \in \{10^k : k = 0, \ldots, 6\}$. This gives $\{\mathbf{w}_0, \ldots, \mathbf{w}_6\}$.
  - Minimize $F_{2.1}$ with $P_{\mathbf{w}_0}, \ldots, P_{\mathbf{w}_6}$ on the other half of examples and keep the final classifier that minimizes Corollary 2.1.
- **REMARK: PBGD1** and **PBGD2** are true bound-minimization algorithms (but not **PBGD3**).

Three Learning Algorithms

- **PBGD1** uses $P_{\mathbf{0}}$ (*i.e.*, $\mathbf{w}_p = \mathbf{0}$) to learn $Q_{\mathbf{w}}$ by minimizing $F_{2.1}$ (with $\delta = .05$).
- **PBGD3** minimizes $F_{2.2}$ (with $P_{\mathbf{0}}$) but uses 10-fold CV to choose the value of C.
- **PBGD2** uses half of the training data to "learn a good prior".
  - Minimize $F_{2.2}$ (with $P_{\mathbf{0}}$) on first half of training data for $C \in \{10^k : k = 0, \ldots, 6\}$. This gives $\{\mathbf{w}_0, \ldots, \mathbf{w}_6\}$.
  - Minimize $F_{2.1}$ with $P_{\mathbf{w}_0}, \ldots, P_{\mathbf{w}_6}$ on the other half of examples and keep the final classifier that minimizes Corollary 2.1.
- **REMARK: PBGD1** and **PBGD2** are true bound-minimization algorithms (but not **PBGD3**).

Three Learning Algorithms

- **PBGD1** uses $P_{\mathbf{0}}$ (*i.e.*, $\mathbf{w}_p = \mathbf{0}$) to learn $Q_{\mathbf{w}}$ by minimizing $F_{2.1}$ (with $\delta = .05$).
- **PBGD3** minimizes $F_{2.2}$ (with $P_{\mathbf{0}}$) but uses 10-fold CV to choose the value of C.
- **PBGD2** uses half of the training data to "learn a good prior".
  - Minimize $F_{2.2}$ (with $P_{\mathbf{0}}$) on first half of training data for $C \in \{10^k : k = 0, \ldots, 6\}$. This gives $\{\mathbf{w}_0, \ldots, \mathbf{w}_6\}$.
  - Minimize $F_{2.1}$ with $P_{\mathbf{w}_0}, \ldots, P_{\mathbf{w}_6}$ on the other half of examples and keep the final classifier that minimizes Corollary 2.1.
- **REMARK: PBGD1** and **PBGD2** are true bound-minimization algorithms (but not **PBGD3**).

Three Learning Algorithms

- **PBGD1** uses $P_{\mathbf{0}}$ (*i.e.*, $\mathbf{w}_p = \mathbf{0}$) to learn $Q_{\mathbf{w}}$ by minimizing $F_{2.1}$ (with $\delta = .05$).
- **PBGD3** minimizes $F_{2.2}$ (with $P_{\mathbf{0}}$) but uses 10-fold CV to choose the value of C.
- **PBGD2** uses half of the training data to "learn a good prior".
  - Minimize $F_{2.2}$ (with $P_{\mathbf{0}}$) on first half of training data for $C \in \{10^k : k = 0, \ldots, 6\}$. This gives $\{\mathbf{w}_0, \ldots, \mathbf{w}_6\}$.
  - Minimize $F_{2.1}$ with $P_{\mathbf{w}_0}, \ldots, P_{\mathbf{w}_6}$ on the other half of examples and keep the final classifier that minimizes Corollary 2.1.
- **REMARK: PBGD1** and **PBGD2** are true bound-minimization algorithms (but not **PBGD3**).

Three Learning Algorithms

- **PBGD1** uses $P_{\mathbf{0}}$ (*i.e.*, $\mathbf{w}_p = \mathbf{0}$) to learn $Q_{\mathbf{w}}$ by minimizing $F_{2.1}$ (with $\delta = .05$).
- **PBGD3** minimizes $F_{2.2}$ (with $P_{\mathbf{0}}$) but uses 10-fold CV to choose the value of C.
- **PBGD2** uses half of the training data to "learn a good prior".
  - Minimize $F_{2.2}$ (with $P_{\mathbf{0}}$) on first half of training data for $C \in \{10^k : k = 0, \ldots, 6\}$. This gives $\{\mathbf{w}_0, \ldots, \mathbf{w}_6\}$.
  - Minimize $F_{2.1}$ with $P_{\mathbf{w}_0}, \ldots, P_{\mathbf{w}_6}$ on the other half of examples and keep the final classifier that minimizes Corollary 2.1.
- **REMARK: PBGD1** and **PBGD2** are true bound-minimization algorithms (but not **PBGD3**).

Three Learning Algorithms

- **PBGD1** uses $P_{\mathbf{0}}$ (*i.e.*, $\mathbf{w}_p = \mathbf{0}$) to learn $Q_{\mathbf{w}}$ by minimizing $F_{2.1}$ (with $\delta = .05$).
- **PBGD3** minimizes $F_{2.2}$ (with $P_{\mathbf{0}}$) but uses 10-fold CV to choose the value of C.
- **PBGD2** uses half of the training data to "learn a good prior".
  - Minimize $F_{2.2}$ (with $P_{\mathbf{0}}$) on first half of training data for $C \in \{10^k : k = 0, \ldots, 6\}$. This gives $\{\mathbf{w}_0, \ldots, \mathbf{w}_6\}$.
  - Minimize $F_{2.1}$ with $P_{\mathbf{w}_0}, \ldots, P_{\mathbf{w}_6}$ on the other half of examples and keep the final classifier that minimizes Corollary 2.1.
- **REMARK: PBGD1** and **PBGD2** are true bound-minimization algorithms (but not **PBGD3**).

Experimental Methodology

- Extensive results on UCI and MNIST data sets.
- About half of the data was used for training and half for testing (except for MNIST and Letters where more than 65% was used for testing).
- The binomial tail inversion test set method (Langford, JMLR 2005) was used to determine statistical significance: see the SSB column in the next tables.

Experimental Methodology

- Extensive results on UCI and MNIST data sets.
- About half of the data was used for training and half for testing (except for MNIST and Letters where more than 65% was used for testing).
- The binomial tail inversion test set method (Langford, JMLR 2005) was used to determine statistical significance: see the SSB column in the next tables.

Experimental Methodology

- Extensive results on UCI and MNIST data sets.
- About half of the data was used for training and half for testing (except for MNIST and Letters where more than 65% was used for testing).
- The binomial tail inversion test set method (Langford, JMLR 2005) was used to determine statistical significance: see the SSB column in the next tables.

Dataset	(A) AdaBoost		(1) PBGD1		(2) PBGD2		(3) PBGD3		SSB
Name	Risk	Bound	Risk	Bound	Risk	Bound	Risk	Bound	
Usvotes	**0.055**	0.346	0.085	0.207	0.060	0.165	0.060	0.261	
Credit-A	0.170	0.504	0.177	0.375	0.187	0.272	**0.143**	0.420	
Glass	0.178	0.636	0.196	0.562	0.168	0.395	**0.150**	0.581	
Haberman	**0.260**	0.590	0.273	0.422	0.267	0.465	0.273	0.424	
Heart	0.259	0.569	**0.170**	0.461	0.190	0.379	0.184	0.473	
Sonar	0.231	0.644	0.269	0.579	0.173	0.547	**0.125**	0.622	
BreastCancer	0.053	0.295	**0.041**	0.129	0.047	0.104	0.044	0.190	
Tic-tac-toe	0.357	0.483	0.294	0.462	**0.207**	0.302	**0.207**	0.474	2,3<A,1
Ionosphere	0.120	0.602	0.120	0.425	0.109	0.347	**0.103**	0.557	
Wdbc	0.049	0.447	0.042	0.272	0.049	0.147	**0.035**	0.319	
MNIST:0vs8	0.008	0.528	0.015	0.191	0.011	0.062	**0.006**	0.262	
MNIST:1vs7	**0.013**	0.541	0.020	0.184	0.015	0.050	0.016	0.233	
MNIST:1vs8	0.025	0.552	0.037	0.247	0.027	0.087	**0.018**	0.305	3<1
MNIST:2vs3	0.047	0.558	0.046	0.264	0.040	0.105	**0.034**	0.356	
Letter:AvsB	0.010	0.254	0.009	0.180	**0.007**	0.065	**0.007**	0.180	
Letter:DvsO	0.036	0.378	0.043	0.314	0.033	0.090	**0.024**	0.360	
Letter:OvsQ	**0.038**	0.431	0.061	0.357	0.053	0.106	0.042	0.454	
Adult	**0.149**	0.394	0.168	0.270	0.169	0.209	0.159	0.364	A<1,2
Mushroom	**0.000**	0.200	0.046	0.130	0.016	0.030	0.002	0.150	A,3<2<1

Dataset	(S) SVM		(1) PBGD1		(2) PBGD2		(3) PBGD3		SSB
Name	Risk	Bound	Risk	Bound	Risk	Bound	Risk	Bound	
Usvotes	0.055	0.370	0.080	0.244	**0.050**	0.153	0.075	0.332	
Credit-A	0.183	0.591	**0.150**	0.341	**0.150**	0.248	0.160	0.375	
Glass	0.178	0.571	**0.168**	0.539	0.215	0.430	**0.168**	0.541	
Haberman	0.280	0.423	0.280	0.417	0.327	0.444	**0.253**	0.555	
Heart	0.197	0.513	0.190	0.441	**0.184**	0.400	0.197	0.520	
Sonar	0.163	0.599	0.250	0.560	0.173	0.477	**0.144**	0.585	
BreastCancer	**0.038**	0.146	0.044	0.132	0.041	0.101	0.047	0.162	
Tic-tac-toe	0.081	0.555	0.365	0.426	0.173	0.287	**0.077**	0.548	$S,3<2<1$
Ionosphere	0.097	0.531	0.114	0.395	0.103	0.376	**0.091**	0.465	
Wdbc	0.074	0.400	0.074	0.366	**0.067**	0.298	0.074	0.367	
MNIST:0vs8	**0.003**	0.257	0.009	0.202	0.007	0.058	0.004	0.320	
MNIST:1vs7	0.011	0.216	0.014	0.161	**0.009**	0.052	0.010	0.250	
MNIST:1vs8	0.011	0.306	0.014	0.204	0.011	0.060	**0.010**	0.291	
MNIST:2vs3	**0.020**	0.348	0.038	0.265	0.028	0.096	0.023	0.326	$S<1$
Letter:AvsB	**0.001**	0.491	0.005	0.170	0.003	0.064	**0.001**	0.485	
Letter:DvsO	0.014	0.395	0.017	0.267	0.024	0.086	**0.013**	0.350	
Letter:OvsQ	0.015	0.332	0.029	0.299	0.019	0.078	**0.014**	0.329	
Adult	**0.159**	0.535	0.173	0.274	0.180	0.224	0.164	0.372	$S,3<2$
Mushroom	**0.000**	0.213	0.007	0.119	0.001	0.011	**0.000**	0.167	$S,2,3<1$

Conclusion

- **PBGD2** is better than **PBGD1** (for R_T and bound).
  - Using half of the data to learn a prior helps.
- **PBGD3** is a bit better than **PBGD2**.
  - The bound is a bit worse than CV for finding quantitatively the right trade-off between $R_S(G_Q)$ and $\mathrm{KL}(Q\|P)$.
- **PBGD3** is a little bit better than AdaBoost and SVM (perhaps) but it is much **slower** (several local minima).
  - In practice, I would still use the SVM instead of **PBGD3**.
- **PBGD2** is the best choice if obtaining a good guarantee on the true risk is mandatory.

Conclusion

- **PBGD2** is better than **PBGD1** (for R_T and bound).
  - Using half of the data to learn a prior helps.
- **PBGD3** is a bit better than **PBGD2**.
  - The bound is a bit worse than CV for finding quantitatively the right trade-off between $R_S(G_Q)$ and $\mathrm{KL}(Q\|P)$.
- **PBGD3** is a little bit better than AdaBoost and SVM (perhaps) but it is much **slower** (several local minima).
  - In practice, I would still use the SVM instead of **PBGD3**.
- **PBGD2** is the best choice if obtaining a good guarantee on the true risk is mandatory.

Conclusion

- **PBGD2** is better than **PBGD1** (for R_T and bound).
  - Using half of the data to learn a prior helps.
- **PBGD3** is a bit better than **PBGD2**.
  - The bound is a bit worse than CV for finding quantitatively the right trade-off between $R_S(G_Q)$ and $\mathrm{KL}(Q\|P)$.
- **PBGD3** is a little bit better than AdaBoost and SVM (perhaps) but it is much **slower** (several local minima).
  - In practice, I would still use the SVM instead of **PBGD3**.
- **PBGD2** is the best choice if obtaining a good guarantee on the true risk is mandatory.

Conclusion

- **PBGD2** is better than **PBGD1** (for R_T and bound).
  - Using half of the data to learn a prior helps.
- **PBGD3** is a bit better than **PBGD2**.
  - The bound is a bit worse than CV for finding quantitatively the right trade-off between $R_S(G_Q)$ and $\mathrm{KL}(Q\|P)$.
- **PBGD3** is a little bit better than AdaBoost and SVM (perhaps) but it is much **slower** (several local minima).
  - In practice, I would still use the SVM instead of **PBGD3**.
- **PBGD2** is the best choice if obtaining a good guarantee on the true risk is mandatory.

Conclusion

- **PBGD2** is better than **PBGD1** (for R_T and bound).
  - Using half of the data to learn a prior helps.
- **PBGD3** is a bit better than **PBGD2**.
  - The bound is a bit worse than CV for finding quantitatively the right trade-off between $R_S(G_Q)$ and $\mathrm{KL}(Q\|P)$.
- **PBGD3** is a little bit better than AdaBoost and SVM (perhaps) but it is much **slower** (several local minima).
  - In practice, I would still use the SVM instead of **PBGD3**.
- **PBGD2** is the best choice if obtaining a good guarantee on the true risk is mandatory.

Conclusion

- **PBGD2** is better than **PBGD1** (for R_T and bound).
  - Using half of the data to learn a prior helps.
- **PBGD3** is a bit better than **PBGD2**.
  - The bound is a bit worse than CV for finding quantitatively the right trade-off between $R_S(G_Q)$ and $\mathrm{KL}(Q\|P)$.
- **PBGD3** is a little bit better than AdaBoost and SVM (perhaps) but it is much **slower** (several local minima).
  - In practice, I would still use the SVM instead of **PBGD3**.
- **PBGD2** is the best choice if obtaining a good guarantee on the true risk is mandatory.

Conclusion

- **PBGD2** is better than **PBGD1** (for R_T and bound).
  - Using half of the data to learn a prior helps.
- **PBGD3** is a bit better than **PBGD2**.
  - The bound is a bit worse than CV for finding quantitatively the right trade-off between $R_S(G_Q)$ and $\mathrm{KL}(Q\|P)$.
- **PBGD3** is a little bit better than AdaBoost and SVM (perhaps) but it is much **slower** (several local minima).
  - In practice, I would still use the SVM instead of **PBGD3**.
- **PBGD2** is the best choice if obtaining a good guarantee on the true risk is mandatory.