<table>
<thead>
<tr>
<th>Date</th>
<th>Day</th>
<th>no.</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct 08</td>
<td>Mon</td>
<td>1</td>
<td>A Hands-On Introduction</td>
</tr>
<tr>
<td>Oct 10</td>
<td>Wed</td>
<td>–</td>
<td>self-study (Christoph traveling)</td>
</tr>
<tr>
<td>Oct 15</td>
<td>Mon</td>
<td>2</td>
<td>Bayesian Decision Theory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Generative Probabilistic Models</td>
</tr>
<tr>
<td>Oct 17</td>
<td>Wed</td>
<td>3</td>
<td>Discriminative Probabilistic Models</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Maximum Margin Classifiers</td>
</tr>
<tr>
<td>Oct 22</td>
<td>Mon</td>
<td>4</td>
<td>Generalized Linear Classifiers, Optimization</td>
</tr>
<tr>
<td>Oct 24</td>
<td>Wed</td>
<td>5</td>
<td>Evaluating Predictors; Model Selection</td>
</tr>
<tr>
<td>Oct 29</td>
<td>Mon</td>
<td>–</td>
<td>self-study (Christoph traveling)</td>
</tr>
<tr>
<td>Oct 31</td>
<td>Wed</td>
<td>6</td>
<td>Overfitting/Underfitting, Regularization</td>
</tr>
<tr>
<td>Nov 05</td>
<td>Mon</td>
<td>7</td>
<td>Learning Theory I: classical/Rademacher bounds</td>
</tr>
<tr>
<td>Nov 07</td>
<td>Wed</td>
<td>8</td>
<td>Learning Theory II: miscellaneous</td>
</tr>
<tr>
<td>Nov 12</td>
<td>Mon</td>
<td>9</td>
<td>Probabilistic Graphical Models I</td>
</tr>
<tr>
<td>Nov 14</td>
<td>Wed</td>
<td>10</td>
<td>Probabilistic Graphical Models II</td>
</tr>
<tr>
<td>Nov 19</td>
<td>Mon</td>
<td>11</td>
<td>Probabilistic Graphical Models III</td>
</tr>
<tr>
<td>Nov 21</td>
<td>Wed</td>
<td>12</td>
<td>Probabilistic Graphical Models IV</td>
</tr>
<tr>
<td>until Nov 25</td>
<td></td>
<td></td>
<td>final project</td>
</tr>
</tbody>
</table>
Beyond complexity measures
Algorithm-dependent bounds

Generalization bounds so far: with probability at least $1 - \delta$:

$$\forall f \in \mathcal{H} : \quad \mathcal{R}(f) \leq \hat{\mathcal{R}}(f) + "something"$$

Observation:
- holds simultaneous for all hypotheses in $\mathcal{H}$, we can pick any we like
- but: in practice, we have some algorithm that choses the hypothesis
  and really only need the result for that
Algorithm-dependent bounds

Generalization bounds so far: \textit{with probability at least $1 - \delta$:}

$$\forall f \in \mathcal{H} : \ R(f) \leq \hat{R}(f) + \text{"something"}$$

Observation:

- holds simultaneous for all hypotheses in $\mathcal{H}$, we can pick any we like
- but: in practice, we have some algorithm that choses the hypothesis and really only need the result for that

Goal: algorithm-dependent bounds

Instead of

- "For which hypothesis sets does learning not overfit?"

ask

- "Which learning algorithms do not overfit?"
- \(\mathcal{Z}\): input set (typically \(\mathcal{Z} = \mathcal{X} \times \mathcal{Y}\))
- \(\mathcal{H}\): set of hypotheses
- \(L(h, z)\): loss function of the form \(L(h, z) = \ell(y, f(x))\)

**Definition (Learning algorithm)**

A **learning algorithm**, \(A\), is a function that takes as input a finite subset, \(D_m \subset \mathcal{Z}\), and outputs a hypothesis \(A[D] \in \mathcal{H}\).
• $\mathcal{Z}$: input set (typically $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$)
• $\mathcal{H}$: set of hypotheses
• $L(h, z)$: loss function of the form $L(h, z) = \ell(y, f(x))$

**Definition (Learning algorithm)**

A **learning algorithm**, $A$, is a function that takes as input a finite subset, $\mathcal{D}_m \subset \mathcal{Z}$, and outputs a hypothesis $A[\mathcal{D}] \in \mathcal{H}$.

**Definition (Uniform stability)**

For a training set, $\mathcal{D} = \{z_1, \ldots, z_m\}$, we call the training set with the $i$-th element removed $\mathcal{D}^{\backslash i} = \{z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_m\}$.

A learning algorithm, $A$, has **uniform stability** $\beta$ with respect to the loss $\ell$ if the following holds,

$$\forall \mathcal{D}_m \subset \mathcal{Z} \forall i \in \{1, 2, \ldots, m\} \quad \|L(A[\mathcal{D}], \cdot) - L(A[\mathcal{D}^{\backslash i}], \cdot)\|_\infty \leq \beta$$

For a uniformly stable algorithm, changing the training set a little has only a small effect.
Theorem (Stable algorithms generalize well [Bousquet et al., 2002])

Let $A$ be a $\beta$-uniformly stable learning algorithm. For a training set $\mathcal{D}$ that consists of $m$ i.i.d. samples, denote by $f = A[\mathcal{D}]$ be the output of $A$ on $\mathcal{D}$. Let $\ell(y, \bar{y})$ be bounded by $M$.

Then, for any $\delta > 0$, with probability at least $1 - \delta$,

$$ R(f) \leq \hat{R}(f) + 2\beta + (4m\beta + M)\sqrt{\frac{\log(1/\delta)}{2m}} $$

Bound is useful, if stability $\beta$ behaves (at least) like $\frac{1}{m}$. 
Stochastic gradient descent (SGD): minimize a function

\[ f(\theta) = \frac{1}{m} \sum_{i=1}^{m} f(\theta; z_i) \]

**Theorem (Stability of Stochastic Gradient Descent [Hardt et al., 2016])**

Let \( f(\cdot, z) \) be \( \gamma \)-smooth, convex and \( L \)-Lipschitz for every \( z \). Suppose that we run SGD with step sizes \( \alpha_t \leq 2/\gamma \) for \( T \) steps. Then, SGD satisfies uniform stability with

\[ \beta \leq \frac{2L^2}{m} \sum_{t=1}^{T} \alpha_t. \]

Let \( f(\cdot, z) \) be \( \gamma \)-smooth and \( L \)-Lipschitz, but not necessarily convex. Assume we run SGD with monotonically non-increasing step sizes \( \alpha_t \leq c/t \) for some \( c \). Then, SGD satisfies uniform stability with

\[ \beta \leq \frac{1 + \frac{1}{\gamma c}}{m - 1} \left( 2cL^2 \right) \frac{1}{\gamma c + 1} T \frac{\gamma c}{\gamma c + 1}. \]
The power of compression
Reminder:

Perceptron – Training

**input** training set $\mathcal{D} \subset \mathbb{R}^d \times \{-1, +1\}$
initialize $w = (0, \ldots, 0) \in \mathbb{R}^d$.

repeat
  for all $(x, y) \in \mathcal{D}$: do
    compute $a := \langle w, x \rangle$ ('activation')
    if $ya \leq 0$ then
      $w \leftarrow w + yx$
    end if
  end for
until $w$ wasn’t updated for a complete pass over $\mathcal{D}$

Let’s assume $\mathcal{D}$ is very large, so we don’t need multiple passes.

Properties:

- sequential training, one pass over data
- only those examples matter, where perceptron made a mistake
  (only those lead to changes of $w$)
Towards Sample Compression Bounds

- Take training set as a sequence:

\[ T = ((x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)) \]

- algorithm \( A \) processes \( T \) in order, producing output \( f := A(T) \)

- What only a subset of examples influence the algorithm output?

- for increasing subsequence, \( I \subset \{1, \ldots, n\} \), with \(|I| = l\), set

\[ T_I = ((x^{i_1}, y^{i_1}), (x^{i_2}, y^{i_2}), \ldots, (x^{i_l}, y^{i_l})) \]

**Definition**

\( I \) is a **compression set** for \( T \), if \( A(T) = A(T_I) \).

Example: \( I = \{\text{set of examples where Perceptron made a mistake}\} \)
**Definition (Compression scheme [Littlestone/Warmuth, 1986])**

A learning algorithm \( A \) is called **compression scheme**, if there is a pair of functions: \( C \) (called compression function), and \( L \) (called reconstruction function), such that:

- \( C \) takes as input a finite dataset and outputs a subsequence of indices
- \( L \) takes as input a finite dataset and outputs a predictor
- \( A \) is the result of applying \( L \) to the data selected by \( C \)

\[
A = L(T_I) \text{ for } I = C(T)
\]

**Examples:**
- Perceptron (\( I = \) indices of examples where will be updated)
- SVMs (\( I = \) set of support vectors)
- \( k \)-NN (\( I = \) set of examples that support the decision boundaries)
\[
\hat{R}_I(h) = \frac{1}{|I|} \sum_{i \in I} \ell(y^i, h(x^i)) \quad \text{and} \quad \hat{R}_{\neg I}(h) = \frac{1}{n - |I|} \sum_{i \notin I} \ell(y^i, h(x^i))
\]

**Theorem (Compression Bound [Littlestone/Warmuth, 1986; Graepel 2005])**

Let \( A \) be a compression scheme with compression function \( C \). Let the loss \( \ell \) be bounded by \([0, 1]\). Then, with probability at least \( 1 - \delta \) over the random draw of \( T \), we have that:

If \( \hat{R}_{\neg I}(A(T)) = 0 \):

\[
\mathcal{R}(A(T)) \leq \frac{1}{m-l} \left( (l + 1) \log m + \log \frac{1}{\delta} \right).
\]

For general \( \hat{R}_{\neg I}(A(T)) \):

\[
\mathcal{R}(A(T)) \leq \frac{m}{m-l} \hat{R}_{\neg I}(A(T)) + \sqrt{\frac{(l + 2) \log m + \log \frac{1}{\delta}}{2(m-l)}}
\]

where \( I = C(T) \) and \( l = |I| \).
The power of randomization
The problem of overfitting emerges mainly because we pick only a single classifier, \( h \), and just by accident it can have \( \mathcal{R}(h) \gg \hat{\mathcal{R}}(h) \).
If we choose many classifiers and combine their decisions, chances of overfitting should be lower.

**Definition (Majority-vote)**

Let \( \mathcal{Y} = \{\pm 1\} \) (only for convenience of notation). Let \( h_1, \ldots, h_T \in \mathcal{H} \) be a set of hypotheses. We define the **uniform majority vote** classifier as

\[
h_{\text{majority}}(x) = \text{sign} \left( \frac{1}{T} \sum_{i=1}^{T} h_i(x) \right)
\]
Definition (Majority-vote)

More generally, for weights $\alpha_i \in [0, 1]$, $\sum_i \alpha_i = 1$, the $\alpha$-weighted majority vote classifier is:

$$h^{\alpha}_{\text{majority}}(x) = \text{sign} \sum_{i=1}^{T} \alpha_i h_i(x) = \mathbb{E}_{i \sim \alpha} [h_i(x)]$$

Weighting make a convenient framework:

- we can use a base set of many (even countably infinite) classifier
- we assign weights to good classifiers, e.g. based on training data
- classical setting is included: for $\alpha = \delta_{i=j}$: $h^{\alpha}_{\text{majority}} = h_j$
Definition (Majority-vote)

More generally, for weights $\alpha_i \in [0, 1]$, $\sum_i \alpha_i = 1$, the \textbf{$\alpha$-weighted majority vote classifier} is:

$$h_{\alpha\text{majority}}(x) = \text{sign} \sum_{i=1}^{T} \alpha_i h_i(x) = \mathbb{E}_{i \sim \alpha}[h_i(x)]$$

Weighting make a convenient framework:
- we can use a base set of many (even countably infinite) classifier
- we assign weights to \textit{good classifiers}, e.g. based on training data
- classical setting is included: for $\alpha = \delta_{i=j}$: $h_{\alpha\text{majority}} = h_j$

Unfortunately, majority vote classifiers are not easy to classify:
- classical bounds hold equally for \textit{any} $h \in \mathcal{H}$
- if $h_{\alpha\text{majority}} \in \mathcal{H}$, bound no better than for others
- if $h_{\alpha\text{majority}} \not\in \mathcal{H}$, no bound at all

Trick: analyze \textit{stochastic classifiers}
Stochastic Classifiers

Standard scenario:
- \( \mathcal{X} \): input set, \( \mathcal{Y} \): output set, \( p \) probability distribution over \( \mathcal{X} \times \mathcal{Y} \)
- \( \mathcal{H} \subset \{ \mathcal{X} \to \mathcal{Y} \} \): hypothesis set, \( \ell \): loss function
- \( \mathcal{D} = \{ (x^1, y^1), \ldots, (x^n, y^n) \} \overset{i.i.d.}{\sim} p(x, y) \): training set

New:
- \( Q \): probability distribution over \( \mathcal{H} \)

Definition (Gibbs classifier)
For a distribution \( Q \) over \( \mathcal{H} \subset \{ h : \mathcal{X} \to \mathcal{Y} \} \), the Gibbs classifier, \( h_Q \), is defined by the procedure:
  - input: \( x \in \mathcal{X} \)
  - sample \( h \sim Q \)
  - output: \( h(x) \)

The Gibbs classifier is a stochastic classifier, its output is a random variable (wrt \( Q \)).
Stochastic Classifiers

Standard scenario:
- $\mathcal{X}$: input set, $\mathcal{Y}$: output set, $p$ probability distribution over $\mathcal{X} \times \mathcal{Y}$
- $\mathcal{H} \subset \{\mathcal{X} \rightarrow \mathcal{Y}\}$: hypothesis set, $\ell$: loss function
- $\mathcal{D} = \{(x^1, y^1), \ldots, (x^n, y^n)\}$ i.i.d. $\sim p(x, y)$: training set

New:
- $Q$ probability distribution over $\mathcal{H}$

Definition (Gibbs classifier)

For a distribution $Q$ over $\mathcal{H} \subset \{h : \mathcal{X} \rightarrow \mathcal{Y}\}$, the Gibbs classifier, $h_Q$, is defined by the procedure:
- input: $x \in \mathcal{X}$
- sample $h \sim Q$
- output: $h(x)$

The Gibbs classifier is a stochastic classifier, its output is a random variable (wrt $Q$).
Definition (Gibbs classifier)

For a distribution $Q$ over $\mathcal{H} \subset \{ h : \mathcal{X} \to \mathcal{Y} \}$, the **Gibbs classifier**, $h_Q$, is defined by the procedure:

- **input**: $x \in \mathcal{X}$
- **sample**: $h \sim Q$
- **output**: $h(x)$

Because the classifier output is random, so are the risks:

$$
\mathcal{R}(h_Q) = \mathbb{E}_{(x,y) \sim p} \ell(y, h_Q(x)) \quad \quad \hat{\mathcal{R}}(h_Q) = \sum_{i=1}^{n} \ell(y^i, h_Q(x^i))
$$

We can study their expected value:

$$
\mathcal{R}(Q) = \mathbb{E}_{h \sim Q} \mathcal{R}(h) = \mathbb{E}_{h \sim Q} \mathbb{E}_{(x,y) \sim p} \ell(y, h(x)) \quad \quad \hat{\mathcal{R}}(Q) = \mathbb{E}_{h \sim Q} \sum_{i=1}^{n} \ell(y^i, h(x^i))
$$
Learning

- $\mathcal{X}$: input set, $\mathcal{Y}$: output set, $p$ probability distribution over $\mathcal{X} \times \mathcal{Y}$
- $\mathcal{H} \subset \{ \mathcal{X} \rightarrow \mathcal{Y} \}$: hypothesis set, $\ell$: loss function

What’s the analog of deterministic learning?

Given a training set, $\mathcal{D} = \{(x^1, y^1), \ldots, (x^n, y^n)\} \overset{i.i.d.}{\sim} p(x, y)$, identify a distribution $Q$ (arbitrary, or from a parametric family), such that $R(Q)$ is as small as possible.

What would a generalization bound look like?

$$R(Q) \leq \hat{R}(Q) + "something"$$
Gibbs classifier vs. majority vote

**Majority vote classifier:** (now calling weights $Q$ instead of $\alpha$)

- evaluate all classifiers, $h(x)$ for $h \in \mathcal{H}$
- combine their outputs according to their weights, $\mathbb{E}_{h \sim Q} h(x)$
- make one decision based on the result, $\text{sign} \mathbb{E}_{h \sim Q} h(x)$
- evaluate the loss of this decision, $\ell(y, \text{sign} \mathbb{E}_{h \sim Q} h(x))$

**Gibbs classifier:**

- evaluate all classifiers, $h(x)$ for $h \in \mathcal{H}$
- evaluate the loss of all their decisions, $\ell(y, h(x))$ for $h \in \mathcal{H}$
- combine their losses according to their weights, $\mathbb{E}_{h \sim Q} \ell(y, h(x))$

How are the two situations related?
Lemma

\[ R_{\text{majority}}(Q) \leq 2R_{\text{Gibbs}}(Q) \]

Observation:

\[ h_{\text{majority}}^Q(x) = \text{sign} \mathbb{E}_{h \sim Q} h(x) = \begin{cases} +1 & \text{if more than 50\% (probability mass) of the individual classifiers say } +1 \\ -1 & \text{otherwise} \end{cases} \]

\[ \ell(y, h_{\text{majority}}(x)) = 1 \quad \Rightarrow \quad \Pr_{h \sim Q} \{ \ell(y, h(x)) = 1 \} \geq 0.5 \]

\[ \ell(y, h_{\text{majority}}(x)) = 1 \quad \Rightarrow \quad 2 \mathbb{E}_{h \sim Q} [\ell(y, h(x))] \geq 1 \]

\[ 2 \mathbb{E}_{h \sim Q} [\ell(y, h(x))] \geq \ell(y, h_{\text{majority}}(x)) \]

\[ 2R_{\text{Gibbs}}(Q) \geq R_{\text{majority}}(Q) \]

Generalization bounds for \( R_{\text{Gibbs}} \) also hold for \( R_{\text{majority}} \) (up to factor 2).
Example: Generalization bound for Gibbs classifier

Theorem (PAC-Bayesian generalization bound [McAllester, 1999])

Let the loss, $\ell$, be a bounded in $[0, 1]$. Let $P$ be a "prior" distribution of $H$, chosen independently of $D$. With prob $1 - \delta$ over $D \overset{i.i.d.}{\sim} p \otimes n$, it holds for all "posterior" distributions $Q$:

$$R(Q) \leq \hat{R}(Q) + \frac{1}{\sqrt{n}} \left( KL(Q||P) + \frac{1}{8} + \log \frac{1}{\delta} \right)$$

- Called PAC-Bayesian, because it makes a PAC-style statement (different between finite sample and expect error), but for Bayesian-style objects (distributions over classifiers/parameters)
- prior and posterior are in quotation marks, because the posterior is not the result of applying Bayes’ rule.
- The prior is only a technical tool and shows up in the KL term. We don’t have to "believe" in it or anything.
Towards a proof:

**Theorem (Change of Measure Inequality)**

For any distributions $P, Q$ over $\mathcal{H}$ and function $\phi : \mathcal{H} \to \mathbb{R}$:

\[
\mathbb{E}_{h \sim Q} [\phi(h)] \leq \frac{1}{\lambda} \left( KL(Q||P) + \log \mathbb{E}_{h \sim P} e^{\lambda \phi(h)} \right)
\]

with

\[
KL(Q||P) = \mathbb{E}_{h \sim Q} \left[ \log \frac{Q(h)}{P(h)} \right]
\]

We shift from an expectation over $P$ to an expectation over $Q$.

Very useful, e.g.

- $P$ will be a typically a simple, data-independent, distribution
- $Q$ will depend on a training set $\rightarrow$ "trained classifier"
- we "pay" for this: $\mathbb{E}_Q(\cdot)$ turns into $\log \mathbb{E}_P \exp(\cdot)$
Proof sketch, pretending $P$ and $Q$ have densities.

General observation:

$$
\mathbb{E}_{h \sim P}[f(h)] = \int_{\mathcal{H}} P(h) f(h) dh = \int_{\mathcal{H}} Q(h) \frac{P(h)}{Q(h)} f(h) dh = \mathbb{E}_{h \sim Q} \left[ \frac{P(h)}{Q(h)} f(h) \right]
$$

$$
\log \mathbb{E}_{h \sim P}[e^{\lambda \phi(h)}] = \log \mathbb{E}_{h \sim Q} \left[ e^{\lambda \phi(h)} \frac{P(h)}{Q(h)} \right]
$$

Jensen’s ineq.

$$
\geq \mathbb{E}_{h \sim Q} \left[ \log e^{\lambda \phi(h)} \frac{P(h)}{Q(h)} \right]
$$

$$
= \mathbb{E}_{h \sim Q} \left[ \lambda \phi(h) - \log \frac{Q(h)}{P(h)} \right]
$$

$$
= \lambda \mathbb{E}_{h \sim Q} [\phi(h)] - \text{KL}(Q||P)
$$

rearrange, $\cdot \frac{1}{\lambda}$

$$
\Rightarrow \mathbb{E}_{h \sim Q} [\phi(h)] \leq \frac{1}{\lambda} \left( \log \mathbb{E}_{h \sim P} [e^{\lambda \phi(h)}] + \text{KL}(Q||P) \right)
$$
**Theorem (Change of Measure Inequality)**

For any distributions \( P, Q \) over \( \mathcal{H} \) and function \( \phi : \mathcal{H} \to \mathbb{R} \):

\[
\mathbb{E}_{h \sim Q}[\phi(h)] \leq \frac{1}{\lambda} \left( \text{KL}(Q||P) + \log \mathbb{E}_{h \sim P} e^{\lambda \phi(h)} \right)
\]

**Theorem (PAC-Bayesian generalization bound [McAllester, 1999])**

\( \ell \) bounded in \([0, 1]\). \( P \) independent of \( \mathcal{D} \).

With prob \( 1 - \delta \) over \( \mathcal{D} \overset{i.i.d.}{\sim} p \otimes n \), it holds for all distributions \( Q \):

\[
\mathcal{R}(Q) \leq \hat{\mathcal{R}}(Q) + \frac{1}{\sqrt{n}} \left( \text{KL}(Q||P) + \frac{1}{8} + \log \frac{1}{\delta} \right)
\]
Proof sketch.

- Change of measure inequality:
  \[ \mathbb{E}_{h \sim Q} [\phi(h)] \leq \frac{1}{\lambda} \left( \text{KL}(Q \| P) + \log \mathbb{E}_{h \sim P} e^{\lambda \phi(h)} \right) \]

- Apply with prior \( P \), posterior \( Q \) and \( \phi(h) = \mathcal{R}(h) - \hat{\mathcal{R}}(h) \):
  \[ \mathcal{R}(Q) - \hat{\mathcal{R}}(Q) \leq \frac{1}{\lambda} \left( \text{KL}(Q \| P) + \log \mathbb{E}_{h \sim P} e^{\lambda [\mathcal{R}(h) - \hat{\mathcal{R}}(h)]} \right) \]

- \( P \) and \( \phi \) are independent (in contrast to \( Q \)), so with prob. \( \geq 1 - \delta \)
  \[ \log \mathbb{E}_{h \sim P} e^{\lambda [\mathcal{R}(h) - \hat{\mathcal{R}}(h)]} \overset{\text{Hoeffing’s lemma, Markov ineq.}}{\leq} \frac{\lambda^2 n}{8} + \log(1/\delta) \]

- Theorem follows by setting \( \lambda = \frac{1}{n} \).
Example: reproving a bound for finite hypothesis sets

- $\mathcal{H} = \{h_1, \ldots, h_T\}$ finite
- $P(h) = (\frac{1}{T}, \ldots, \frac{1}{T})$ uniform distribution
- $Q(h) = \delta_{h=h_k}(h)$ indicator on one hypothesis
- $KL(Q||P) = \sum_t Q(t) \log \frac{Q(t)}{P(t)} = \log \frac{1}{P(h_k)} = \log T$
Example: reproving a bound for finite hypothesis sets

- $\mathcal{H} = \{h_1, \ldots, h_T\}$ finite
- $P(h) = (\frac{1}{T}, \ldots, \frac{1}{T})$ uniform distribution
- $Q(h) = \delta_{h=h_k}(h)$ indicator on one hypothesis
- $\text{KL}(Q||P) = \sum_t Q(t) \log \frac{Q(t)}{P(t)} = \log \frac{1}{P(h_k)} = \log T$

The PAC-Bayesian statement for Gibbs classifiers:

For every dist. $Q$: $\mathcal{R}(Q) \leq \hat{\mathcal{R}}(Q) + \frac{1}{\sqrt{n}} \left( \text{KL}(Q||P) + \frac{1}{8} + \log \frac{1}{\delta} \right)$

translates into a bound for a ordinary (deterministic) classifiers:

For every $h \in \mathcal{H}$: $\mathcal{R}(h) \leq \hat{\mathcal{R}}(h) + \frac{1}{\sqrt{n}} \left( \log T + \frac{1}{8} + \log \frac{1}{\delta} \right)$

which is similar to the previous bound for finite hypotheses sets.
New: we can freely chose the prior, it does not have to be uniform.

- $\mathcal{H} = \{h_1, \ldots, h_T\}$ finite (or countable infinite)
- $P(h) = (\pi_1, \ldots, \pi_T)$ arbitrary prior distribution (fix before seeing $D$)
- $Q(h) = \delta_{h=h_k}(h)$ indicator on one hypothesis
- $\text{KL}(Q||P) = \sum_t Q(t) \log \frac{Q(t)}{P(t)} = \log \frac{1}{\pi_k}$

For every $h_k \in \mathcal{H}$:

$$\mathcal{R}(h_k) \leq \hat{\mathcal{R}}(h_k) + \frac{1}{\sqrt{n}} \left( \log \frac{1}{\pi_k} + \frac{1}{8} + \log \frac{1}{\delta} \right)$$

Better bound, if well-working hypotheses are (a priori) more likely.
Example: justifying $L^2$-regularization

- $\mathcal{H} = \{ h_w(x) : \mathcal{X} \to \mathcal{Y}, \ w \in \mathbb{R}^d \}$ parameterized by $w \in \mathbb{R}^d$
- $P(w) \propto e^{-\lambda \|w\|^2}$ prior: Gaussian around 0
- $Q(w) \propto e^{-\lambda \|w-v\|^2}$ posterior: Gaussian around $v$
- $\text{KL}(Q \| P) = \lambda \|v\|^2$

\[
\mathcal{R}(Q) \leq \hat{\mathcal{R}}(Q) + \frac{1}{\sqrt{n}} \left( \lambda \|v\|^2 + \frac{1}{8} + \log \frac{1}{\delta} \right)
\]

- most promising classifier: minimize right hand side w.r.t $v$
  -> "regularizer" $\|v\|^2$ appears naturally in the objective
Example: justifying $L^2$-regularization

- $\mathcal{H} = \{ h_w(x) : \mathcal{X} \to \mathcal{Y}, \ w \in \mathbb{R}^d \}$ parameterized by $w \in \mathbb{R}^d$
- $P(w) \propto e^{-\lambda \|w\|^2}$ prior: Gaussian around 0
- $Q(w) \propto e^{-\lambda \|w-v\|^2}$ posterior: Gaussian around $v$
- $\text{KL}(Q||P) = \lambda \|v\|^2$

$$\mathcal{R}(Q) \leq \hat{\mathcal{R}}(Q) + \frac{1}{\sqrt{n}} \left( \lambda \|v\|^2 + \frac{1}{8} + \log \frac{1}{\delta} \right)$$

- most promising classifier: minimize right hand side w.r.t $v$
  $\rightarrow$ "regularizer" $\|v\|^2$ appears naturally in the objective

Caveat: $\| \cdot \|^2$ appears because we put it into the exponents of $P$ and $Q$. Other distributions (which are our choice) yield other bounds/regularizers.

"PAC-Bayes is a bound-generation machine."
Example: SVM bound

- $\mathcal{H} = \{ h(x) = \text{sign} \langle w, x \rangle, \ w \in \mathbb{R}^d \}$  \quad \text{linear classifiers}
- $P(w) \propto e^{-\lambda \|w\|^2}$  \quad \text{prior: Gaussian around 0}
- $Q(w) \propto e^{-\lambda \|w-v\|^2}$  \quad \text{posterior: Gaussian around } v

Prior: uniform w.r.t. direction  \quad \text{Posterior: non-uniform}
Example: SVM bound

- $\mathcal{H} = \{ h(x) = \text{sign}(w, x), \ w \in \mathbb{R}^d \}$ linear classifiers
- $P(w) \propto e^{-\lambda \|w\|^2}$ prior: Gaussian around 0
- $Q(w) \propto e^{-\lambda \|w-v\|^2}$ posterior shifted by $v$ (non-uniform)

$$R(Q) \leq \hat{R}(Q) + \frac{1}{\sqrt{n}} \left( \lambda \|v\|^2 + \frac{1}{8} + \log \frac{1}{\delta} \right)$$
Example: SVM bound

- $\mathcal{H} = \{ h(x) = \text{sign} \langle w, x \rangle, \ w \in \mathbb{R}^d \}$  \hspace{1cm} linear classifiers
- $P(w) \propto e^{-\lambda \|w\|^2}$  \hspace{1cm} prior: Gaussian around 0
- $Q(w) \propto e^{-\lambda \|w-v\|^2}$  \hspace{1cm} posterior shifted by $v$ (non-uniform)

$$\mathcal{R}(Q) \leq \hat{\mathcal{R}}(Q) + \frac{1}{\sqrt{n}} \left( \lambda \|v\|^2 + \frac{1}{8} + \log \frac{1}{\delta} \right)$$

When $\ell$ is 0-1 loss:

- deterministic classifier $\text{sign} \langle v, x \rangle$ is identical to majority vote of $Q$
- we can relate $\hat{\mathcal{R}}(Q)$ to $\hat{\mathcal{R}}(v)$:

$$\hat{\mathcal{R}}(Q) = \frac{1}{n} \sum_{i=1}^{n} \Phi \left( \frac{y_i \langle v, x_i \rangle}{\|x_i\|} \right) \text{ for } \Phi(t) = \frac{1}{2} \left( 1 - \text{erf} \left( \frac{t}{\sqrt{2}} \right) \right),$$

Together:

$$\frac{1}{2} \mathcal{R}(v) \leq \frac{1}{n} \sum_{i=1}^{n} \Phi \left( \frac{y_i \langle v, x_i \rangle}{\|x_i\|} \right) + \frac{\lambda}{\sqrt{n}} \|v\|^2 + \frac{1}{8} + \log \frac{1}{\delta}$$
Example: Transfer bound

- $\mathcal{H} = \{ h_w(x) : \mathcal{X} \to \mathcal{Y}, \ w \in \mathbb{R}^d \}$ parameterized by $w \in \mathbb{R}^d$
- $P(w) \propto e^{-\lambda \|w-v_0\|^2}$ prior: Gaussian around $v_0$
- $Q(w) \propto e^{-\lambda \|w-v\|^2}$ posterior: Gaussian around $v$
- $\text{KL}(Q||P) = \lambda \|v - v_0\|^2$

$$\mathcal{R}(Q) \leq \hat{\mathcal{R}}(Q) + \frac{1}{\sqrt{n}} \left( \lambda \|v - v_0\|^2 + \frac{1}{8} + \log \frac{1}{\delta} \right)$$

Typical situation for fine-tuning:

- initialize classifier parameters as $v_0$
- train on $\mathcal{D}$ using (stochastic) gradient descent

Good generalization, if parameters don’t move far from initialization.
"dropout rate" $\alpha \in [0, 1]$

set of posterior distributions: $Q_{\theta, \alpha}$:

for each weight: $w_i = \begin{cases} 0 & \text{with prob. } \alpha \\ \theta_i + \epsilon_i & \text{otherwise, for } \epsilon_i \sim \mathcal{N}(0, 1) \end{cases}$

prior distribution: $P = Q_{0, \alpha}$

$\text{KL}(Q || P) = \frac{1-\alpha}{2} \|\theta\|^2$

Zero-ing out weights reduces complexity by factor $\frac{1-\alpha}{2}$:

$$R(Q_{\theta, \alpha}) \leq \hat{R}(Q_{\theta, \alpha}) + \frac{1}{\sqrt{n}} \left( \frac{1-\alpha}{2} \|\theta\|^2 + \frac{1}{8} + \log \frac{1}{\delta} \right)$$

Training: optimize $\hat{R}(Q_{\theta, \alpha}) + \ldots$ via SGD → "dropout training"

Prediction: majority vote over many stochastic networks
Bounds for Deep Learning?
"Understanding deep learning requires rethinking generalization"

[Zhang, Bengio, Hardt, Recht, Vinyals, ICLR 2017]

Observation:

- Deep Neural Networks can have 100s of millions parameters.
- We train them with less than 1 million examples.
- Yet, they don’t seem to overfit.
- Could it be that their capacity is much smaller than one would expect from the number of parameters?
"Understanding deep learning requires rethinking generalization"

[Zhang, Bengio, Hardt, Recht, Vinyals, ICLR 2017]

Observation:
- Deep Neural Networks can have 100s of millions parameters.
- We train them with less than 1 million examples.
- Yet, they don’t seem to overfit.
- Could it be that their capacity is much smaller than one would expect from the number of parameters?

Empirical study:
- let’s explore their empirical Rademacher complexity
- train network with real input data, but random $\pm 1$ labels
Observation:
- Deep Neural Networks can have 100s of millions parameters.
- We train them with less than 1 million examples.
- Yet, they don’t seem to overfit.
- Could it be that their capacity is much smaller than one would expect from the number of parameters?

Empirical study:
- let’s explore their empirical Rademacher complexity
- train network with real input data, but random ±1 labels
- result: networks can learn random labels ($\hat{R} \to 0$)
Observation:
- Deep Neural Networks can have 100s of millions parameters.
- We train them with less than 1 million examples.
- Yet, they don’t seem to overfit.
- Could it be that their capacity is much smaller than one would expect from the number of parameters?

Empirical study:
- let’s explore their empirical Rademacher complexity
- train network with real input data, but random ±1 labels
- result: networks can learn random labels ($\hat{R} \to 0$)

Conclusion:
- we still don’t know why deep networks don’t overfit
- Rademacher-style learning theory does not explain it
• $f : \mathcal{X} \rightarrow \mathcal{Y}$: trained network with many parameters

• $\mathcal{G}$: a set of (smaller) neural networks parametrized by $q$ parameters, each of which can take $r$ different values.

**Theorem**

Let $S = \{(x^1, y^1), \ldots, (x^m, y^m)\}$ be a training set with $m$ samples. For $\lambda > 0$, if $f$ can be approximated by a network $g \in \mathcal{G}$ in the sense that $|f(x^i) - g(x^i)| \leq \gamma$ for $i = 1, \ldots, m$, then (with high probability),

$$\mathcal{R}(g) \leq \frac{1}{m} \sum_{i=1}^{m} [y^i f(x^i) \leq \gamma] + O\left(\sqrt{\frac{q \log r}{m}}\right)$$

**Examples:**

• quantize real-valued network parameter to a few (e.g. $r = 4$) bits

• low-rank decomposition of weight matrices to reduce number of coefficients
"Stronger generalization bounds for deep nets via a compression approach"

[Arora, Ge, Neyshabur, Zhang. ICML 2018]

- $f: \mathcal{X} \rightarrow \mathcal{Y}$: trained network with many parameters
- $\mathcal{G}$: a set of (smaller) neural networks parametrized by $q$ parameters, each of which can take $r$ different values.

**Theorem**

Let $S = \{(x^1, y^1), \ldots, (x^m, y^m)\}$ be a training set with $m$ samples. For $\lambda > 0$, if $f$ can be approximated by a network $g \in \mathcal{G}$ in the sense that $|f(x^i) - g(x^i)| \leq \gamma$ for $i = 1, \ldots, m$, then (with high probability),

$$R(g) \leq \frac{1}{m} \sum_{i=1}^{m} [y^i f(x^i) \leq \gamma] + O\left(\sqrt{\frac{q \log r}{m}}\right)$$

**Problem:**

- theorem bounds quality of $g$, not $f$.
- the bound itself follows immediately from finite hypothesis set:
  - $R(g) \leq \hat{R}(g) + \sqrt{\frac{\log |\mathcal{G}| + \log 1/\delta}{m}}$ and $\log |\mathcal{G}| = \log r^q = q \log r$
  - $\hat{R}(g) = \frac{1}{m} \sum_{i=1}^{m} [y^i g(x^i) \leq 0] \leq \frac{1}{m} \sum_{i=1}^{m} [y^i f(x^i) \leq \gamma]$
Observation:
- deep networks trained by SGD work well

Hypothesis:
- solution found by SGD are "shallow" minima of the objective, so it is robust against small perturbations of the network parameters

Approach:
- PAC-Bayesian bound:
  - prior: Gaussian around weight initialization $w_0$
  - posterior: Gaussian around learned parameters
- variance of Gaussians learned from bound itself (needs union bound)
- several approximations to approximate empirical risk
Observation:
• deep networks trained by SGD work well

Hypothesis:
• solution found by SGD are "shallow" minima of the objective, so it is robust against small perturbations of the network parameters

Approach:
• PAC-Bayesian bound:
  ▶ prior: Gaussian around weight initialization $w_0$
  ▶ posterior: Gaussian around learned parameters
• variance of Gaussians learned from bound itself (needs union bound)
• several approximations to approximate empirical risk

<table>
<thead>
<tr>
<th>Experiment (MNIST)</th>
<th>T-600</th>
<th>T-1200</th>
<th>T-300^2</th>
<th>T-600^2</th>
<th>T-1200^2</th>
<th>T-600^3</th>
<th>R-600</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test error</td>
<td>0.018</td>
<td>0.018</td>
<td>0.015</td>
<td>0.016</td>
<td>0.015</td>
<td>0.013</td>
<td>0.508</td>
</tr>
<tr>
<td>SNN test error</td>
<td>0.034</td>
<td>0.035</td>
<td>0.034</td>
<td>0.033</td>
<td>0.035</td>
<td>0.032</td>
<td>0.503</td>
</tr>
<tr>
<td>PAC-Bayes bound</td>
<td>0.161</td>
<td>0.179</td>
<td>0.170</td>
<td>0.186</td>
<td>0.223</td>
<td>0.201</td>
<td>1.352</td>
</tr>
<tr>
<td>VC dimension</td>
<td>26m</td>
<td>56m</td>
<td>26m</td>
<td>66m</td>
<td>187m</td>
<td>121m</td>
<td>26m</td>
</tr>
</tbody>
</table>
**"Spectrally-normalized margin bounds for neural networks"**  
[Bartlett, Foster Telgarsky, NIPS 2017]

**Theorem 1.1.** Let nonlinearities $(\sigma_1, \ldots, \sigma_L)$ and reference matrices $(M_1, \ldots, M_L)$ be given as above (i.e., $\sigma_i$ is $\rho_i$-Lipschitz and $\sigma_i(0) = 0$). Then for $(x, y), (x_1, y_1), \ldots, (x_n, y_n)$ drawn iid from any probability distribution over $\mathbb{R}^d \times \{1, \ldots, k\}$, with probability at least $1 - \delta$ over $((x_i, y_i))_{i=1}^n$, every margin $\gamma > 0$ and network $F_A : \mathbb{R}^d \to \mathbb{R}^k$ with weight matrices $A = (A_1, \ldots, A_L)$ satisfy

$$
\text{Pr} \left[ \arg \max \limits_j F_A(x)_j \neq y \right] \leq \hat{R}_\gamma(F_A) + \tilde{O} \left( \frac{\|X\|_2 R_A}{\gamma n} \ln(W) + \sqrt{\frac{\ln(1/\delta)}{n}} \right),
$$

where $\hat{R}_\gamma(f) \leq n^{-1} \sum_i \mathbb{1} [f(x_i)_y \leq \gamma + \max_{j \neq y} f(x_i)_j]$ and $\|X\|_2 = \sqrt{\sum_i \|x_i\|_2^2}$.

---

**"A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks"**  
[Neyshabur, Bhojanapalli, Srebro, ICML 2018]

**Theorem 1 (Generalization Bound).** For any $B, d, h > 0$, let $f_w : \mathcal{X}_{B,n} \to \mathbb{R}^k$ be a $d$-layer feedforward network with ReLU activations. Then, for any $\delta, \gamma > 0$, with probability $\geq 1 - \delta$ over a training set of size $m$, for any $w$, we have:

$$L_0(f_w) \leq \tilde{L}_\gamma(f_w) + O \left( \sqrt{\frac{B^2 d^2 h \ln(dh) \prod_{i=1}^d \|W_i\|_2^2 \sum_{i=1}^d \|W_i\|_2^2 + \ln \frac{dm}{\delta}}{\gamma^2 m}} \right).$$