<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>
Evaluating Predictors
So, you’ve trained a predictor, $f : \mathcal{X} \rightarrow \mathcal{Y}$. How good is it really?

- The **loss on the training set**, $\mathcal{D} = \{ (x^1, y^1), \ldots, (x^n, y^n) \}$,

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(y^i, f(x^i))$$

tells us little about the quality of a learned predictor. Reporting it would be misleading as best.

- Really, we would care about the **expected loss** (generalization loss),

$$\mathcal{R}(f) = \mathbb{E}_{(x, y) \sim p(x, y)} \ell(y, f(x)).$$

Unfortunately, we cannot compute it, because $p(x, y)$ is unknown.

- In practice, we use a **test set**, 

$$\mathcal{D}_{\text{tst}} = \{ (x^1, y^1), \ldots, (x^m, y^m) \},$$

i.e. examples that were not used for training, and compute

$$\hat{R}_{\text{tst}}(f) = \frac{1}{m} \sum_{i=1}^{m} \ell(y^i, f(x^i))$$

Why?
So, you’ve trained a predictor, \( f : \mathcal{X} \rightarrow \mathcal{Y} \). How good is it really?

- The **loss on the training set**, \( \mathcal{D} = \{ (x^1, y^1), \ldots, (x^n, y^n) \} \),

\[
\hat{\mathcal{R}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(y^i, f(x^i))
\]

tells us little about the quality of a learned predictor. Reporting it would be misleading as best.

- Really, we would care about the **expected loss** (\(=\)generalization loss),

\[
\mathcal{R}(f) = \mathbb{E}_{(x,y) \sim p(x,y)} \ell(y, f(x)).
\]

Unfortunately, we cannot compute it, because \( p(x, y) \) is unknown.

- In practice, we use a **test set**, \( \mathcal{D}_{\text{tst}} = \{ (x^1, y^1), \ldots, (x^m, y^m) \} \),

i.e. examples that were not used for training, and compute

\[
\hat{\mathcal{R}}_{\text{tst}}(f) = \frac{1}{m} \sum_{i=1}^{m} \ell(y^i, f(x^i))
\]

Why? Let’s look at \( \hat{\mathcal{R}}_{\text{tst}}(f) \) as an **estimator** of \( \mathcal{R}(f) \).
Excurse: Estimators
Estimators

An estimator is a rule for calculating an estimate, $\hat{E}(S)$, of a quantity $E$ based on observed data, $S$. If $S$ is random, then $\hat{E}(S)$ is also random.

Properties of estimators: bias

Let $\hat{E}$ be an estimator of $E$. We can compute the expected value of the estimate, $\mathbb{E}_S[\hat{E}(S)]$, and define:

$$\text{bias}(\hat{E}) = \mathbb{E}_S[\hat{E}(S)] - E$$

Properties of estimators: unbiasedness

If $\hat{E}$ is an estimator of $E$, we call it unbiased, if

$$\text{bias}(\hat{E}) = 0 \quad (\text{i.e. } \mathbb{E}_S[\hat{E}(S)] = E)$$

If $\hat{E}$ is unbiased, we can think of it as a noisy version of $E$. 
**Example: Estimating the mean of a Gaussian**

Examples: let \( S = \{ z^1, z^2, \ldots, z^n \} \) be independent samples from \( \mathcal{N}(x; \mu, \sigma^2) \). We look at different estimators for \( \mu \):

- \( \hat{E}(S) = 1 \) has bias \( 1 - \mu \).  
  \[
  \text{bias}(\hat{E}) = \mathbb{E}_S \hat{E}(S) - \mu = 1 - \mu
  \]
Example: Estimating the mean of a Gaussian

Examples: let $S = \{z^1, z^2, \ldots, z^n\}$ be independent samples from $\mathcal{N}(x; \mu, \sigma^2)$. We look at different estimators for $\mu$:

- $\hat{E}(S) = 1$ has bias $1 - \mu$.  
  \[
  \text{bias}(\hat{E}) = \mathbb{E}_S \hat{E}(S) - \mu = 1 - \mu
  \]

- $\hat{E}(S) = \frac{1}{n} \sum_{i=1}^{n} z^i$ is unbiased.

Example: Estimating the mean of a Gaussian

Examples: let $S = \{z^1, z^2, \ldots, z^n\}$ be independent samples from $\mathcal{N}(x; \mu, \sigma^2)$. We look at different estimators for $\mu$:

- $\hat{E}(S) = 1$ has bias $1 - \mu$.  
  
  $\text{bias}(\hat{E}) = \mathbb{E}_S \hat{E}(S) - \mu = 1 - \mu$

- $\hat{E}(S) = \frac{1}{n} \sum_{i=1}^{n} z^i$ is unbiased.

  
  $\mathbb{E}_S[\hat{E}(S)] = \mathbb{E}_S[\frac{1}{n} \sum_i z^i] = \frac{1}{n} \sum_i \mathbb{E}_S[z^i] = \frac{1}{n} \sum_i \mu = \mu$
Example: Estimating the mean of a Gaussian

Examples: let $S = \{z^1, z^2, \ldots, z^n\}$ be independent samples from $\mathcal{N}(x; \mu, \sigma^2)$. We look at different estimators for $\mu$:

- $\hat{E}(S) = 1$ has bias $1 - \mu$. 
  \[ \text{bias}(\hat{E}) = \mathbb{E}_S \hat{E}(S) - \mu = 1 - \mu \]

- $\hat{E}(S) = \frac{1}{n} \sum_{i=1}^{n} z^i$ is unbiased.
  \[ \mathbb{E}_S[\hat{E}(S)] = \mathbb{E}_S[\frac{1}{n} \sum_{i} z^i] = \frac{1}{n} \sum_{i} \mathbb{E}_S[z^i] = \frac{1}{n} \sum_{i} \mu = \mu \]

- $\hat{E}(S) = z^1$ is unbiased: 
  \[ \mathbb{E}_S[\hat{E}(S)] = \mathbb{E}_S[z^1] = \mu \]
Example: Estimating the mean of a Gaussian

Examples: let $S = \{z^1, z^2, \ldots, z^n\}$ be independent samples from $\mathcal{N}(x; \mu, \sigma^2)$. We look at different estimators for $\mu$:

- $\hat{E}(S) = 1$ has bias $1 - \mu$. $\text{bias}(\hat{E}) = \mathbb{E}_S \hat{E}(S) - \mu = 1 - \mu$

- $\hat{E}(S) = \frac{1}{n} \sum_{i=1}^{n} z^i$ is unbiased.

\[
\mathbb{E}_S[\hat{E}(S)] = \mathbb{E}_S[\frac{1}{n} \sum_i z^i] = \frac{1}{n} \sum_i \mathbb{E}_S[z^i] = \frac{1}{n} \sum_i \mu = \mu
\]

- $\hat{E}(S) = z^1$ is unbiased: $\mathbb{E}_S[\hat{E}(S)] = \mathbb{E}_S[z^1] = \mu$

- $\hat{E}(S) = \frac{1}{n} + \frac{1}{n} \sum_{i=1}^{n} z^i$ has bias $\frac{1}{n}$
Example: Stochastic Gradient Descent

Reminder: we wanted to optimize

\[ f(\theta) = \sum_{j=1}^{n} f_j(\theta) \]

Instead of

\[ v := \nabla f(\theta) \]

we use

\[ \hat{v} := n \nabla f_i(\theta) \quad \text{with} \quad i \text{ uniformly } \sim \{1, \ldots, n\} \]

Claim: \( \hat{v} \) is an unbiased estimator for \( v \).
Example: Stochastic Gradient Descent

Reminder: we wanted to optimize

$$f(\theta) = \sum_{j=1}^{n} f_j(\theta)$$

Instead of

$$\nu := \nabla f(\theta)$$

we use

$$\hat{\nu} := n\nabla f_i(\theta) \quad \text{with} \quad i \overset{\text{uniformly}}{\sim} \{1, \ldots, n\}$$

Claim: $\hat{\nu}$ is an unbiased estimator for $\nu$.

$$\mathbb{E}_i[\hat{\nu}] = \sum_{i=1}^{n} p(i) \hat{\nu}[i] = \sum_{i=1}^{n} \frac{1}{n} n\nabla f_i(\theta) = \sum_{i=1}^{n} \nabla f_i(\theta) = \nabla f(\theta)$$
How far is one estimate, \( \hat{E}(S) \), from its expected value, \( \mathbb{E}_S[\hat{E}(S)] \)?

**Properties of estimators: variance**

\[
\text{Var}(\hat{E}) = \mathbb{E}_S\left[ (\hat{E}(S) - \mathbb{E}_S[\hat{E}(S)])^2 \right]
\]

If \( \text{Var}(\hat{E}) \) is large, then the estimate for different \( S \) differ a lot.

**Examples:**

Let \( S = \{z^1, z^2, \ldots, z^n\} \) be independent samples from \( \mathcal{N}(x; \mu, \sigma^2) \). We look at different estimators for \( \mu \):

- \( \hat{E}(S) = 1 \) has variance 0.
- \( \hat{E}(S) = \frac{1}{n} \sum_{i=1}^{n} z_i \) has variance \( \frac{\sigma^2}{n} \)  \( \text{(exercise)} \)
- \( \hat{E}(S) = z_1 \) has variance \( \sigma^2 \)
- \( \hat{E}(S) = \frac{1}{n-1} \sum_{i=1}^{n} z_i \) has variance \( ? \)  \( \text{(exercise)} \)
It’s good to have small or no bias, and it’s good to have small variance.

If you can’t have both at the same time, look for a reasonable trade-off.

Image: adapted from http://scott.fortmann-roe.com/docs/BiasVariance.html
What if we get more and more data, \( S_n = \{z_1, \ldots, z_n\} \) for \( n \to \infty \)?

### Properties of estimators: consistency

An estimator \( \hat{E} \) is called **consistent**, if

\[
\hat{E}(S_n) \to E \quad \text{for} \quad n \to \infty.
\]

Convergence is "in probability", i.e. it means,

\[
\lim_{n \to \infty} \Pr\{ |\hat{E}(S_n) - E| > \epsilon \} = 0.
\]

Any estimator \( \hat{E} \) with \( \text{bias}(\hat{E}) \xrightarrow{n \to \infty} 0 \) and \( \text{Var}(\hat{E}) \xrightarrow{n \to \infty} 0 \) is consistent.

Proof... follows from later observations
Is
\[ \hat{R}_{\text{tst}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i)) \]
a good estimator of
\[ R(f) = \mathbb{E}_{(x,y) \sim p(x,y)} \ell(y, f(x)) \]

Yes, if we use the right data:

**Test error as an unbiased estimator**

If \( D_{\text{tst}} = \{ (x^1, y^1), \ldots, (x^m, y^m) \} \) are sampled independently from the distribution \( p(x, y) \), and \( f \) was chosen independently of them. Then \( \hat{R}_{\text{tst}} \) is an unbiased and consistent estimator of \( R \):

Otherwise? Things might go wrong...
Proof: unbiased

- $\mathcal{D}$ is a set of random variables, $\mathcal{(X^1, Y^1), \ldots, (X^m, Y^m) \in X \times Y}$.
- All $(X^1, Y^1), \ldots, (X^m, Y^m)$ are independent with distribution $p$.
- For fixed functions $f, \ell$, chosen independently of $\mathcal{D}$

$$
\ell(Y^1, f(X^1)), \ldots, \ell(Y^m, f(X^m))
$$

are independent (real-valued) random variables.

$$
\mathbb{E}_{\mathcal{D} \sim p^\otimes m} \hat{\mathcal{R}}_{tst}(\mathcal{D}) = \mathbb{E}_{(X^1, Y^1), \ldots, (X^m, Y^m) \sim p} \frac{1}{m} \sum_{i=1}^{m} \ell(Y^i, f(X^i)) \\
= \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_{(X^1, Y^1), \ldots, (X^m, Y^m) \sim p} \ell(Y^i, f(X^i)) \\
= \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_{(X^i, Y^i) \sim p} \ell(Y^i, f(X^i)) \\
= \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_{(X, Y) \sim p} \ell(Y, f(X)) \\
= \mathbb{E}_{(X, Y) \sim p} \ell(Y, f(X)) = \mathcal{R}(f)
$$

$\square$
Excurse: Concentration of Measure I
Concentration of Measure Inequalities

- $Z$ random variables, taking values $z \in Z \subseteq \mathbb{R}$.
- $p(Z = z)$ probability distribution
  - $\mu = \mathbb{E}[Z]$ mean
  - $\text{Var}(z) = \mathbb{E}[(Z - \mu)^2]$ variance

**Lemma (Law of Large Numbers)**

Let $Z_1, Z_2, \ldots$, be i.i.d. random variables with mean $\mathbb{E}[Z] < \infty$, then

$$\frac{1}{m} \sum_{i=1}^{m} Z_i \xrightarrow{m \to \infty} \mathbb{E}[Z] \quad \text{with probability 1}.$$

In machine learning, we have finite data, so $m \to \infty$ is less important. Concentration of measure inequalities quantify the deviation between average and expectation for finite $m$. 
Assumption: \( \mathcal{Z} \subseteq \mathbb{R}_+ \), i.e. \( \mathcal{Z} \) takes only non-negative values.

**Lemma (Markov’s inequality)**

\[
\forall a > 0 : \quad \mathbb{P}[Z \geq a] \leq \frac{\mathbb{E}[Z]}{a}.
\]

**Proof.** Step 1) We can write

\[
\mathbb{E}[Z] = \int_{x=0}^{\infty} \mathbb{P}[Z \geq x] \, dx
\]

Step 2) Since \( \mathbb{P}[Z \geq x] \) is non-increasing in \( x \), we have for any \( a \geq 0 \):

\[
\mathbb{E}[Z] \geq \int_{x=0}^{a} \mathbb{P}[Z \geq x] \, dx \geq \int_{x=0}^{a} \mathbb{P}[Z \geq a] \, dx = a \, \mathbb{P}[Z \geq a]
\]
Proof sketch of Step 1 inequality (ignoring questions of measurability and exchange of limit processes and writing the expression as if \( Z \) had a density \( p(z) \))

\[
\mathbb{P}[Z \geq x] = \int_{z=x}^{\infty} p(z) \, dz = \int_{z=0}^{\infty} [z \geq x] \, p(z) \, dz
\]

\[
\int_{x=0}^{\infty} \mathbb{P}[Z \geq x] \, dx = \int_{x=0}^{\infty} \int_{z=0}^{\infty} [z \geq x] \, p(z) \, dz \, dx
\]

\[
= \int_{z=0}^{\infty} \int_{x=0}^{\infty} [z \geq x] \, dx \, p(z) \, dz
\]

\[
= \int_{z=0}^{\infty} \int_{x=0}^{z} dx \, p(z) \, dz
\]

\[
= \int_{z=0}^{\infty} z \, p(z) \, dz
\]

\[
= \mathbb{E}[Z]
\]
Assumption: \( Z \subseteq \mathbb{R}_+ \), i.e. \( Z \) takes only non-negative values.

**Lemma (Markov’s inequality)**

\[ \forall a \geq 0 : \quad P[Z \geq a] \leq \frac{E[Z]}{a}. \]

**Example**

Is it possible that more than half of the population have a salary more than twice the mean salary?

No, by corollary with \( a = 2 \).

Easily: \( P(\$1) = 0.99 \), \( P(\$100000) = 0.01 \).
Assumption: $\mathcal{Z} \subseteq \mathbb{R}_+$, i.e. $\mathcal{Z}$ takes only non-negative values.

**Lemma (Markov’s inequality)**

$$\forall a \geq 0 : \Pr[Z \geq a] \leq \frac{\mathbb{E}[Z]}{a}.$$  

**Corollary**

$$\forall a \geq 0 : \Pr[Z \geq a \mathbb{E}[Z]] \leq \frac{1}{a}.$$  

**Example**

Is it possible that more than half of the population have a salary more than twice the mean salary? No, by corollary with $a = 2$. 

Is it possible that more than 90% of the population have a salary less than one tenth of the mean? Easily: $p(1) = 0.99$, $p(100000) = 0.01$. 

Assumption: $\mathcal{Z} \subseteq \mathbb{R}_+$, i.e. $\mathcal{Z}$ takes only non-negative values.

**Lemma (Markov’s inequality)**

\[ \forall a \geq 0 : \quad P[Z \geq a] \leq \frac{\mathbb{E}[Z]}{a}. \]

**Corollary**

\[ \forall a \geq 0 : \quad P[Z \geq a \mathbb{E}[Z]] \leq \frac{1}{a}. \]

**Example**

Is it possible that more than half of the population have a salary more than twice the mean salary? No, by corollary with $a = 2$.

**Example**

Is it possible that more than 90% of the population have a salary less than one tenth of the mean?
Assumption: $Z \subseteq \mathbb{R}_+$, i.e. $Z$ takes only non-negative values.

**Lemma (Markov’s inequality)**

$$\forall a \geq 0 : \ P[Z \geq a] \leq \frac{E[Z]}{a}.$$  

**Corollary**

$$\forall a \geq 0 : \ P[Z \geq a \ E[Z]] \leq \frac{1}{a}.$$  

**Example**

Is it possible that more than half of the population have a salary more than twice the mean salary? No, by corollary with $a = 2$.

**Example**

Is it possible that more than 90% of the population have a salary less than one tenth of the mean? Easily: $p(1) = 0.99$, $p(100000) = 0.01$.  

Lemma (Chebyshev’s inequality)

\[ \forall a \geq 0 : \ P[|Z - \mathbb{E}[Z]| \geq a] \leq \frac{\text{Var}[Z]}{a^2} \]

Proof. Apply Markov’s Inequality to the random variable \((Z - \mathbb{E}[Z])^2\).
Lemma (Chebyshev’s inequality)

\[ \forall a \geq 0 : \quad P[|Z - \mathbb{E}[Z]| \geq a] \leq \frac{\text{Var}[Z]}{a^2} \]

**Proof.** Apply Markov’s Inequality to the random variable \((Z - \mathbb{E}[Z])^2\).

For any \(a \geq 0\):

\[
P[|Z - \mathbb{E}[Z]| \geq a] = P[(Z - \mathbb{E}[Z])^2 \geq a^2] \overset{\text{Markov}}{\leq} \frac{\mathbb{E}[(Z - \mathbb{E}[Z])^2]}{a^2} = \frac{\text{Var}[Z]}{a^2}.
\]
Lemma (Chebyshev’s inequality)

\[ \forall a \geq 0 : \ P[|Z - \mathbb{E}[Z]| \geq a] \leq \frac{\text{Var}[Z]}{a^2} \]

**Proof.** Apply Markov’s Inequality to the random variable \((Z - \mathbb{E}[Z])^2\).

For any \(a \geq 0\):

\[
P[|Z - \mathbb{E}[Z]| \geq a] = P[(Z - \mathbb{E}[Z])^2 \geq a^2] \overset{\text{Markov}}{\leq} \frac{\mathbb{E}[(Z - \mathbb{E}[Z])^2]}{a^2} = \frac{\text{Var}[Z]}{a^2}.
\]

**Remark:** Chebyshev ineq. has similar role as "\(\sigma\)-rules" for Gaussians:

- 68% of probability mass of a Gaussian lie within \(\mu \pm \sigma\),
- 95% of probability mass of a Gaussian lie within \(\mu \pm 2\sigma\),
- 99.7% of probability mass of a Gaussian lie within \(\mu \pm 3\sigma\),

Chebyshev holds for arbitrary probability distributions, not just Gaussians.
Example (Soccer Match Statistics)

- \( z = -1 \) for loss, \( z = 0 \) for draw, \( z = 1 \) for win.
- \( p(-1) = \frac{1}{10}, \ p(1) = \frac{1}{10}, \ p(0) = \frac{4}{5} \).
- \( \mathbb{E}[Z] = 0 \).
- \( \text{Var}[Z] = \mathbb{E}[(Z)^2] = \frac{1}{10}(-1)^2 + \frac{4}{5}0^2 + \frac{1}{10}(1)^2 = \frac{1}{5} \)

What if we pretended \( Z \) is Gaussian?

- \( \mu = 0, \ \sigma = \sqrt{\frac{1}{5}} \approx 0.45 \),
- we expect \( \leq 5\% \) prob. mass outside of the \( 2\sigma \)-interval \([-0.9, 0.9]\)
- but really, its 20%!

With Chebyshev:
- \( \mathbb{P}[|Z| \geq 0.9] \leq \frac{1}{5}/(0.9)^2 \approx 0.247 \), so bound is correct
Applying Chebyshev’s Inequality

**Lemma (Quantitative Version of the Law of Large Numbers)**

Set $Z_1, \ldots, Z_m$ be i.i.d. random variables with $\mathbb{E}[Z_i] = \mu$ and $\text{Var}[Z_i] \leq C$. Then, for any $\delta \in (0, 1)$, the following inequality holds with probability at least $1 - \delta$:

$$
\left| \frac{1}{m} \sum_{i=1}^{m} Z_i - \mu \right| < \sqrt{\frac{C}{\delta m}}.
$$

Equivalent formulations:

$$
\Pr \left[ \left| \frac{1}{m} \sum_{i=1}^{m} Z_i - \mu \right| < \sqrt{\frac{C}{\delta m}} \right] \geq 1 - \delta.
$$

$$
\Pr \left[ \left| \frac{1}{m} \sum_{i=1}^{m} Z_i - \mu \right| \geq \sqrt{\frac{C}{\delta m}} \right] \leq \delta.
$$
Lemma (Quantitative Version of the Law of Large Numbers)

Set $Z_1, \ldots, Z_m$ be i.i.d. random variables with $\mathbb{E}[Z_i] = \mu$ and $\text{Var}[Z_i] \leq C$. Then, for any $\delta \in (0, 1)$,

$$\Pr \left[ \left| \frac{1}{m} \sum_{i=1}^{m} Z_i - \mu \right| \geq \sqrt{\frac{C}{\delta m}} \right] \leq \delta.$$
Applying Chebyshev’s Inequality

**Lemma (Quantitative Version of the Law of Large Numbers)**

Set $Z_1, \ldots, Z_m$ be i.i.d. random variables with $\mathbb{E}[Z_i] = \mu$ and $\text{Var}[Z_i] \leq C$. Then, for any $\delta \in (0, 1)$,

$$
\Pr \left[ \left| \frac{1}{m} \sum_{i=1}^{m} Z_i - \mu \right| \geq \sqrt{\frac{C}{\delta m}} \right] \leq \delta.
$$

**Proof.** The $Z_i$ are indep., so $\text{Var} \left[ \frac{1}{m} \sum_{i=1}^{m} Z_i \right] = \frac{1}{m^2} \sum_{i=1}^{m} \text{Var}[Z_i] \leq \frac{C}{m}$.

2) Chebyshev’s inequality gives us for any $a \geq 0$: $

\mathbb{P} \left[ \left| \frac{1}{m} \sum_{i=1}^{m} Z_i - \mu \right| \geq a \right] \leq \frac{\text{Var} \left[ \frac{1}{m} \sum_{i=1}^{m} Z_i \right]}{a^2} \leq \frac{C}{ma^2}.

$Setting $\delta = \frac{C}{ma^2}$ and solving for $a$ yields $a = \sqrt{\frac{C}{\delta m}}$. 
Sanity check: How large should my test set be?

\[
\mathbb{P} \left[ \left| \frac{1}{m} \sum_{i=1}^{m} Z_i - \mu \right| \leq \sqrt{\frac{C}{\delta m}} \right] \geq 1 - \delta.
\]

Setup: fixed classifier \( g : \mathcal{X} \rightarrow \mathcal{Y} \), 0/1-loss: \( \ell(\bar{y}, y) = [\bar{y} \neq y] \)

- test set \( \mathcal{D} = \{(x^1, y^1), \ldots, (x^m, y^m)\} \sim i.i.d. p(x, y) \),
- random variables \( Z_i = [g(x^i) \neq y^i] \in \{0, 1\} \),
- \( \mathbb{E}[Z^i] = \mathbb{E}\{[g(x^i) \neq y^i]\} = \mu \) (generalization error of \( g \))
- \( \text{Var}[Z^i] = \mathbb{E}\{(Z^i - \mu)^2\} = \mu(1-\mu)^2 + (1-\mu)\mu^2 = \mu(1-\mu) \leq \frac{1}{4} =: C \)

Setup: fixed confidence, e.g. \( \delta = 0.1 \), \( \sqrt{\frac{C}{\delta m}} = \sqrt{\frac{0.25}{0.1m}} = \sqrt{\frac{2.5}{m}} \)

\[
\mathbb{P} \left[ \left| \frac{1}{m} \sum_{i=1}^{m} Z_i - \mu \right| \leq \sqrt{\frac{2.5}{m}} \right] \geq 0.9
\]

To be 90%-certain that the error is within \( \pm 0.05 \), use \( m \geq 1,000 \).
Sanity check: How large should my test set be?

\[
\mathbb{P} \left[ \left| \frac{1}{m} \sum_{i=1}^{m} Z_i - \mu \right| \leq \sqrt{\frac{C}{\delta m}} \right] \geq 1 - \delta.
\]

Setup: fixed classifier \( g : \mathcal{X} \rightarrow \mathcal{Y} \), 0/1-loss: \( \ell(\bar{y}, y) = [\bar{y} \neq y] \)

- test set \( \mathcal{D} = \{(x^1, y^1), \ldots, (x^m, y^m)\} \sim i.i.d. p(x, y) \),
- random variables \( Z_i = [g(x^i) \neq y^i] \in \{0, 1\} \),
- \( \mathbb{E}[Z^i] = \mathbb{E}\{[g(x^i) \neq y^i]\} = \mu \) (generalization error of \( g \))
- \( \text{Var}[Z^i] = \mathbb{E}\{(Z^i - \mu)^2\} = \mu(1-\mu)^2 + (1-\mu)\mu^2 = \mu(1-\mu) \leq \frac{1}{4} =: C \)

Setup: fixed confidence, e.g. \( \delta = 0.1 \), \( \sqrt{\frac{C}{\delta m}} = \sqrt{\frac{0.25}{0.1m}} = \sqrt{\frac{2.5}{m}} \)

\[
\mathbb{P} \left[ \left| \frac{1}{m} \sum_{i=1}^{m} Z_i - \mu \right| \leq \sqrt{\frac{2.5}{m}} \right] \geq 0.9
\]

To be 90%-certain that the error is within \( \pm 0.05 \), use \( m \geq 1,000 \).
To be 99%-certain that the error is within \( \pm 0.05 \), use \( m \geq 10,000 \).
To be 90%-certain that the error is within \( \pm 0.005 \), use \( m \geq 100,000 \).

(for this case, tighter bounds are possible: later...)
Back to machine learning
Predictor Training (idealized)

**input** training data $\mathcal{D}_{\text{trn}}$

**input** learning procedure $A$

$$g \leftarrow A[\mathcal{D}] \quad \text{(apply } A \text{ with } \mathcal{D} \text{ as training set)}$$

**output** resulting predictor $g : \mathcal{X} \rightarrow \mathcal{Y}$

Predictor Evaluation

**input** trained predictor $g : \mathcal{X} \rightarrow \mathcal{Y}$

**input** test data $\mathcal{D}_{\text{tst}}$

apply $g$ to $\mathcal{D}_{\text{tst}}$ and measure performance $R_{\text{tst}}$

**output** performance estimate $R_{\text{tst}}$
Predictor Training (idealized)

**input** training data $\mathcal{D}_{\text{trn}}$

**input** learning procedure $A$

\[ g \leftarrow A[\mathcal{D}] \quad \text{(apply $A$ with $\mathcal{D}$ as training set)} \]

**output** resulting predictor $g : \mathcal{X} \rightarrow \mathcal{Y}$

Predictor Evaluation

**input** trained predictor $g : \mathcal{X} \rightarrow \mathcal{Y}$

**input** test data $\mathcal{D}_{\text{tst}}$

  apply $g$ to $\mathcal{D}_{\text{tst}}$ and measure performance $R_{\text{tst}}$

**output** performance estimate $R_{\text{tst}}$

**Remark:** In commercial applications, this is realistic:

- given some training set one builds a single system,
- one deploys it to the customers,
- the customers use it on their own data, and complain if disappointed

In research, one typically has no customer, but only a fixed amount of data to work with, so one *simulates* the above protocol.
Classifier Training and Evaluation

**input** data $\mathcal{D}$

**input** learning method $A$

- split $\mathcal{D} = \mathcal{D}_{trn} \cup \mathcal{D}_{tst}$ disjointly
- set aside $\mathcal{D}_{tst}$ to a safe place // do not look at it
- $g \leftarrow A[\mathcal{D}_{trn}]$ // learn a predictor from $\mathcal{D}_{trn}$
- apply $g$ to $\mathcal{D}_{tst}$ and measure performance $R_{tst}$

**output** performance estimate $R_{tst}$

Remark. $\mathcal{D}_{tst}$ should be as small as possible, to keep $\mathcal{D}_{trn}$ as big as possible, but large enough to be convincing.

- sometimes: 50%/50% for small datasets
- more often: 80% training data, 20% test data
- for large datasets: 90% training, 10% test data.

$\mathcal{D}_{tst}$ is “use once”: it cannot be used for any decisions in building the predictor, only to evaluate it at the very end.
Classifier Training and Evaluation

**input** data $\mathcal{D}$

**input** learning method $A$

split $\mathcal{D} = \mathcal{D}_{\text{trn}} \cup \mathcal{D}_{\text{tst}}$ disjointly

set aside $\mathcal{D}_{\text{tst}}$ to a safe place // do not look at it

$g \leftarrow A[\mathcal{D}_{\text{trn}}]$ // learn a predictor from $\mathcal{D}_{\text{trn}}$

apply $g$ to $\mathcal{D}_{\text{tst}}$ and measure performance $R_{\text{tst}}$

**output** performance estimate $R_{\text{tst}}$

**Remark.** $\mathcal{D}_{\text{tst}}$ should be as small as possible, to keep $\mathcal{D}_{\text{trn}}$ as big as possible, but large enough to be convincing.

- sometimes: 50%/50% for small datasets
- more often: 80% training data, 20% test data
- for large datasets: 90% training, 10% test data.
**Classifier Training and Evaluation**

**input** data \( D \)

**input** learning method \( A \)

- split \( D = D_{\text{trn}} \cup D_{\text{tst}} \) disjointly
- set aside \( D_{\text{tst}} \) to a safe place // do not look at it

\( g \leftarrow A[D_{\text{trn}}] \) // learn a predictor from \( D_{\text{trn}} \)

apply \( g \) to \( D_{\text{tst}} \) and measure performance \( R_{\text{tst}} \)

**output** performance estimate \( R_{\text{tst}} \)

**Remark.** \( D_{\text{tst}} \) should be as small as possible, to keep \( D_{\text{trn}} \) as big as possible, but large enough to be convincing.

- sometimes: 50%/50% for small datasets
- more often: 80% training data, 20% test data
- for large datasets: 90% training, 10% test data.

\( D_{\text{tst}} \) is "use once": it cannot be used for any decisions in building the predictor, only to evaluate it at the very end.
Classifier Training and Evaluation

**input** data $\mathcal{D}$

**input** learning method $A$

split $\mathcal{D} = \mathcal{D}_{trn} \cup \mathcal{D}_{tst}$ disjointly

set aside $\mathcal{D}_{tst}$ to a safe place \hspace{1cm} // do not look at it

$g \leftarrow A[\mathcal{D}_{trn}]$ \hspace{1cm} // learn a predictor from $\mathcal{D}_{trn}$

apply $g$ to $\mathcal{D}_{tst}$ and measure performance $R_{tst}$

**output** performance estimate $R_{tst}$

In practice we often want more: not just train a classifier and evaluate it, but

- select the best algorithm out of multiple ones,
- select the best (hyper)parameters for a training algorithm.

We simulate the classifier evaluation step during the training procedure. This needs (at least) one additional data split:
Training and Selecting between Multiple Models

**input** data $\mathcal{D}$

**input** set of method $\mathcal{A} = \{A_1, \ldots, A_K\}$

split $\mathcal{D} = \mathcal{D}_{\text{trnval}} \cup \mathcal{D}_{\text{tst}}$ disjointly

set aside $\mathcal{D}_{\text{tst}}$ to a safe place (do not look at it)

split $\mathcal{D}_{\text{trnval}} = \mathcal{D}_{\text{trn}} \cup \mathcal{D}_{\text{val}}$ disjointly

**for all** models $A_i \in \mathcal{A}$ **do**

$$g_i \leftarrow A_i[D_{\text{trn}}]$$

apply $g_i$ to $\mathcal{D}_{\text{val}}$ and measure performance $E_{\text{val}}(A_i)$

**end for**

pick best performing $A_i$

(optional) $g_i \leftarrow A_i[D_{\text{trnval}}]$ \hspace{1em} // retrain best method on larger dataset

apply $g_i$ to $\mathcal{D}_{\text{tst}}$ and measure performance $R_{\text{tst}}$

**output** performance estimate $R_{\text{tst}}$

How to split? For example $\frac{1}{3} : \frac{1}{3} : \frac{1}{3}$ or $70\% : 10\% : 20\%$. 
Discussion.

- Each algorithm is trained on $\mathcal{D}_{\text{trn}}$ and evaluated on disjoint $\mathcal{D}_{\text{val}}$ ✓
- You select a predictor based on $R_{\text{val}}$ (its performance on $\mathcal{D}_{\text{val}}$), only afterwards $\mathcal{D}_{\text{tst}}$ is used. ✓
- $\mathcal{D}_{\text{tst}}$ is used to evaluate the final predictor and nothing else. ✓
Discussion.

- Each algorithm is trained on $D_{trn}$ and evaluated on disjoint $D_{val}$ ✓
- You select a predictor based on $R_{val}$ (its performance on $D_{val}$), only afterwards $D_{tst}$ is used. ✓
- $D_{tst}$ is used to evaluate the final predictor and nothing else. ✓

Problems.

- small $D_{val}$ is bad: $R_{val}$ could be bad estimate of $g_A$’s true performance, and we might pick a suboptimal method.
- large $D_{val}$ is bad: $D_{trn}$ is much smaller than $D_{trnval}$, so the classifier learned on $D_{trn}$ might be much worse than necessary.
- retraining the best model on $D_{trnval}$ might overcome that, but it comes at a risk: just because a model worked well when trained on $D_{trn}$, this does not mean it’ll also work well when trained on $D_{trnval}$. 

Leave-one-out Evaluation (for a single model/algorithm)

**input** algorithm $A$

**input** loss function $\ell$

**input** data $D$ (trnval part only: test part set aside earlier)

for all $(x^i, y^i) \in D$ do

$g^{-i} \leftarrow A[D \setminus \{(x^i, y^i)\}]$  // $D_{trn}$ is $D$ with $i$-th example removed

$r^i \leftarrow \ell(y^i, g^{-i}(x^i))$  // $D_{val} = \{(x^i, y^i)\}$, disjoint to $D_{trn}$

end for

**output** $R_{\text{loo}} = \frac{1}{n} \sum_{i=1}^{n} r^i$  (average leave-one-out risk)

**Properties.**

- Each $r^i$ is a unbiased (but high variance) estimate of the risk $\mathcal{R}(g^{-i})$
- $D \setminus \{(x^i, y^i)\}$ is almost the same as $D$, so we can hope that each $g^{-i}$ is almost the same as $g = A[D]$.
- Therefore, $R_{\text{loo}}$ can be expected a good estimate of $\mathcal{R}(g)$

**Problem:** slow, trains $n$ times on $n - 1$ examples instead of once on $n$
Compromise: use fixed number of small $D_{\text{val}}$

**$K$-fold Cross Validation (CV)**

**input** algorithm $A$, loss function $\ell$, data $D$ (trnval part)

- split $D = \bigcup_{k=1}^{K} D_k$ into $K$ equal sized disjoint parts

**for** $k = 1, \ldots, K$ **do**

- $g^{-k} \leftarrow A[D \setminus D_k]$
- $r^k \leftarrow \frac{1}{|D_k|} \sum_{(x,y) \in D_k} \ell(y^i, g^{-k}(x))$

**end for**

**output** $R_{K-CV} = \frac{1}{K} \sum_{k=1}^{n} r^k$ (K-fold cross-validation risk)

**Observation.**

- for $K = |D|$ same as leave-one-out error.
- approximately $k$ times increase in runtime.
- most common: $k = 10$ or $k = 5$.

**Problem:** training sets overlap, so the error estimates are correlated.

**Exception:** $K = 2$
5 × 2 Cross Validation (5 × 2-CV)

**input**  algorithm \( A \), loss function \( \ell \), data \( \mathcal{D} \) (trnval part)

```
for \( k = 1, \ldots, 5 \) do
    Split \( \mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \)
    \( g_1 \leftarrow A[\mathcal{D}_1] \),
    \( r^k_1 \leftarrow \text{evaluate } g_1 \text{ on } \mathcal{D}_2 \)
    \( g_2 \leftarrow A[\mathcal{D}_2] \),
    \( r^k_2 \leftarrow \text{evaluate } g_2 \text{ on } \mathcal{D}_1 \)
    \( r^k \leftarrow \frac{1}{2}(r^1_k + r^2_k) \)
end for
```

**output**  \( \mathcal{R}_{5 \times 2} = \frac{1}{5} \sum_{k=1}^{5} r^k \)

Observation.

- 5 × 2-CV is really the average of 5 runs of 2-fold CV
- very easy to implement: shuffle the data and split into halves
- within each run the training sets are disjoint and the classifiers \( g_1 \) and \( g_2 \) are independent

Problem:  training sets are smaller than in 5- or 10-fold CV.
If classes are imbalanced accuracy might not tell us much:

- \( p(y = -1) = 0.99, p(y = +1) = 0.01 \rightarrow \) "always no" is 99% correct
- there might not be a better non-constant classifier

Three "solutions":

- balancing
  - use only subset of the majority class to balance data (5:1, or 1:1)
- reweighting
  - multiple loss in optimization with class-dependent constant \( C_{y_i} \),

\[
\frac{1}{|D_+|} \sum_{(x_i, y_i) \in D_+} \ell(y_i, f(x_i)) + \frac{1}{|D_-|} \sum_{(x_i, y_i) \in D_-} \ell(y_i, f(x_i)) + \Omega(f)
\]

- treat as a retrieval problem instead of classification
Some classification tasks are really rather retrieval tasks, e.g.

- database lookup: is an entry $x$ relevant ($y = 1$) or not ($y = -1$)?

A typical property:

- prediction is performed on a fixed database
- we have access to all elements of the test set at the same time
- positives ($y = 1$) are important, negative ($y = -1$) are a nuisance
- we don’t need all decisions, a few correct positives is enough

For a classifier $g(x) = \text{sign} \ f(x)$ with $f(x) : \mathcal{X} \to \mathbb{R}$ (e.g., $f(x) = \langle w, x \rangle$), we interpret $f(x)$ as its confidence.

To produce $K$ positives we return the test samples of highest confidence.

Equivalently, we decide by $g_\theta(x) = \text{sign}(f(x) - \theta)$, for the right $\theta$. 
Retrieval quality is often measure in terms of *precision* and *recall*:

**Definition (Precision, Recall, F-Score)**

For $\mathcal{Y} = \{\pm 1\}$, let $g : \mathcal{X} \rightarrow \mathcal{Y}$ a decision function and $\mathcal{D} = \{(x^1, y^1), \ldots, (x^n, y^n)\} \subset \mathcal{X} \times \mathcal{Y}$ be a database.

Then we define

\[
\text{precision}(g) = \frac{\text{number of test samples with } g(x^j) = 1 \text{ and } y^j = 1}{\text{number of test samples with } g(x^j) = 1}
\]

\[
\text{recall}(g) = \frac{\text{number of test samples with } g(x^j) = 1 \text{ and } y^j = 1}{\text{number of test samples with } y^j = 1}
\]

\[
F\text{-score}(g) = 2 \frac{\text{precision}(g) \cdot \text{recall}(g)}{\text{precision}(g) + \text{recall}(g)}
\]
For different thresholds, $\theta$, we obtain different precision and recall values. They are summarized by a precision-recall curve:

![Precision-Recall Curve](image)

- If pressured, summarize into one number: average precision.
- Curve/value depends on class ratio: higher values for more positives.
For different thresholds, $\theta$, we obtain different precision and recall values. They are summarized by a precision-recall curve:

- If pressured, summarize into one number: **average precision**.
- Curve/value depends on class ratio: higher values for more positives.
A similar role in different context:

**Receiver Operating Characteristic (ROC) Curve**

\[
\text{true-positive-rate}(g) = \frac{\text{number of samples with } g(x^j) = 1 \text{ and } y^j = 1}{\text{number of samples with } y^j = 1}
\]

\[
\text{false-positive-rate}(g) = \frac{\text{number of samples with } g(x^j) = 1 \text{ and } y^j = -1}{\text{number of samples with } y^j = -1}
\]

Summarize into: area under ROC curve (AUC).
A similar role in different context:

**Receiver Operating Characteristic (ROC) Curve**

\[
\text{true-positive-rate}(g) = \frac{\text{number of samples with } g(x^j) = 1 \text{ and } y^j = 1}{\text{number of samples with } y^j = 1}
\]

\[
\text{false-positive-rate}(g) = \frac{\text{number of samples with } g(x^j) = 1 \text{ and } y^j = -1}{\text{number of samples with } y^j = -1}
\]

Summarize into: area under ROC curve (AUC).
A similar role in different context:

**Receiver Operating Characteristic (ROC) Curve**

true-positive-rate\( (g) = \frac{\text{number of samples with } g(x^j) = 1 \text{ and } y^j = 1}{\text{number of samples with } y^j = 1} \)

false-positive-rate\( (g) = \frac{\text{number of samples with } g(x^j) = 1 \text{ and } y^j = -1}{\text{number of samples with } y^j = -1} \)

Random classifier: \( AUC = 0.5 \), regardless of class proportions.