<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</td>
<td>Nov</td>
<td></td>
<td>final project</td>
</tr>
</tbody>
</table>
In the real world, $p(x,y)$ is unknown, but we have a training set $\mathcal{D}$.

**Definition**

Given a training set $\mathcal{D}$, we call it

- a **generative probabilistic approach**: if we use $\mathcal{D}$ to build a model $\hat{p}(x,y)$ of $p(x,y)$, and then define

  $$f(x) := \arg\min_{y \in \mathcal{Y}} \mathbb{E}_{\bar{y} \sim \hat{p}(x,\bar{y})} \ell(\bar{y}, y).$$

- a **discriminative probabilistic approach**: if we use $\mathcal{D}$ to build a model $\hat{p}(y|x)$ of $p(y|x)$ and define

  $$f(x) := \arg\min_{y \in \mathcal{Y}} \mathbb{E}_{\bar{y} \sim \hat{p}(\bar{y}|x)} \ell(\bar{y}, y).$$

- a **decision theoretic approach**: if we use $\mathcal{D}$ to directly search for a classifier $f$ in a hypothesis class $\mathcal{H} \subset \{h : \mathcal{X} \to \mathcal{Y}\}$. 
Empirical Risk Minimization

**Definition**

Given a training set $\mathcal{D} = \{ (x^1, y^1), \ldots, (x^n, y^n) \}$, we call it **empirical risk minimization**, if we find a classifier by minimizing the empirical risk:

$$f := \arg\min_{h \in \mathcal{H}} \hat{\mathcal{R}}(h) \quad \text{for} \quad \hat{\mathcal{R}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(y^i, f(x^i))$$

where $\mathcal{H} \subset \{ h : \mathcal{X} \to \mathcal{Y} \}$ is called the **hypothesis set**.
Empirical Risk Minimization

Definition

Given a training set $\mathcal{D} = \{(x^1, y^1), \ldots, (x^n, y^n)\}$, we call it empirical risk minimization, if we find a classifier by minimizing the empirical risk:

$$f := \arg\min_{h \in \mathcal{H}} \hat{R}(h) \quad \text{for} \quad \hat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(y^i, f(x^i))$$

where $\mathcal{H} \subset \{h : \mathcal{X} \to \mathcal{Y}\}$ is called the hypothesis set.

Examples:

- Least-Squared Regression: $\min_w \sum_i (\langle w, x^i \rangle - y^i)^2$
Empirical Risk Minimization

Definition

Given a training set \( D = \{ (x^1, y^1), \ldots, (x^n, y^n) \} \), we call it **empirical risk minimization**, if we find a classifier by minimizing the empirical risk:

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

where \( \mathcal{H} \subset \{ h : \mathcal{X} \to \mathcal{Y} \} \) is called the hypothesis set.

Examples:

- Least-Squared Regression: \( \min_w \sum_i (\langle w, x^i \rangle - y^i)^2 \)
- Logistic Regression: \( \min_w \sum_i \log(1 + e^{-y^i \langle w, x^i \rangle}) \)
Empirical Risk Minimization

Definition

Given a training set $\mathcal{D} = \{(x^1, y^1), \ldots, (x^n, y^n)\}$, we call it **empirical risk minimization**, if we find a classifier by minimizing the empirical risk:

$$f := \arg\min_{h \in \mathcal{H}} \hat{R}(h) \quad \text{for} \quad \hat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(y^i, f(x^i))$$

where $\mathcal{H} \subset \{h : \mathcal{X} \to \mathcal{Y}\}$ is called the hypothesis set.

Examples:

- Least-Squared Regression: $\min_w \sum_i (\langle w, x^i \rangle - y^i)^2$
- Logistic Regression: $\min_w \sum_i \log(1 + e^{-y^i \langle w, x^i \rangle})$
- SVM: $\min_w \ C \sum_i \max\{0, 1 - y^i \langle w, x^i \rangle\} + \|w\|^2$
Empirical Risk Minimization

Definition

Given a training set \( D = \{ (x^1, y^1), \ldots, (x^n, y^n) \} \), we call it **empirical risk minimization**, if we find a classifier by minimizing the empirical risk:

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

where \( \mathcal{H} \subset \{ h : \mathcal{X} \to \mathcal{Y} \} \) is called the hypothesis set.

Examples:

- Least-Squared Regression: \( \min_w \sum_i (\langle w, x^i \rangle - y^i)^2 \)
- Logistic Regression: \( \min_w \sum_i \log(1 + e^{-y^i \langle w, x^i \rangle}) \)
- SVM: \( \min_w C \sum_i \max\{0, 1 - y^i \langle w, x^i \rangle\} + \|w\|^2 \)

We know that for any fixed \( h \), \( \hat{R}(h) \) is an unbiased estimate of \( R(h) \).

Does that mean that \( \hat{R}(f) \) is an unbiased estimate of \( R(f) \)?

**No, unfortunately not!**
Empirical Risk Minimization

1) first choose $f : \mathcal{X} \rightarrow \mathcal{Y}$, then observe $\mathcal{D} = \{(x^1, y^1), \ldots, (x^n, y^n)\}$:

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

unbiased, consistent estimator of $\mathcal{R}(f)$

- $Z^i := \ell(y^i, f(x^i))$ are independent random variables

2) first observe $\mathcal{D} = \{(x^1, y^1), \ldots, (x^n, y^n)\}$, then choose $f$ based on $\mathcal{D}$:

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

$\mathbb{E}_\mathcal{D}[\hat{R}(f)] = ???$

- $Z^i := \ell(y^i, f(x^i))$ are not independent, no law of large numbers.

So why would minimizing one be useful for the other?
Relation between training loss and generalization loss

Example: 1D curve fitting
Relation between training loss and generalization loss

Example: 1D curve fitting

![Graph showing the relationship between training loss and generalization loss for a 1D curve fitting problem. The best learned polynomial of degree 2 has a large training loss \( \hat{R} = 8.44 \) and a large generalization loss \( R = 14.64 \).]
Relation between training loss and generalization loss

Example: 1D curve fitting

best learned polynomial of degree 7: small $\hat{\mathcal{R}}$, small $\mathcal{R}$
Relation between training loss and generalization loss

Example: 1D curve fitting

Best learned polynomial of degree 12: small $\hat{R}$, large $R$
We found a model $f_{\theta^*}$ by minimizing the training error $\hat{R}$.

Q: Will its generalization error, $R$, be small?

A: Unfortunately, that is not guaranteed.

---

**Underfitting/Overfitting**

Underfitting

(to some extend) detectable from $\hat{R}$

Overfitting

not detectable from $\hat{R}$!
Where does overfitting come from?

Choosing a predictor based on $\hat{R}$ vs. $R$

Generalization error $R$ for 7 different predictors
Where does overfitting come from?

Choosing a predictors based on $\hat{R}$ vs. $R$

generalization error $R$ for 7 different predictors
Where does overfitting come from?

Choosing a predictors based on $\hat{R}$ vs. $R$

training error $\hat{R}$ for a training set, $S$
Where does overfitting come from?

Choosing hypothesis based on $\hat{R}$ vs. $R$

- $R(\theta_i)$
- $\hat{R}_{S_1}(\theta_i)$
- $\hat{R}_{S_2}(\theta_i)$
- $\hat{R}_{S_3}(\theta_i)$
- $\hat{R}_{S_4}(\theta_i)$
- $\hat{R}_{S_5}(\theta_i)$

training errors $\hat{R}$ for 5 possible training sets
Where does overfitting come from?

Choosing hypothesis based on $\hat{R}$ vs. $R$

A model with smallest training error can have high generalization error.
Preventing Overfitting
Reminder: Overfitting

How can we prevent overfitting when learning a model?
Preventing overfitting

1) larger training set
→ smaller variance of $\hat{R}_{11}/40$
Preventing overfitting 1) larger training set

Choosing hypothesis based on $\hat{R}$ vs. $R$

larger training set $\rightarrow$ smaller variance of $\hat{R}$
Preventing overfitting 1) larger training set

Choosing hypothesis based on $\hat{R}$ vs. $R$

Lower probability that $\hat{R}$ differs strongly from $R$
Preventing overfitting

1) larger training set

Choosing hypothesis based on $\hat{R}$ vs. $R$

lower probability that $\hat{R}$ differs strongly from $R \rightarrow$ overfitting less likely
Preventing overfitting 2) reduce the number of hypotheses

Choosing hypothesis based on \( \hat{R} \) vs. \( R \)

- \( R(\theta_i) \)
- \( \hat{R}_{S_1}(\theta_i) \)
- \( \hat{R}_{S_2}(\theta_i) \)
- \( \hat{R}_{S_3}(\theta_i) \)
- \( \hat{R}_{S_4}(\theta_i) \)
- \( \hat{R}_{S_5}(\theta_i) \)
Preventing overfitting 2) reduce the number of hypotheses

Choosing a predictors based on $\hat{R}$ vs. $R$

$R(\theta_i)$
Preventing overfitting 2) reduce the number of hypotheses

Choosing hypothesis based on $\hat{R}$ vs. $R$

$R(\theta_i)$
Preventing overfitting 2) reduce the number of hypotheses

Choosing hypothesis based on $\hat{R}$ vs. $R$

Fewer models $\rightarrow$ lower probability of a model with small $\hat{R}$ but high $R$
Preventing overfitting 2) reduce the number of hypotheses

Choosing hypothesis based on $\hat{R}$ vs. $R$

fewer models $\rightarrow$ lower probability of a model with small $\hat{R}$ but high $R$. 

$\theta_i$

1

2

3

0.0

0.2

0.4

0.6

0.8

1.0
But: danger of underfitting
But: danger of underfitting

Choosing hypothesis based on $\hat{R}$ vs. $R$

$\theta_i$

0.0
0.2
0.4
0.6
0.8
1.0

Choosing hypothesis based on $\hat{R}$ vs. $R$

$R(\theta_i)$

to few models select to from $\rightarrow$ danger that no model with low $R$ is left!
But: danger of underfitting

Choosing hypothesis based on \( \hat{R} \) vs. \( R \)

\[
\begin{array}{c|c|c|c|c|c}
\theta_i & 0.0 & 0.2 & 0.4 & 0.6 & 0.8 & 1.0 \\
\hline
\hat{R} & 0.8 & 0.6 & 0.4 & 0.2 & 0.0 & 0.0 \\
R & 0.8 & 0.6 & 0.4 & 0.2 & 0.0 & 0.0 \\
\end{array}
\]

→ danger that no model with low \( R \) is left!

To few models select to from
But: danger of underfitting

Choosing hypothesis based on $\hat{R}$ vs. $R$

Underfitting!
But: danger of underfitting

Choosing hypothesis based on $\hat{R}$ vs. $R$

$R(\theta_i)$
$\hat{R}_{S_1}(\theta_i)$
$\hat{R}_{S_2}(\theta_i)$
$\hat{R}_{S_3}(\theta_i)$
$\hat{R}_{S_4}(\theta_i)$
$\hat{R}_{S_5}(\theta_i)$

Underfitting!
Overfitting happens when . . .

- there are too many models to choose from (not strictly true: there’s usually infinitely many models anyway)
- the models we search over are too "flexible", so they fit not only the signal but also the noise (not strictly true: the models themselves are not "flexible" at all)
- the models have too many free parameters (not strictly true: even models with very few parameters can overfit)

How to avoid overfitting? Use a model class that is

- "as simple as possible", but
- still contains a model with low $\hat{R}$
Regularization
Models with big difference between training error and generalization error are typically **extreme cases**:

- a large number of model parameters
- large values of the model parameters
- for polynomials: high degree, etc.

Coeffs: $\theta_i \in [-2.4, 4.6]$  
Coeffs: $\theta_i \in [-1312.5, 1136.6]$
Regularization

Models with big difference between training error and generalization error are typically **extreme cases**:  
- a large number of model parameters  
- large values of the model parameters  
- for polynomials: high degree, etc.

Regularization: avoid overfitting by preventing extremes to occur  
- explicit regularization (changing the objective function)  
- implicit regularization (modifying the optimization procedure)
Explicit regularization

Add a regularization term (=regularizer) to the empirical risk that gives large values to extreme parameter choices.

Regularized risk minimization

Take a training set, \( S = \{(x^1, y^1), \ldots, (x^n, y^n)\} \), find \( \theta^* \) by solving,

\[
\min_{\theta} J_\lambda(\theta) \quad \text{with} \quad J_\lambda(\theta) = \sum_{i=1}^{n} \ell(y^i, f_\theta(x^i)) + \lambda \Omega(\theta)
\]

- empirical risk
- regularizer

\[\text{e.g. with} \quad \Omega(\theta) = \|\theta\|_2^2 = \sum_j \theta_j^2 \quad \text{or} \quad \Omega(\theta) = \|\theta\|_1 = \sum_j |\theta_j|\]
Explicit regularization

Add a **regularization term** (＝regularizer) to the empirical risk that gives large values to extreme parameter choices.

**Regularized risk minimization**

Take a training set, $S = \{(x^1, y^1), \ldots, (x^n, y^n)\}$, find $\theta^*$ by solving,

$$\min_{\theta} J_\lambda(\theta) \quad \text{with} \quad J_\lambda(\theta) = \sum_{i=1}^{n} \ell(y^i, f_\theta(x^i)) + \lambda \Omega(\theta)$$

\[\begin{align*}
\text{empirical risk} & \quad \text{regularizer}
\end{align*}\]

e.g. with $\Omega(\theta) = \|\theta\|_{L2}^2 = \sum_j \theta_j^2$ or $\Omega(\theta) = \|\theta\|_{L1} = \sum_j |\theta_j|$

Optimization looks for model with small empirical risk, but also small absolute values of the model parameters.

**Regularization (hyper)parameter** $\lambda \geq 0$: trade-off between both.

- $\lambda = 0$: empirical risk minimization (risk of overfitting)
- $\lambda \to \infty$: all parameters 0 (risk of underfitting)
Explicit regularization

Add a **regularization term** (=regularizer) to the empirical risk that gives large values to extreme parameter choices.

### Regularized risk minimization

Take a training set, \( S = \{(x^1, y^1), \ldots, (x^n, y^n)\} \), find \( \theta^* \) by solving,

\[
\min_{\theta} J_\lambda(\theta) \quad \text{with} \quad J_\lambda(\theta) = \sum_{i=1}^{n} \ell(y^i, f_\theta(x^i)) + \lambda \Omega(\theta)
\]

where

- \( \ell(\cdot, \cdot) \) is the empirical risk
- \( \Omega(\cdot) \) is the regularizer

e.g. with \( \Omega(\theta) = \|\theta\|_{L^2}^2 = \sum_j \theta_j^2 \) or \( \Omega(\theta) = \|\theta\|_{L^1} = \sum_j |\theta_j| \)

### Examples:

- **Ridge Regression:** \( \min_{w} \quad \lambda \|w\|^2 + \sum_i (\langle w, x^i \rangle - y^i)^2 \)
- **Logistic Regression:** \( \min_{w} \quad \lambda \|w\|^2 + \sum_i \log(1 + e^{-y^i \langle w, x^i \rangle}) \)
- **SVM:** \( \min_{w} \quad \|w\|^2 + C \sum_i \max\{0, 1 - y^i \langle w, x^i \rangle\} \)
Regularization as Trading Off Bias and Variance

Training error, $\hat{R}$, is a noise estimate of the generalization error, $R$

- original risk $\hat{R}$ is unbiased, but variance can be huge
- regularization introduces a bias, but reduces variance
- for $\lambda \to \infty$, the variance goes to 0, but the bias gets very big
Regularization as Trading Off Bias and Variance

Training error, $\hat{R}$, is a noise estimate of the generalization error, $R$

- original risk $\hat{R}$ is unbiased, but variance can be huge
- regularization introduces a bias, but reduces variance
- for $\lambda \to \infty$, the variance goes to 0, but the bias gets very big
Regularization as Trading Off Bias and Variance

Training error, $\hat{R}$, is a noise estimate of the generalization error, $\mathcal{R}$

- original risk $\hat{R}$ is unbiased, but variance can be huge
- regularization introduces a bias, but reduces variance
- for $\lambda \to \infty$, the variance goes to 0, but the bias gets very big
Regularization as Trading Off Bias and Variance

Training error, $\hat{R}$, is a noise estimate of the generalization error, $R$

- original risk $\hat{R}$ is unbiased, but variance can be huge
- regularization introduces a bias, but reduces variance
- for $\lambda \to \infty$, the variance goes to 0, but the bias gets very big
Example: regularized linear least-squared regression

\[ \min_w J_\lambda(w) \quad \text{for} \quad J_\lambda(w) = \sum_{i=1}^{n} (w^\top x^i - y^i)^2 + \lambda \|w\|^2 \]
Example: regularized linear least-squared regression

$$\min_w J_\lambda(w) \quad \text{for} \quad J_\lambda(w) = \sum_{i=1}^{n}(w^\top x^i - y^i)^2 + \lambda \|w\|^2$$

Train/test error for classifier $c(x) = \text{sign}(\langle w, x \rangle)$ from minimizing $J_\lambda$ with varying amounts of regularization:

![Graph showing training error for varying regularization strengths]
Example: regularized linear least-squared regression

\[ \min_w J_\lambda(w) \quad \text{for} \quad J_\lambda(w) = \sum_{i=1}^{n} (w^\top x^i - y^i)^2 + \lambda\|w\|^2 \]

Train/test error for classifier \( c(x) = \text{sign} \langle w, x \rangle \) from minimizing \( J_\lambda \) with varying amounts of regularization:
Example: regularized linear least-squared regression

\[
\min_w J_\lambda(w) \quad \text{for} \quad J_\lambda(w) = \sum_{i=1}^{n} (w^\top x_i - y_i)^2 + \lambda \|w\|^2
\]

Train/test error for classifier \( c(x) = \text{sign} \langle w, x \rangle \) from minimizing \( J_\lambda \) with varying amounts of regularization:

over-fitting

sweet spot

under-fitting

dataset: 737 examples for training, 736 examples for evaluation
Implicit regularization

Numerical optimization is performed iteratively, e.g. gradient descent

**Gradient descent optimization**

- initialize $\theta^{(0)}$
- **for** $t = 1, 2, \ldots$
  - $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla_{\theta} J(\theta^{(t-1)})$ \hspace{1cm} ($\eta_t \in \mathbb{R}$ is some stepsize rule)
- **until** convergence

**Implicit regularization** methods modify these steps, e.g.

- early stopping
- weight decay
- data jittering
- dropout
Gradient descent optimization with **early stopping**

- initialize $\theta^{(0)}$
- **for** $t = 1, 2, \ldots, T$ \hspace{1cm} ($T \in \mathbb{N}$ is number of steps)
- $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla_\theta J(\theta^{(t-1)})$
Implicit regularization: early stopping

Gradient descent optimization with **early stopping**

- initialize $\theta^{(0)}$
- **for** $t = 1, 2, \ldots, T$ \quad ($T \in \mathbb{N}$ is number of steps)
  - $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla_{\theta} J(\theta^{(t-1)})$

**Early stopping**: stop optimization before convergence

- idea: if parameters are update only a small number of time, they might not reach extreme values

- $T$ hyperparameter controls trade-off:
  - large $T$: parameters approach risk minimizer $\rightarrow$ risk of overfitting
  - small $T$: parameters stay close to initialization $\rightarrow$ risk of underfitting
Implicit regularization: weight decay

**Gradient descent optimization with weight decay**

- initialize $\theta^{(0)}$
- **for** $t = 1, 2, \ldots$
  - $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla \theta J(\theta^{(t-1)})$
  - $\theta^{(t)} \leftarrow \gamma \theta^{(t)}$ for, e.g., $\gamma = 0.99$
- **until** convergence

Note: essentially same effect as explicit regularization with $\Omega = \gamma^2 \| \theta \|_2^2$
Implicit regularization: weight decay

Gradient descent optimization with weight decay

- initialize $\theta^{(0)}$
- for $t = 1, 2, \ldots$
  - $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla \theta J(\theta^{(t-1)})$
  - $\theta^{(t)} \leftarrow \gamma \theta^{(t)}$ for, e.g., $\gamma = 0.99$
- until convergence

Weight decay:
Multiply parameters with a constant smaller than 1 in each iteration
- two 'forces' in parameter update:
  - $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla \theta J(\theta^{(t-1)})$
    - pull towards empirical risk minimizer $\rightarrow$ risk of overfitting
  - $\theta^{(t)} \leftarrow \gamma \theta^{(t)}$ pulls towards 0 $\rightarrow$ risk of underfitting
- convergence: both effects cancel out $\rightarrow$ trade-off controlled by $\eta_t, \gamma$

Note: essentially same effect as explicit regularization with $\Omega = \frac{\gamma}{2} \| \theta \|^2$
Implicit regularization: data jittering ("virtual samples")

Gradient descent optimization with data jittering

• initialize $\theta^{(0)}$
• for $t = 1, 2, \ldots$
  • for $i = 1, \ldots, n$:
    • $\tilde{x}^i \leftarrow$ randomly perturbed version of $x^i$
    • set $\tilde{J}(\theta) = \sum_{i=1}^{n} \ell(y^i, f_{\theta}(\tilde{x}^i))$
    • $\theta(t) \leftarrow \theta(t-1) - \eta_t \nabla_{\theta} \tilde{J}(\theta(t-1))$
• until convergence

Jittering:
• idea: a good model should be robust to small changes of the data
• simulate (infinitely-)large training set → hopefully less overfitting
• problem: coming up with perturbations needs domain knowledge
Implicit regularization: data jittering (="virtual samples")

Gradient descent optimization with data jittering

- initialize $\theta^{(0)}$
- for $t = 1, 2, \ldots$
  - for $i = 1, \ldots, n$:
    - $\tilde{x}^i \leftarrow$ randomly perturbed version of $x^i$
    - set $\tilde{J}(\theta) = \sum_{i=1}^{n} \ell(y^i, f_\theta(\tilde{x}^i))$
    - $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla_\theta \tilde{J}(\theta^{(t-1)})$
- until convergence

**Jittering:** use randomly perturbed examples in each iteration
- idea: a good model should be robust to small changes of the data
- simulate (infinitely-)large training set $\rightarrow$ hopefully less overfitting
  (also possible: just create large training set of jittered examples in the beginning)
- problem: coming up with perturbations needs domain knowledge
Implicit regularization: dropout

Gradient descent optimization with dropout

- initialize $\theta^{(0)}$
- **for** $t = 1, 2, \ldots$
- $\tilde{\theta} \leftarrow \theta^{(t-1)}$ with a random fraction $p$ of values set to 0, e.g. $p = \frac{1}{2}$
- $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla_{\theta} J(\tilde{\theta})$
- **until** convergence

Dropout: every time we evaluate the model, a random subset of its parameters are set to zero.

- aims for model with low empirical risk even if parameters are missing
- idea: no single parameter entry can become 'too important'
- similar to jittering, but without need for domain knowledge about $x$'s
- overfitting vs. underfitting tradeoff controlled by $p$
Implicit regularization: dropout

Gradient descent optimization with dropout

- initialize $\theta^{(0)}$
- for $t = 1, 2, \ldots$
  - $\tilde{\theta} \leftarrow \theta^{(t-1)}$ with a random fraction $p$ of values set to 0, e.g. $p = \frac{1}{2}$
  - $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_t \nabla_{\theta} J(\tilde{\theta})$
- until convergence

 Dropout: every time we evaluate the model, a random subset of its parameters are set to zero.

- aims for model with low empirical risk even if parameters are missing
- idea: no single parameter entry can become 'too important'
- similar to jittering, but without need for domain knowledge about $x$’s
- overfitting vs. underfitting tradeoff controlled by $p$
Often, more than one regularization techniques are combined, e.g.

Explicit regularization: e.g. "elastic net"
- $\Omega(\theta) = \alpha \|\theta\|_{L2}^2 + (1 - \alpha)\|\theta\|_{L1}$

Explicit/implicit regularization: e.g. large-scale support vector machines
- $\Omega(\theta) = \|\theta\|_{L2}^2$, early stopping, potentially jittering

Implicit regularization: e.g. deep networks
- early stopping, weight decay, dropout, potentially jittering
Regularization can prevent overfitting

Intuition: avoid "extreme" models, e.g. very large parameter values

Explicit Regularization: modify object function

Implicit Regularization: change optimization procedure

Regularization introduces additional (hyper)parameters

How much of a regularization method to apply is a free parameter, often called regularization constant. The optimal values are problem specific.
Understanding the test error from the training error

Generalization Bound

For every $f \in H$, it holds:

$$E_{(x,y)} \ell(y, f(x)) \leq \frac{1}{n} \sum_i \ell(y_i, f(x_i)) + \text{something}$$

Image: http://typemoon.wikia.com/
The Holy Grail of Statistical Machine Learning

Understanding the test error from the training error

Generalization Bound

For every $f \in \mathcal{H}$ it holds:

$$
\mathbb{E}_{(x,y)} \ell(y, f(x)) \leq \frac{1}{n} \sum_i \ell(y_i, f(x_i)) + \text{something}
$$

- $\mathbb{E}_{(x,y)} \ell(y, f(x))$ generalization loss
- $\frac{1}{n} \sum_i \ell(y_i, f(x_i))$ training loss
Typical structure of a generalization bound

Fixed learning setting:

- input data $\mathcal{X}$, output space $\mathcal{Y}$,
- data distribution $p$ over $\mathcal{X} \times \mathcal{Y}$ (with some properties),
- hypothesis set $\mathcal{H} \subset \{ f : \mathcal{X} \to \mathcal{Y} \}$,
- loss function, $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ (with some properties),

For any $\delta > 0$, the following statement holds with probability at least $1 - \delta$ over the (random) training set $\mathcal{D}_n = \{ (x_1, y_1), \ldots, (x_n, y_n) \} \overset{i.i.d.}{\sim} p$.

For all $f \in \mathcal{H}$:

$$\mathbb{E}_{(x,y)} \ell(y, f(x)) \leq \frac{1}{n} \sum_{i=1}^n \ell(y, f(x)) + \text{something}$$

"something" typically increases for $\delta \to 0$ and decreases for $n \to \infty$.

Observation: if inequality holds, it holds uniformly for all $f$.

$\rightarrow$ by minimizing the right hand side, we can find the "most promising" $f$
Reminder: (soft-margin) support vector machine (SVM):

$$\min_w \frac{\lambda}{2} \|w\|^2 + \frac{1}{m} \sum_i \max\{0, 1 - y_i \langle w, x_i \rangle\}$$
Reminder: (soft-margin) support vector machine (SVM):

$$\min_w \frac{\lambda}{2} \|w\|^2 + \frac{1}{m} \sum_{i} \max\{0, 1 - y_i \langle w, x_i \rangle\}$$

### Example: SVM radius/margin bound

Let $\ell(x,y;w) := \max\{0, 1 - y \langle w, x \rangle\}$ be the hinge loss. Let $p$ be a distribution on $\mathbb{R}^d \times \mathcal{Y}$ such that $\Pr\{\|x\| \leq R\} = 1$ and let $\mathcal{H} = \{w : \|w\| \leq B\}$.

Then, with prob. at least $1 - \delta$ over $\mathcal{D}_m \overset{i.i.d.}{\sim} p$ the following inequality holds for all $w \in \mathcal{H}$:

$$\mathbb{E}_{(x,y) \sim p} \left[ \langle w, x \rangle \neq y \right] \leq \frac{1}{m} \sum_{i=1}^{m} \ell(x_i, y_i, w) + \frac{2BR}{\sqrt{m}} + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}.$$  

### Properties:

- uniform in $w$, i.e. holds even for minimizer of r.h.s. $\rightarrow$ almost SVM
- $B$ is a upper bound on $\|w\| \rightarrow$ small $\|w\|$ are most promising
- dimensionality of $x$ does not show up, no curse of dimensionality!
Excurse: Concentration of Measure II
Lemma (Hoeffding’s Lemma)

Let $Z$ be a random variable that takes values in $[a, b]$ and $\mathbb{E}[Z] = 0$. Then, for every $\lambda > 0$,

$$\mathbb{E}[e^{\lambda X}] \leq e^{\frac{\lambda^2 (b-a)^2}{8}}.$$  

Proof: Exercise...
Lemma (Hoeffding’s Inequality)

Let $Z_1, \ldots, Z_m$ be i.i.d. random variables that take values in the interval $[a, b]$. Let $\bar{Z} = \frac{1}{m} \sum_{i=1}^{m} Z_i$ and denote $\mathbb{E}[\bar{Z}] = \mu$. Then, for any $\epsilon > 0$,

$$
P \left[ \left( \frac{1}{m} \sum_{i=1}^{m} Z_i - \mu \right) > \epsilon \right] \leq e^{-m \frac{\epsilon^2}{(b-a)^2}}.
$$

and

$$
P \left[ \left( \mu - \frac{1}{m} \sum_{i=1}^{m} Z_i \right) > \epsilon \right] \leq e^{-m \frac{\epsilon^2}{(b-a)^2}}.
$$

and

$$
P \left[ \left| \frac{1}{m} \sum_{i=1}^{m} Z_i - \mu \right| > \epsilon \right] \leq 2e^{-m \frac{\epsilon^2}{(b-a)^2}}.$$
Hoeffding’s Inequality – Proof

Define new RVs: \( X_i = Z_i - \mathbb{E}[Z_i] \), \( \bar{X} = \frac{1}{m} \sum_i X_i \)

- \( \mathbb{E}[X_i] = 0 \); \( \mathbb{E}[\bar{X}] = 0 \); each \( X_i \) takes values in \([a - \mathbb{E}[Z_i], b - \mathbb{E}[Z_i]]\)

Use 1) monotonicity of \( \exp \) and 2) Markov’s inequality to check

\[
P[\bar{X} \geq \epsilon] = \mathbb{P}[e^{\lambda \bar{X}} \geq e^{\lambda \epsilon}] \leq e^{-\lambda \epsilon} \mathbb{E}[e^{\lambda \bar{X}}]
\]

From 3) the independence of the \( X_i \) we have

\[
\mathbb{E}[e^{\lambda \bar{X}}] = \mathbb{E}[\prod_{i=1}^n e^{\lambda X_i/m}] = \prod_{i=1}^n \mathbb{E}[e^{\lambda X_i/m}]
\]

Use 4) Hoeffding’s Lemma for every \( i \):

\[
\mathbb{E}[e^{\lambda X_i/m}] \leq e^{\frac{\lambda^2 (b-a)^2}{8m^2}}
\]

In combination:

\[
P[\bar{X} \geq \epsilon] \leq e^{-\lambda \epsilon} e^{\frac{\lambda^2 (b-a)^2}{8m}}
\]
Hoeffding’s Inequality – Proof cont.

Previous step:

\[ \mathbb{P}[\bar{X} \geq \epsilon] \leq e^{-\lambda \epsilon} e^{\frac{\lambda^2 (b-a)^2}{8m}} \]

So far, \( \lambda \) was arbitrary. Now we set \( \lambda = \frac{4m \epsilon}{(b-a)^2} \)

\[ \mathbb{P}[\bar{X} \geq \epsilon] \leq e^{-\frac{4m \epsilon}{(b-a)^2} \epsilon + \left(\frac{4m \epsilon}{(b-a)^2}\right)^2 \frac{(b-a)^2}{8m}} = e^{-\frac{2m \epsilon^2}{(b-a)^2}} \]

This proves the first statement.

If we repeat the same steps again for \(-\bar{X}\) instead of \(X\), we get

\[ \mathbb{P}[\bar{X} \leq -\epsilon] \leq e^{-\frac{2m \epsilon^2}{(b-a)^2}} \]

This proves the second statement.

Use the union bound: \( \mathbb{P}[A \lor B] \leq \mathbb{P}[A] + \mathbb{P}[B] \), to combine both directions:

\[ \mathbb{P}[|\bar{X}| \geq \epsilon] = \mathbb{P}[ (\bar{X} \geq \epsilon) \lor (\bar{X} \leq -\epsilon) ] \leq 2e^{-\frac{2m \epsilon^2}{(b-a)^2}}. \]
How large should my test set be?

\[
P\left[ \left| \frac{1}{m} \sum_{i=1}^{m} Z_i - \mu \right| > \epsilon \right] \leq 2e^{-\frac{2m\epsilon^2}{(b-a)^2}}.
\]

Setup: fixed classifier \( g : \mathcal{X} \rightarrow \mathcal{Y} \)

- test set \( \mathcal{D} = \{(x^1, y^1), \ldots, (x^m, y^m)\} \) \( \text{i.i.d.} \) \( p(x, y) \),
- random variables \( Z_i = \mathbb{I}[g(x^i) \neq y^i] \in \{0, 1\}, \rightarrow b - a = 1 \)
- \( \mathbb{E}[Z_i] = \mathbb{E}\{\mathbb{I}[g(x^i) \neq y^i]\} = \mu \) (test error of \( g \))

Setup: \( m = \frac{1}{2} \log\left(\frac{2}{\delta}\right)/\epsilon^2 \).

For fixed confidence \( \delta = 0.1 \Rightarrow \epsilon = \sqrt{\log(20)/(2m)} \approx 1.22 \sqrt{\frac{1}{m}} \)

\[
P\left[ \left| \frac{1}{m} \sum_{i=1}^{m} Z_i - \mu \right| \leq 1.22 \sqrt{\frac{1}{m}} \right] \geq 0.9
\]

To be 90%-certain that the error is within 0.05, use \( m \geq 600 \).
How large should my test set be?

\[ \Pr \left[ \left| \frac{1}{m} \sum_{i=1}^{m} Z_i - \mu \right| > \epsilon \right] \leq 2e^{-\frac{2m\epsilon^2}{(b-a)^2}}. \]

Setup: fixed classifier \( g : \mathcal{X} \to \mathcal{Y} \)

- test set \( \mathcal{D} = \{(x^1, y^1), \ldots, (x^m, y^m)\} \ i.i.d. \sim p(x, y) \),
- random variables \( Z_i = [g(x^i) \neq y^i] \in \{0, 1\}, \quad b - a = 1 \)
- \( \mathbb{E}[Z_i] = \mathbb{E}\{[g(x^i) \neq y^i]\} = \mu \)  (test error of \( g \))

Setup: \( m = \frac{1}{2} \log(\frac{2}{\delta})/\epsilon^2 \).

For fixed confidence \( \delta = 0.1 \Rightarrow \epsilon = \sqrt{\log(20)/(2m)} \approx 1.22\sqrt{\frac{1}{m}} \)

\[ \Pr \left[ \left| \frac{1}{m} \sum_{i=1}^{m} Z_i - \mu \right| \leq 1.22\sqrt{\frac{1}{m}} \right] \geq 0.9 \]

To be 90%-certain that the error is within 0.05, use \( m \geq 600 \).
To be 99%-certain that the error is within 0.05, use \( m \geq 1060 \).
To be 90%-certain that the error is within 0.005, use \( m \geq 59914 \).
Difference: Chebyshev’s vs. Hoeffding’s Inequality

With $\hat{R} = \frac{1}{m} \sum_{i=1}^{m} Z_i$ and $R = \mathbb{E}[\frac{1}{m} \sum_{i=1}^{m} Z_i]$: 

- **Chebyshev’s:** $\text{Var}[Z_i] \leq C$

  $$\mathbb{P}\left[ |\hat{R} - R| > \sqrt{\frac{C}{\delta m}} \right] \leq \delta, \quad \mathbb{P}\left[ |\hat{R} - R| > \epsilon \right] \leq \frac{C}{\epsilon^2 m}$$

  - interval decreases like $\frac{1}{\sqrt{m}}$, confidence grows like $1 - \frac{1}{m}$

- **Hoeffding’s:** $Z_i$ takes values in $[a, b]$

  $$\mathbb{P}\left[ |\hat{R} - R| > \sqrt{\frac{(b-a)^2 \log \frac{2}{\delta}}{m}} \right] \leq \delta, \quad \mathbb{P}\left[ |\hat{R} - R| > \epsilon \right] \leq 2e^{-\frac{2m\epsilon^2}{(b-a)^2}}.$$  

  - interval decreases like $\frac{1}{\sqrt{m}}$, confidence grows like $1 - e^{-m}$

Both are typical **PAC (probably approximately correct)** statements:

“With **prob.** $1 - \delta$, the estimated $\hat{R}$ is an $\epsilon$-close approximation of $R$.”
Back to Machine Learning
Classical Generalization Bounds

Finite Hypothesis Set

Setup:
- \( \ell(y, \bar{y}) = \mathbb{1}[y \neq \bar{y}] \) (0-1 loss)
- finite number of possible classifiers \( \mathcal{H} = \{f_1, \ldots, f_T\} \subset \mathcal{Y}^X \)

For any \( \delta > 0 \), the following statement holds with probability at least \( 1 - \delta \) over the training set \( \mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \overset{i.i.d.}{\sim} p(x, y) \):

For all \( f \in \mathcal{H} \):

\[
\mathcal{R}(f) \leq \hat{\mathcal{R}}(f) + \sqrt{\frac{\log |\mathcal{H}| + \log 1/\delta}{2n}}
\]

Proof: blackboard...
Classical Generalization Bounds

Proof.
1) For any fixed $f \in \mathcal{H}$, we get from Hoeffding’s inequality:

$$
P[\mathcal{R}(f) - \hat{\mathcal{R}}(f) > \epsilon] \leq e^{-2n\epsilon^2}.
= C_f
$$

2) By a union bound, $P[\bigvee_{f \in \mathcal{H}} C_f] \leq \sum_{f \in \mathcal{H}} P[C_f]$, we obtain

$$
P[\exists f \in \mathcal{H} : \mathcal{R}(f) > \hat{\mathcal{R}}(f) + \epsilon] \leq |\mathcal{H}|e^{-2n\epsilon^2}.
$$

3) Right hand side should be $\delta$, solve for $\epsilon$:

$$
\epsilon = \sqrt{\frac{\log(\frac{\mathcal{H}}{\delta})}{2n}}
$$

4) Put together, using that

$$
P[\forall f \in \mathcal{H} : \mathcal{R}(f) \leq \hat{\mathcal{R}}(f) + \epsilon] = 1 - P[\exists f \in \mathcal{H} : \mathcal{R}(f) > \hat{\mathcal{R}}(f) + \epsilon]
$$
Examples: Finite hypothesis classes

Model selection:
- Clients offer me trained classifiers: 1) decision tree, 2) LogReg or an 3) SVM? Which of the three should I buy?

Finite precision:
- For \( X \subset \mathbb{R}^d \), the hypothesis set \( \mathcal{H} = \{ f(x) = \text{sign} \langle w, x \rangle \} \) is infinite.
- But: on a computer with \( w \) restricted to 32-bit floats: \( |\mathcal{H}| = 2^{32d} \). \( \log |\mathcal{H}| \approx 22d \)

Implementation:
- \( \mathcal{H} = \{ \text{all algorithms implementable in 1MB C-code} \} \) is finite.

Logarithmic dependence on \( |\mathcal{H}| \) makes even large (finite) hypothesis sets (kind of) practical.