Recognition II
General recipe

• Fix **hypothesis class**
  \[ h(x; \mathbf{w}, b) = \sigma(\mathbf{w}^T \phi(x) + b) \]

• Define **loss function**
  \[ L(h(x; \mathbf{w}, b), y) = -y \log h(x; \mathbf{w}, b) + (1 - y) \log(1 - h(x; \mathbf{w}, b)) \]

• **Minimize total loss** on the training set
  \[ \min_{\mathbf{w}, b} \sum_{i=1}^{N} L(h(x_i; \mathbf{w}, b), y_i) \]

• **Why should this work?**

• **How do we do the minimization in practice?**
Training = Optimization

• Need to minimize an objective

\[ \min_{\mathbf{w}, \mathbf{b}} \sum_{i=1}^{N} L(\mathbf{h}(x_i; \mathbf{w}, b), y_i) \]

• More generally, objective takes the form

\[ \min_{\theta} \sum_{i=1}^{N} f(x_i, y_i, \theta) \equiv \min_{\theta} F(\theta) \]
Training = optimization

\[
\min_{\theta} \sum_{i=1}^{N} f(x_i, y_i, \theta) \equiv \min_{\theta} F(\theta)
\]

• How do we minimize this?
• Start from an initial estimate
• Iteratively reduce F. How?
Optimization and function gradients

- Suppose current estimate is $\theta^{(t)}$
- Consider changing this to $\theta^{(t)} + \Delta \theta$
- How does the objective value change?
- For small $\Delta \theta$, can approximate $F$ using Taylor expansion
  - $F$ is locally linear

\[
F(\theta^{(t)} + \Delta \theta) \approx F(\theta^{(t)}) + \nabla F(\theta^{(t)})^T \Delta \theta
\]

\[
\Rightarrow F(\theta^{(t)} + \Delta \theta) - F(\theta^{(t)}) \approx \nabla F(\theta^{(t)})^T \Delta \theta
\]
Optimization and function gradients

\[ F(\theta^{(t)} + \Delta \theta) - F(\theta^{(t)}) \approx \nabla F(\theta^{(t)})^T \Delta \theta \]

- We want \( F(\theta^{(t)} + \Delta \theta) - F(\theta^{(t)}) \) to be negative
  - As highly negative as possible
- So we want \( \nabla F(\theta^{(t)})^T \Delta \theta \) to be as negative as possible

\[ \Delta \theta = -\lambda \nabla F(\theta^{(t)}) \]

\[ \Rightarrow \nabla F(\theta^{(t)})^T \Delta \theta = -\lambda \| \nabla F(\theta^{(t)}) \|^2 \]

- \( \lambda \) is step size
Optimization using gradient descent

• Randomly initialize $\theta^{(0)}$

• For $i = 1$ to max_iterations:
  • Compute gradient of $F$ at $\theta^{(t)}$
  • $\theta^{(t+1)} \leftarrow \theta^{(t)} - \lambda \nabla F(\theta^{(t)})$
    • Function value will decrease by $\lambda \| \nabla F(\theta^{(t)}) \|^2$
  • Repeat until $\| \nabla F(\theta^{(t)}) \|^2$ drops below a threshold
Gradient descent

\[ z = x^2 + 2y^2 \]

https://yihui.name(animation/example/grad-desc/)
Gradient descent - convergence

• Every step leads to a reduction in the function value
• If function is bounded below, we will eventually stop
• But will we stop at the right “global minimum”?
  • Not necessarily: local optimum!
Gradient descent in machine learning

\[ \min_{\theta} \sum_{i=1}^{N} f(x_i, y_i, \theta) \equiv \min_{\theta} F(\theta) \]

\[ \nabla F(\theta) = \sum_{i=1}^{N} \nabla f(x_i, y_i, \theta) \]

- Computing the gradient requires a \textit{loop over all training examples}
- Very expensive for large datasets
Stochastic gradient descent

\[
\nabla F(\theta) = \sum_{i=1}^{N} \nabla f(x_i, y_i, \theta)
\]

\[
\nabla F(\theta) \approx \sum_{j=1}^{K} \nabla f(x_{ij}, y_{ij}, \theta)
\]

- Randomly sample small subset of examples
- Compute gradient on small subset
  - *Unbiased estimate of true gradient*
- Take step along estimated gradient
General recipe

- Fix hypothesis class
  \[ h(x; w, b) = \sigma(w^T \phi(x) + b) \]
- Define loss function
  \[ L(h(x; w, b), y) = -y \log h(x; w, b) + (1 - y) \log(1 - h(x; w, b)) \]
- Minimize total loss on the training set using SGD
  \[ \min_{w, b} \sum_{i=1}^{N} L(h(x_i; w, b), y_i) \]
General recipe

• Fix hypothesis class
  \[ h(x; w, b) = \sigma(w^T \phi(x) + b) \]

• Define loss function
  \[ L(h(x; w, b), y) = -y \log h(x; w, b) + (1 - y) \log(1 - h(x; w, b)) \]

• Minimize total loss on the training set using SGD
  \[ \min_{w, b} \sum_{i=1}^{N} L(h(x_i; w, b), y_i) \]

• Why should this work?
Why should this work?

• Let us look at the objective more carefully

\[
\min_{\mathbf{w}, \mathbf{b}} \sum_{i=1}^{N} L(h(x_i; \mathbf{w}, \mathbf{b}), y_i)
\]

\[
\equiv \min_{\mathbf{w}, \mathbf{b}} \frac{1}{N} \sum_{i=1}^{N} L(h(x_i; \mathbf{w}, \mathbf{b}), y_i)
\]

• We are basically minimizing average loss on the training set
• Is this what we actually care about?
Risk

• Given:
  • Distribution $\mathcal{D}$ over (x,y) pairs
  • A hypothesis $h \in H$ from hypothesis class H
  • Loss function $L$

• We are interested in Expected Risk:
  $$R(h) = \mathbb{E}_{(x,y) \sim \mathcal{D}} L(h(x), y)$$

• Given training set $S$, and a particular hypothesis $h$, Empirical Risk:
  $$\hat{R}(S, h) = \frac{1}{|S|} \sum_{(x,y) \in S} L(h(x), y)$$
Risk

\[ R(h) = \mathbb{E}_{(x,y) \sim \mathcal{D}} L(h(x), y) \]

\[ \hat{R}(S, h) = \frac{1}{|S|} \sum_{(x,y) \in S} L(h(x), y) \]

- Left: true quantity of interest, right: estimate
- How good is this estimate?
- If \( h \) is \textit{randomly chosen}, actually a pretty good estimate!
  - In statistics-speak, it is an \textit{unbiased estimator}: correct in expectation

\[ \mathbb{E}_{S \sim \mathcal{D}^n} \hat{R}(S, h) = R(h) \]
Risk

- Empirical risk unbiased estimate of expected risk
- Want to minimize expected risk
- Idea: Minimize \textit{empirical risk} instead
- This is the \textbf{Empirical Risk Minimization Principle}

\[
R(h) = \mathbb{E}_{(x,y) \sim \mathcal{D}} L(h(x), y) \quad \hat{R}(S, h) = \frac{1}{|S|} \sum_{(x,y) \in S} L(h(x), y)
\]

\[
h^* = \arg \min_{h \in H} \hat{R}(S, h)
\]
Overfitting

- For *randomly chosen* $h$, empirical risk (training error) good estimate of expected risk
- But we are *choosing* $h$ by minimizing training error
- Empirical risk of chosen hypothesis *no longer* unbiased estimate:
  - We chose hypothesis based on $S$
  - Might have chosen $h$ for which $S$ is a special case
- Overfitting:
  - Minimize training error, but generalization error *increases*
Overfitting = fitting the noise

Minimizer of expected risk

True distribution

Minimizer of empirical risk

Sampled training set
Generalization

\[ R(h) = \mathbb{E}_{(x,y) \sim \mathcal{D}} L(h(x), y) \]

\[ \hat{R}(S, h) = \frac{1}{|S|} \sum_{(x,y) \in S} L(h(x), y) \]

\[ R(h) = \hat{R}(S, h) + (R(h) - \hat{R}(S, h)) \]

- Training error
- Generalization error
Controlling generalization error

- Variance of empirical risk inversely proportional to size of $S$
  - Choose very large $S$!

- Larger the hypothesis class $H$, Higher the chance of hitting bad hypotheses with low training error and high generalization error
  - Choose small $H$!

- For many models, can bound generalization error using some property of parameters
  - Regularize during optimization!
  - Eg. L2 regularization
Controlling the size of the hypothesis class

\[ h(x; \mathbf{w}, b) = \sigma(\mathbf{w}^T \phi(x) + b) \]

- How many parameters (\( \mathbf{w}, b \)) are there to find?
- Depends on dimensionality of \( \phi \)
- Large dimensionality = large number of parameters = more chance of overfitting
- Rule of thumb: size of training set should be at least 10x number of parameters
- Often training sets are much smaller
Regularization

• Old objective

\[
\min_{w, b} \sum_{i=1}^{N} L(h(x_i; w, b), y_i)
\]

• New objective

\[
\min_{w, b} \sum_{i=1}^{N} L(h(x_i; w, b), y_i) + \lambda \|w\|^2
\]

• Why does this help?
Regularization

\[
\min_{w, b} \sum_{i=1}^{N} L(h(x_i; w, b), y_i) + \lambda \| w \|^2
\]

- Ensures classifier does not weigh any one feature too highly
- Makes sure classifier scores *vary slowly* when image changes

\[
|w^T \phi(x_1) - w^T \phi(x_2)| \leq \| w \| \| \phi(x_1) - \phi(x_2) \|
\]
Controlling generalization error

• How do we know we are overfitting?
  • Use a held-out “validation set”
  • To be an unbiased sample, must be completely unseen
Putting it all together

• Want model with least expected risk = expected loss
• But expected risk hard to evaluate
• Empirical Risk Minimization: minimize empirical risk in training set
• Might end up picking special case: overfitting
• Avoid overfitting by:
  • Constructing large training sets
  • Reducing size of model class
  • Regularization
Putting it all together

• Collect training set and validation set
• Pick hypothesis class
• Pick loss function
• Minimize empirical risk (+ regularization)
• Measure performance on held-out validation set
• Profit!
Loss functions and hypothesis classes

<table>
<thead>
<tr>
<th>Loss function</th>
<th>Problem</th>
<th>Range of $h$</th>
<th>$Y$</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log loss</td>
<td>Binary Classification</td>
<td>$\mathbb{R}$</td>
<td>${0, 1}$</td>
<td>$\log(1 + e^{-yh(x)})$</td>
</tr>
<tr>
<td>Negative log likelihood</td>
<td>Multiclass classification</td>
<td>$[0, 1]^k$</td>
<td>${1, \ldots, k}$</td>
<td>$- \log h_y(x)$</td>
</tr>
<tr>
<td>Hinge loss</td>
<td>Binary Classification</td>
<td>$\mathbb{R}$</td>
<td>${0, 1}$</td>
<td>$\max(0, 1 - yh(x))$</td>
</tr>
<tr>
<td>MSE</td>
<td>Regression</td>
<td>$\mathbb{R}$</td>
<td>$\mathbb{R}$</td>
<td>$(y - h(x))^2$</td>
</tr>
</tbody>
</table>
Back to images

\[ h(x; w, b) = \sigma(w^T \phi(x) + b) \]

• What should \( \phi \) be?
• Simplest solution: string 2D image intensity values into vector
Linear classifiers on pixels are bad

- Solution 1: Better feature vectors
- Solution 2: Non-linear classifiers
Better feature vectors

These must have similar feature vectors: *invariance*

These must have different feature vectors: *discriminability*
Better feature vectors

• Invariance to
  • Illumination
  • Deformation
  • Translations/rotations
Color and Lighting
Out-of-plane rotation
SIFT

• Match *pattern of edges*
  • Edge orientation – clue to shape

• Be resilient to *small deformations*
  • Deformations might move pixels around, but slightly
  • Deformations might change edge orientations, but slightly
Invariance to deformation by quantization

Between 30 and 45
Invariance to deformation by quantization

\[
g(\theta) = \begin{cases} 
0 & \text{if } 0 < \theta < \frac{2\pi}{N} \\
1 & \text{if } \frac{2\pi}{N} < \theta < \frac{4\pi}{N} \\
2 & \text{if } \frac{4\pi}{N} < \theta < \frac{6\pi}{N} \\
N - 1 & \text{if } 2(N - 1)\frac{\pi}{N} < \theta < \frac{2\pi}{N} \\
& \cdots
\end{cases}
\]
Spatial invariance by histograms

2 blue balls, one red box

![Diagram showing 2 blue balls and 1 red box with a bar chart showing 2 balls and 1 box]
Rotation Invariance by Orientation Normalization

- Compute orientation histogram
- Select dominant orientation
- Normalize: rotate to fixed orientation

[Lowe, SIFT, 1999]
The SIFT descriptor

SIFT – Lowe IJCV 2004
Same but different: HOG

Histogram of oriented gradients
Same as SIFT but without orientation normalization. Why?
Invariance to large deformations
Invariance to large deformations

• Large deformations can cause objects / object parts to move a lot (much more than single grid cell)
• Yet, object parts themselves have precise appearance

• Idea: want to represent the image as a “bag of object parts”