# Gaussian Mixture, EM Algorithm, Bias-Variance Trade-off 

## Milan Straka

甸 December 20, 2021


Charles University in Prague
Faculty of Mathematics and Physics Institute of Formal and Applied Linguistics

unless otherwise stated

Input: Input points $\boldsymbol{x}_{1}, \ldots, \boldsymbol{x}_{N}$, number of clusters $K$.

- Initialize $\boldsymbol{\mu}_{1}, \ldots, \boldsymbol{\mu}_{K}$ as $K$ random input points.
- Repeat until convergence (or until patience runs out):
- Compute the best possible $z_{i, k}$. It is easy to see that the smallest $J$ is achieved by

$$
z_{i, k}= \begin{cases}1 & \text { if } k=\arg \min _{j}\left\|\boldsymbol{x}_{i}-\boldsymbol{\mu}_{j}\right\|^{2} \\ 0 & \text { otherwise }\end{cases}
$$

- Compute the best possible $\boldsymbol{\mu}_{k}=\arg \min _{\boldsymbol{\mu}} \sum_{i} z_{i, k}\left\|\boldsymbol{x}_{i}-\boldsymbol{\mu}\right\|^{2}$. By computing a derivative with respect to $\boldsymbol{\mu}$, we get

$$
\boldsymbol{\mu}_{k}=\frac{\sum_{i} z_{i, k} \boldsymbol{x}_{i}}{\sum_{i} z_{i, k}}
$$



## Gaussian Mixture vs K-Means

It could be useful to consider that different clusters might have different radii or even be ellipsoidal.

## Different cluster analysis results on "mouse" data set: <br> Original Data <br> k-Means Clustering <br> EM Clustering





Recall that

$$
\mathcal{N}\left(x ; \mu, \sigma^{2}\right)=\sqrt{\frac{1}{2 \pi \sigma^{2}}} \exp \left(-\frac{(x-\mu)^{2}}{2 \sigma^{2}}\right)
$$

For $D$-dimensional vector $\boldsymbol{x}$, the multivariate Gaussian distribution takes the form

$$
\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \stackrel{\text { def }}{=} \frac{1}{\sqrt{(2 \pi)^{D}|\Sigma|}} \exp \left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{T} \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)
$$

The biggest difference compared to the single-dimensional case is the covariance matrix $\boldsymbol{\Sigma}$, which is (in the non-degenerate case, which is the only one considered here) a symmetric positive-definite matrix of size $D \times D$.

If the covariance matrix is an identity, then the multivariate Gaussian distribution simplifies to

$$
\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{I})=\frac{1}{\sqrt{(2 \pi)^{D}}} \exp \left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{T}(\boldsymbol{x}-\boldsymbol{\mu})\right)
$$

We can rewrite the exponent in this case to

$$
\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{I}) \propto \exp \left(-\frac{\|\boldsymbol{x}-\boldsymbol{\mu}\|^{2}}{2}\right)
$$

Therefore, the constant surfaces are concentric hyperspheres (circles in 2D, spheres in 3D) centered at the mean $\boldsymbol{\mu}$.
The same holds if the covariance is $\sigma^{2} \boldsymbol{I}$, only the hyperspheres' diameter changes.


Now consider a diagonal covariance matrix $\boldsymbol{\Lambda}$. The exponent then simplifies to

$$
\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Lambda}) \propto \exp \left(-\sum_{i} \frac{1}{2 \boldsymbol{\Lambda}_{i, i}}\left(\boldsymbol{x}_{i}-\boldsymbol{\mu}_{i}\right)^{2}\right)
$$

The constant surfaces in this case are axis-aligned hyperellipsoids (ellipses in 2D, ellipsoids in 3D) centered at the mean $\boldsymbol{\mu}$ with size of the axes depending on the corresponding diagonal entries in the covariance matrix.


In the general case of a full covariance matrix, the fact that it is positive definite implies it has real positive eigenvalues $\lambda_{i}$. Considering the corresponding eigenvectors $\boldsymbol{u}_{i}$, it can be shown that the constant surfaces are again hyperellipsoids centered at $\boldsymbol{\mu}$, but this time rotated so that their axes are the eigenvectors $\boldsymbol{u}_{i}$ with sizes $\lambda_{i}^{1 / 2}$.


Generally, we can rewrite a positive-definite matrix $\boldsymbol{\Sigma}$ as $\boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^{T}=\left(\boldsymbol{U} \boldsymbol{\Lambda}^{1 / 2}\right)\left(\boldsymbol{U} \boldsymbol{\Lambda}^{1 / 2}\right)^{T}$, and then

$$
\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \Longleftrightarrow \boldsymbol{x} \sim \boldsymbol{\mu}+\boldsymbol{U} \mathbf{\Lambda}^{1 / 2} \mathcal{N}(0, \boldsymbol{I})
$$

Therefore, when sampling from a distribution with a full covariance matrix, we can sample from a standard multivariate $\mathcal{N}(0, \boldsymbol{I})$, scale by the eigenvalues of the covariance matrix, rotate according to the eigenvectors of the covariance matrix and finally shifting by $\boldsymbol{\mu}$.
Note that different forms of covariance allows more generality, but also requires more parameters:

- the $\sigma^{2} \boldsymbol{I}$ has a single parameter,
- the $\boldsymbol{\Lambda}$ has $D$ parameters,

(a)

(b)

(c)

Figure 2.8 of Pattern Recognition and Machine Learning.

- the full covariance matrix $\boldsymbol{\Sigma}$ has $\binom{D+1}{2}$ parameters, i.e., $\Theta\left(D^{2}\right)$.


## Gaussian Mixture

Let $\boldsymbol{x}_{1}, \boldsymbol{x}_{2}, \ldots, \boldsymbol{x}_{N}$ be a collection of $N$ input examples, each being a $D$-dimensional vector $\boldsymbol{x}_{i} \in \mathbb{R}^{D}$. Let $K$, the number of target clusters, be given.

Our goal is to represent the data as a Gaussian mixture, which is a combination of $K$ Gaussians in the form

$$
p(\boldsymbol{x})=\sum_{k=1}^{K} \pi_{k} \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}_{k}, \boldsymbol{\Sigma}_{k}\right)
$$

Therefore, each cluster is parametrized as $\mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}_{k}, \boldsymbol{\Sigma}_{k}\right)$.


Let $\boldsymbol{x}_{1}, \boldsymbol{x}_{2}, \ldots, \boldsymbol{x}_{N}$ be a collection of $N$ input examples, each being a $D$-dimensional vector $\boldsymbol{x}_{i} \in \mathbb{R}^{D}$. Let $K$, the number of target clusters, be given.

Our goal is to represent the data as a Gaussian mixture, which is a combination of $K$ Gaussians in the form

$$
p(\boldsymbol{x})=\sum_{k=1}^{K} \pi_{k} \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}_{k}, \boldsymbol{\Sigma}_{k}\right)
$$

Therefore, each cluster is parametrized as $\mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}_{k}, \boldsymbol{\Sigma}_{k}\right)$.
Let $\boldsymbol{z} \in\{0,1\}^{K}$ be a $K$-dimensional random variable, such that exactly one $z_{k}$ is 1 , denoting to which cluster a training example belongs. Let the marginal distribution of $z_{k}$ be

$$
p\left(z_{k}=1\right)=\pi_{k}
$$

so that the priors $\pi_{k}$ represent the "fertility" of the clusters. Then, $p(\boldsymbol{z})=\prod_{k} \pi_{k}^{z_{k}}$.

## Gaussian Mixture

We can write

$$
p(\boldsymbol{x})=\sum_{\boldsymbol{z}} p(\boldsymbol{z}) p(\boldsymbol{x} \mid \boldsymbol{z})=\sum_{k=1}^{K} \pi_{k} \mathcal{N}\left(\boldsymbol{x} \mid \boldsymbol{\mu}_{k}, \boldsymbol{\Sigma}_{k}\right)
$$

and the probability of the whole clustering is therefore

$$
\log p(\boldsymbol{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})=\sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_{k} \mathcal{N}\left(\boldsymbol{x}_{i} \mid \boldsymbol{\mu}_{k}, \boldsymbol{\Sigma}_{k}\right)\right) .
$$

To fit a Gaussian mixture model, we utilize maximum likelihood estimation and minimize

$$
\mathcal{L}(\boldsymbol{X})=-\sum_{i} \log \sum_{k=1}^{K} \pi_{k} \mathcal{N}\left(\boldsymbol{x}_{i} \mid \boldsymbol{\mu}_{k}, \boldsymbol{\Sigma}_{k}\right)
$$

## Gaussian Mixture

The derivative of the loss with respect to $\boldsymbol{\mu}_{k}$ gives

$$
\frac{\partial \mathcal{L}(\boldsymbol{X})}{\partial \boldsymbol{\mu}_{k}}=-\sum_{i} \frac{\pi_{k} \mathcal{N}\left(\boldsymbol{x}_{i} \mid \boldsymbol{\mu}_{k}, \boldsymbol{\Sigma}_{k}\right)}{\sum_{l=1}^{K} \pi_{l} \mathcal{N}\left(\boldsymbol{x}_{i} \mid \boldsymbol{\mu}_{l}, \boldsymbol{\Sigma}_{l}\right)} \boldsymbol{\Sigma}_{k}^{-1}\left(\boldsymbol{x}_{i}-\boldsymbol{\mu}_{k}\right)
$$

Denoting $r\left(z_{i, k}\right)=\frac{\pi_{k} \mathcal{N}\left(\boldsymbol{x}_{i} \mid \boldsymbol{\mu}_{k}, \boldsymbol{\Sigma}_{k}\right)}{\sum_{l=1}^{K} \pi_{l} \mathcal{N}\left(\boldsymbol{x}_{i} \mid \boldsymbol{\mu}_{l}, \boldsymbol{\Sigma}_{l}\right)}$, setting the derivative equal to zero and multiplying by $\boldsymbol{\Sigma}_{k}$, we get

$$
\boldsymbol{\mu}_{k}=\frac{\sum_{i} r\left(z_{i, k}\right) \boldsymbol{x}_{i}}{\sum_{i} r\left(z_{i, k}\right)} .
$$

The $r\left(z_{i, k}\right)$ are usually called responsibilities and denote the probability $p\left(z_{k}=1 \mid \boldsymbol{x}_{i}\right)$. Note that the responsibilities depend on $\boldsymbol{\mu}_{k}$, so the above equation is not an analytical solution for $\boldsymbol{\mu}_{k}$, but can be used as an iterative algorithm for converging to a local optimum.

For $\boldsymbol{\Sigma}_{k}$, we again compute the derivative of the loss, which is technically complicated (we need to compute a derivative of a matrix inverse, and also we need to differentiate matrix determinant) and results in an analogous equation:

$$
\boldsymbol{\Sigma}_{k}=\frac{\sum_{i} r\left(z_{i, k}\right)\left(\boldsymbol{x}_{i}-\boldsymbol{\mu}_{k}\right)\left(\boldsymbol{x}_{i}-\boldsymbol{\mu}_{k}\right)^{T}}{\sum_{i} r\left(z_{i, k}\right)}
$$

To minimize the loss with respect to $\boldsymbol{\pi}$, we need to include the constraint $\sum_{k} \pi_{k}=1$, so we form a Lagrangian $\mathcal{L}(\boldsymbol{X})-\lambda\left(\sum_{k} \pi_{k}-1\right)$, and get

$$
\frac{\partial \mathcal{L}(\boldsymbol{X})}{\partial \pi_{k}}=\sum_{i} \frac{\mathcal{N}\left(\boldsymbol{x}_{i} \mid \boldsymbol{\mu}_{k}, \boldsymbol{\Sigma}_{k}\right)}{\sum_{l=1}^{K} \pi_{l} \mathcal{N}\left(\boldsymbol{x}_{i} \mid \boldsymbol{\mu}_{l}, \boldsymbol{\Sigma}_{l}\right)}-\lambda
$$

Setting the derivative to zero and multiplying it by $\pi_{k}$, we obtain $\pi_{k}=\frac{1}{\lambda} \cdot \sum_{i} r\left(z_{i, k}\right)$, so

$$
\pi_{k}=1 / N \cdot \sum_{i} r\left(z_{i, k}\right)
$$

Input: Input points $\boldsymbol{x}_{1}, \ldots, \boldsymbol{x}_{N}$, number of clusters $K$.

- Initialize $\boldsymbol{\mu}_{k}, \boldsymbol{\Sigma}_{k}$ and $\pi_{k}$. It is common to start by running the K-Means algorithm to obtain $z_{i, k}$, set $r\left(z_{i, k}\right) \leftarrow z_{i, k}$ and use the $\mathbf{M}$ step below.
- Repeat until convergence (or until patience runs out):
- E step. Evaluate the responsibilities as

$$
r\left(z_{i, k}\right)=\frac{\pi_{k} \mathcal{N}\left(\boldsymbol{x}_{i} \mid \boldsymbol{\mu}_{k}, \boldsymbol{\Sigma}_{k}\right)}{\sum_{l=1}^{K} \pi_{l} \mathcal{N}\left(\boldsymbol{x}_{i} \mid \boldsymbol{\mu}_{l}, \boldsymbol{\Sigma}_{l}\right)}=p\left(z_{k}=1 \mid \boldsymbol{x}_{i}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}\right)
$$

- M step. Maximize the log-likelihood by setting

$$
\boldsymbol{\mu}_{k}=\frac{\sum_{i} r\left(z_{i, k}\right) \boldsymbol{x}_{i}}{\sum_{i} r\left(z_{i, k}\right)}, \quad \boldsymbol{\Sigma}_{k}=\frac{\sum_{i} r\left(z_{i, k}\right)\left(\boldsymbol{x}_{i}-\boldsymbol{\mu}_{k}\right)\left(\boldsymbol{x}_{i}-\boldsymbol{\mu}_{k}\right)^{T}}{\sum_{i} r\left(z_{i, k}\right)}, \pi_{k}=\frac{\sum_{i} r\left(z_{i, k}\right)}{N} .
$$







## EM Algorithm

The algorithm for estimating the Gaussian mixture is an example of an EM algorithm. The EM algorithm algorithm can be used when given a joint distribution $p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{w})$ over observed variables $\boldsymbol{X}$ and latent (hidden, unseen) variables $\boldsymbol{Z}$, parametrized by $\boldsymbol{w}$, we maximize

$$
\log p(\boldsymbol{X} ; \boldsymbol{w})=\log \left(\sum_{\boldsymbol{Z}} p(\boldsymbol{X}, \boldsymbol{Z} ; \boldsymbol{w})\right)
$$

with respect to $\boldsymbol{w}$.
Usually, the latent variables $\boldsymbol{Z}$ indicate membership of the data in one of the set of groups.
The main idea is to replace the computation of the logarithm of the sum over all latent variable values by the expectation of a logarithm of the joint probability under the posterior latent variable distribution $p(\boldsymbol{Z} \mid \boldsymbol{X} ; \boldsymbol{w})$.

- Initialize the parameters $\boldsymbol{w}^{\text {new }}$.
- Repeat until convergence (or until patience runs out):
$\circ \boldsymbol{w}^{\text {old }} \leftarrow \boldsymbol{w}^{\text {new }}$
- E step. Evaluate

$$
Q\left(\boldsymbol{w} \mid \boldsymbol{w}^{\text {old }}\right)=\mathbb{E}_{\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{w}^{\text {old }}}[\log p(\boldsymbol{X}, \boldsymbol{Z} ; \boldsymbol{w})]
$$

- M step. Maximize the log-likelihood by computing

$$
\boldsymbol{w}^{\text {new }} \leftarrow \underset{\boldsymbol{w}}{\arg \max } Q\left(\boldsymbol{w} \mid \boldsymbol{w}^{\text {old }}\right) .
$$

## EM Algorithm - Proof

The EM algorithm updates $\boldsymbol{w}$ to maximize $Q\left(\boldsymbol{w} \mid \boldsymbol{w}^{\text {old }}\right)$ on every step, and we now prove that this update of weights also causes the $\log p(\boldsymbol{X} ; \boldsymbol{w})$ to increase.


First note that for any $\boldsymbol{Z}$ with nonzero probability, we can write

$$
\log p(\boldsymbol{X} ; \boldsymbol{w})=\log p(\boldsymbol{X}, \boldsymbol{Z} ; \boldsymbol{w})-\log p(\boldsymbol{Z} \mid \boldsymbol{X} ; \boldsymbol{w})
$$

Computing the expectation with respect to $p\left(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{w}^{\text {old }}\right)$, we get

$$
\begin{aligned}
\log p(\boldsymbol{X} ; \boldsymbol{w}) & =\sum_{\boldsymbol{Z}} p\left(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{w}^{\text {old }}\right) \log p(\boldsymbol{X}, \boldsymbol{Z} ; \boldsymbol{w})-\sum_{\boldsymbol{Z}} p\left(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{w}^{\text {old }}\right) \log p(\boldsymbol{Z} \mid \boldsymbol{X} ; \boldsymbol{w}) \\
& =Q\left(\boldsymbol{w} \mid \boldsymbol{w}^{\text {old }}\right)+H\left(p\left(\boldsymbol{Z} \mid \boldsymbol{X} ; \boldsymbol{w}^{\text {old }}\right), p(\boldsymbol{Z} \mid \boldsymbol{X} ; \boldsymbol{w})\right)
\end{aligned}
$$

The above equation holds for any $\boldsymbol{w}$, so also for $\boldsymbol{w}^{\text {old }}$ :

$$
\log p\left(\boldsymbol{X} ; \boldsymbol{w}^{\text {old }}\right)=Q\left(\boldsymbol{w}^{\text {old }} \mid \boldsymbol{w}^{\text {old }}\right)+H\left(p\left(\boldsymbol{Z} \mid \boldsymbol{X} ; \boldsymbol{w}^{\text {old }}\right), p\left(\boldsymbol{Z} \mid \boldsymbol{X} ; \boldsymbol{w}^{\text {old }}\right)\right)
$$

Subtracting the second term $\log p\left(\boldsymbol{X} ; \boldsymbol{w}^{\text {old }}\right)$ from the first $\log p(\boldsymbol{X} ; \boldsymbol{w})$, we obtain

$$
\begin{aligned}
& \log p(\boldsymbol{X} ; \boldsymbol{w})-\log p\left(\boldsymbol{X} ; \boldsymbol{w}^{\text {old }}\right) \\
& =Q\left(\boldsymbol{w} \mid \boldsymbol{w}^{\text {old }}\right)-Q\left(\boldsymbol{w}^{\text {old }} \mid \boldsymbol{w}^{\text {old }}\right)+H\left(p\left(\boldsymbol{Z} \mid \boldsymbol{X} ; \boldsymbol{w}^{\text {old }}\right), p(\boldsymbol{Z} \mid \boldsymbol{X} ; \boldsymbol{w})\right)-H\left(p\left(\boldsymbol{Z} \mid \boldsymbol{X} ; \boldsymbol{w}^{\text {old }}\right)\right) \\
& =Q\left(\boldsymbol{w} \mid \boldsymbol{w}^{\text {old }}\right)-Q\left(\boldsymbol{w}^{\text {old }} \mid \boldsymbol{w}^{\text {old }}\right)+D_{\mathrm{KL}}\left(p\left(\boldsymbol{Z} \mid \boldsymbol{X} ; \boldsymbol{w}^{\text {old }}\right) \| p(\boldsymbol{Z} \mid \boldsymbol{X} ; \boldsymbol{w})\right)
\end{aligned}
$$

Given that KL divergence is non-negative, we get

$$
\log p(\boldsymbol{X} ; \boldsymbol{w})-\log p\left(\boldsymbol{X} ; \boldsymbol{w}^{\text {old }}\right) \geq Q\left(\boldsymbol{w} \mid \boldsymbol{w}^{\text {old }}\right)-Q\left(\boldsymbol{w}^{\text {old }} \mid \boldsymbol{w}^{\text {old }}\right)
$$

so if $\arg \max _{\boldsymbol{w}} Q\left(\boldsymbol{w} \mid \boldsymbol{w}^{\text {old }}\right)$ is larger than $Q\left(\boldsymbol{w}^{\text {old }} \mid \boldsymbol{w}^{\text {old }}\right)$, we also increase $\log p(\boldsymbol{X} ; \boldsymbol{w})$.
To show that $\log p(\boldsymbol{X} ; \boldsymbol{w})$ actually converges to a stationary point, some additional regularity conditions are needed (one possibility is to require $Q\left(\boldsymbol{w} \mid \boldsymbol{w}^{\text {old }}\right)$ to be continuous in both $\boldsymbol{w}$ and $\left.\boldsymbol{w}^{\text {old }}\right)$. For a more detailed treatment, see the 1983 paper On the Convergence Properties of the EM Algorithm by C. F. Jeff Wu.

Consider a model $y(\boldsymbol{x})$ solving a regression problem with MSE loss

$$
\mathcal{L}=\mathbb{E}_{\boldsymbol{x}, t}\left[(y(\boldsymbol{x})-t)^{2}\right] .
$$

Denoting $g(\boldsymbol{x}) \stackrel{\text { def }}{=} \mathbb{E}_{t \mid \boldsymbol{x}}[t]$, we can rewrite $(y(\boldsymbol{x})-t)^{2}$ as

$$
\begin{aligned}
(y(\boldsymbol{x})-t)^{2} & =(y(\boldsymbol{x})-g(\boldsymbol{x})+g(\boldsymbol{x})-t)^{2} \\
& =(y(\boldsymbol{x})-g(\boldsymbol{x}))^{2}+2(y(\boldsymbol{x})-g(\boldsymbol{x}))(g(\boldsymbol{x})-t)+(g(\boldsymbol{x})-t)^{2}
\end{aligned}
$$

When computing an expectation with respect to $p_{\text {data }}(\boldsymbol{x}, t)$, we obtain

$$
\begin{aligned}
\mathcal{L} & =\mathbb{E}_{\boldsymbol{x}, t}\left[(y(\boldsymbol{x})-g(\boldsymbol{x}))^{2}\right]+2 \mathbb{E}_{\boldsymbol{x}, t}[(y(\boldsymbol{x})-g(\boldsymbol{x}))(g(\boldsymbol{x})-t)]+\mathbb{E}_{\boldsymbol{x}, t}\left[(g(\boldsymbol{x})-t)^{2}\right] \\
& =\mathbb{E}_{\boldsymbol{x}, t}\left[(y(\boldsymbol{x})-g(\boldsymbol{x}))^{2}\right]+\mathbb{E}_{\boldsymbol{x}, t}\left[(g(\boldsymbol{x})-t)^{2}\right],
\end{aligned}
$$

because $\mathbb{E}_{t \mid \boldsymbol{x}}[g(\boldsymbol{x})-t]=0$.

We have decomposed the loss into two components, where the second is the "label noise" called irreducible error.
We now further decompose the first component $\mathbb{E}_{\boldsymbol{x}, t}\left[(y(\boldsymbol{x})-g(\boldsymbol{x}))^{2}\right]$.
Assuming $\mathcal{D}$ is a dataset obtained from the data generating distribution, we denote the prediction of a model trained using this dataset as $y(\boldsymbol{x} ; \mathcal{D})$.

$$
\begin{aligned}
(y(\boldsymbol{x} ; \mathcal{D})-g(\boldsymbol{x}))^{2} & =\left(y(\boldsymbol{x} ; \mathcal{D})-\mathbb{E}_{\mathcal{D}}[y(\boldsymbol{x} ; \mathcal{D})]+\mathbb{E}_{\mathcal{D}}[y(\boldsymbol{x} ; \mathcal{D})]-g(\boldsymbol{x})\right)^{2} \\
& =\left(y(\boldsymbol{x} ; \mathcal{D})-\mathbb{E}_{\mathcal{D}}[y(\boldsymbol{x} ; \mathcal{D})]\right)^{2} \\
& +2\left(y(\boldsymbol{x} ; \mathcal{D})-\mathbb{E}_{\mathcal{D}}[y(\boldsymbol{x} ; \mathcal{D})]\right)\left(\mathbb{E}_{\mathcal{D}}[y(\boldsymbol{x} ; \mathcal{D})]-g(\boldsymbol{x})\right) \\
& +\left(\mathbb{E}_{\mathcal{D}}[y(\boldsymbol{x} ; \mathcal{D})]-g(\boldsymbol{x})\right)^{2} .
\end{aligned}
$$

Note that $\mathbb{E}_{\mathcal{D}}\left[y(\boldsymbol{x} ; \mathcal{D})-\mathbb{E}_{\mathcal{D}}[y(\boldsymbol{x} ; \mathcal{D})]\right]=0$, therefore, for a given $\boldsymbol{x}$, we have

$$
\mathbb{E}_{\mathcal{D}}\left[(y(\boldsymbol{x} ; \mathcal{D})-g(\boldsymbol{x}))^{2}\right]=\mathbb{E}_{\mathcal{D}}\left[\left(y(\boldsymbol{x} ; \mathcal{D})-\mathbb{E}_{\mathcal{D}}[y(\boldsymbol{x} ; \mathcal{D})]\right)^{2}\right]+\left(\mathbb{E}_{\mathcal{D}}[y(\boldsymbol{x} ; \mathcal{D})]-g(\boldsymbol{x})\right)^{2}
$$

Putting all the parts together, we get that

$$
\begin{aligned}
\mathbb{E}_{\mathcal{D}}[\mathcal{L}] & =\mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{\boldsymbol{x}, t}\left[(y(\boldsymbol{x} ; \mathcal{D})-t)^{2}\right]\right] \\
& =\mathbb{E}_{\boldsymbol{x}, t}[\underbrace{\left(\mathbb{E}_{\mathcal{D}}[y(\boldsymbol{x} ; \mathcal{D})]-g(\boldsymbol{x})\right)^{2}}_{\text {bias }^{2}}+\underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(y(\boldsymbol{x} ; \mathcal{D})-\mathbb{E}_{\mathcal{D}}[y(\boldsymbol{x} ; \mathcal{D})]\right)^{2}\right]}_{\text {variance }}+\underbrace{(g(\boldsymbol{x})-t)^{2}}_{\text {irreducible error }}]
\end{aligned}
$$

This is the so-called bias-variance trade-off, showing that the expected loss decomposes into the three above components.
For classification problems, we can use the same decomposition on MSE of the probabilities, and it is also possible to derive an analogy using the so-called 0-1 loss (see A Unified BiasVariance Decomposition by P. Domingos for the exact formulation).
This decomposition has been long interpreted as:
The price to pay for achieving low bias is high variance.




For a k-NN search, when we consider an expectation over all possible labelings of a fixed training set, the MSE decomposes as

$$
\mathbb{E}\left[(y(\boldsymbol{x})-t(\boldsymbol{x}))^{2}\right]=\left(t(\boldsymbol{x})-\frac{1}{K} \sum_{k=1}^{K} t\left(N_{k}(\boldsymbol{x})\right)\right)^{2}+\frac{\sigma^{2}}{K}+\sigma^{2}
$$

where $N_{k}(\boldsymbol{x})$ is the k -nearest neighbor of $\boldsymbol{x}$ and $\sigma^{2}$ is the irreducible error.


Quoting from Neural Networks and the Bias/Variance Decomposition by S. Geman, 1992:
The basic trend is what we expect: bias falls and variance increases with the number of hidden units. The effects are not perfectly demonstrated (notice, for example, the dip in variance in the experiments with the largest numbers of hidden units), presumably because the phenomenon of overfitting is complicated by convergence issues and perhaps also by our decision to stop the training prematurely.

However, in past years, neural networks with increasing capacity have performed better and better.

Traditional view of bias-variance


Worst-case analysis


Measure concentrates
Figure 4.1 of the paper "On the Bias-Variance Tradeoff: Textbooks Need an Update" by B. Neal.


Capacity of $\mathcal{H}$
(a)

(b)

Figure 1: Curves for training risk (dashed line) and test risk (solid line). (a) The classical U-shaped risk curve arising from the bias-variance trade-off. (b) The double descent risk curve, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high capacity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

## Double Descent - Overparametrized with Minimum L2



Figure 3 of paper "Reconciling modern machine learning practice and the bias-variance trade-off" by M. Belkin et al.





Figure 1: Left: Train and test error as a function of model size, for ResNet18s of varying width on CIFAR-10 with $15 \%$ label noise. Right: Test error, shown for varying train epochs. All models trained using Adam for 4 K epochs. The largest model (width 64) corresponds to standard ResNet18.

## Deep Double Descent



Figure 2: Left: Test error as a function of model size and train epochs. The horizontal line corresponds to model-wise double descent-varying model size while training for as long as possible. The vertical line corresponds to epoch-wise double descent, with test error undergoing double-descent as train time increases. Right Train error of the corresponding models. All models are Resnet18s trained on CIFAR-10 with $15 \%$ label noise, data-augmentation, and Adam for up to 4 K epochs.

Figure 2 of the paper "Deep Double Descent: Where Bigger Models and More Data Hurt" by P. Nakkiran et al.

(a) CIFAR-100. There is a peak in test error even with no label noise.

(b) CIFAR-10. There is a "plateau" in test error around the interpolation point with no label noise, which develops into a peak for added label noise.

